Smarte Städtebauliche Objekte für eine adaptive Stadt: Ein Verfahren der Künstlichen Intelligenz zur Erhöhung der Wohlfahrt

Dissertation zur Erlangung des Doktorgrades
der Wirtschaftswissenschaften (Dr. oec.)

Fakultät Wirtschafts- und Sozialwissenschaften
Universität Hohenheim

Institut für Health Care & Public Management

vorgelegt von
Marvin Hubl

aus Sindelfingen

2021

Datum der mündlichen Prüfung:	16. Juni 2021
Erstgutachter:	Prof. Dr. Stefan Kirn
Zweitgutachter:	Prof. Dr. Bogdan Franczyk
Vorsitzender der Prüfungskommission:	Prof. Dr. Christian Ernst
Dekan:	Prof. Dr. Karsten Hadwich

Für „eine bessere Bewältigung unseres Lebens“ und
„Frieden ohne Armut“
(nach Paul Lorenzen, 2000, S. 9 und 1994, S. 129).

Gewidmet meiner lieben Frau, meinen Eltern und
besonders den Kindern.

Danksagung

Allen, die zum Gelingen meiner Promotion beitrugen, möchte ich meinen Dank zum Ausdruck bringen.

Ein besonderer Dank gilt meinem Doktorvater Herrn Prof. Dr. Stefan Kirn. Er unterstützte mein Promotionsvorhaben und förderte mich darin, eigene Ansätze und Ideen zu entwickeln, diese aktiv zu verfolgen sowie – nicht unwichtig – auch aus vermeintlichen Rückschlägen wertvolle Erkenntnisse zu gewinnen.

Herrn Prof. Dr. Franczyk danke ich für die freundliche Übernahme des Zweitgutachtens und Herrn Prof. Dr. Ernst für die angenehme Leitung der Disputation.

Bei allen, mit denen ich am Lehrstuhl Wirtschaftsinformatik 2 zusammenarbeiten durfte, bedanke ich mich für die gute Kollegialität und den regen wissenschaftlichen Austausch.

Darüber hinaus geht mein Dank an alle Projektmitarbeiterinnen und -mitarbeiter des Projektes UrbanLife+ für die respektvolle Zusammenarbeit und die Gastfreundschaft bei den Projekttreffen.

Schließlich gebührt meiner Familie Dank für ihren bedingungslosen Rückhalt.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
Bsp.	Beispiel
bspw.	beispielsweise
bzgl.	bezüglich
bzw.	beziehungsweise
ca.	circa (ungefähr)
Def.	Definition
d. h.	das heißt
ebd.	ebenda
EKG	Elektrokardiogramm
engl.	englisch
et al.	et alii/aliae (und andere)
f.	folgende
ff.	fortfolgende
FIFO	first in, first out (Warteschlangenverarbeitung)
gdw.	genau dann, wenn (Äquivalenzbeziehung)
GE	Geldeinheiten
gem.	gemäß
ggf.	gegebenenfalls
i. Allg.	im Allgemeinen

i. d. R.	in der Regel
i. e. S.	im engeren Sinne
inkl.	inklusive
insb.	insbesondere
i. S. v.	im Sinne von
IT	Informationstechnologie
IoT	Internet of Things (Internet der Dinge)
i. w. S.	im weiteren Sinne
Kap.	Kapitel
KI	Künstliche Intelligenz
LED	Light emitting diode
Nr.	Nummer
o. Ä.	oder Ähnliches
o. g.	oben genannt
OECD	Organisation for Economic Co-operation and Development
ÖPNV	öffentlicher Personennahverkehr
PAVK	periphere arterielle Verschlusskrankheit
PC	Personal Computer
RFID	Radio-Frequency Identification
S.	Seite
s. u.	siehe unten
sog.	sogenannt
tlw.	teilweise
SSO	smartes städtebauliches Objekt

u.	und
u. Ä.	und Ähnliches
usw.	und so weiter
v. a.	vor allem
VDI	Verein deutscher Ingenieure
vgl.	vergleiche
v. v.	vice versa (umgekehrt)
z. B.	zum Beispiel

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Hinweis: Das Symbolverzeichnis gilt ab Kap. 3.

$agg(\hat{\mathbf{s}})$	Aggregierungsfunktion für die Safety-Bewertungen.
B	Belegungsmatrix.
B_τ	Bis zum Diskursweltzustand τ bekannte Belegung.
$b_{i,j}$	Eintrag in $B : b_{i,j} = 1$, wenn p_i Sitzgelegenheit j belegt, sonst $b_{i,j} = 0$.
$\text{card}(\mathcal{M})$	Kardinalität einer Menge $\mathcal{M}$ (Anzahl der Elemente in $\mathcal{M}$).
C	Oberer Schranke für die Anzahl an Expansionen.
c	Zahl der Expansionen.
d_i^{max}	Maximale komfortable Gehdistanz von Passant i.
$d_i^{fatigued}(\tau)$	Kumulierte Distanz, die p_i bis τ mit Pausenerfordernis zurücklegte.
$d_i^{togo}(\tau)$	Bis zur nächsten Pause nach τ von p_i noch zu gehende Distanz.
$d_i^{went}(\tau)$	Seit letzter Pause vor τ von Passant i zurückgelegte Distanz.
dir_i	Gehrichtung des Passanten i.
$e(\tau)$	Heuristische Evaluationsfunktion: $g(\tau) + h(\tau)$.
$f_i(\tau)$	Ermüdungsgrad von Passant i im Diskursweltzustand τ.
$\dot{f}_i$	Ermüdungsrate von Passant i.
$g(\tau)$	Bewertung des Pfades vom Start bis zum Zustand τ.

$h(\tau)$	Die Heuristik: Schätzbewertung des Pfades vom Zustand τ zum Ziel.
i	Laufvariable, die einen Passanten referenziert ($i \mapsto p_i$).
j	Laufvariable für adaptive Sitzgelegenheiten bzw. Wegpunkte.
k	Anzahl der adaptiven Sitzgelegenheiten.
n	Anzahl der Passanten.
$\underline{occ}$	Anzahl der Passanten, die auf einer spezifizierten Sitzgelegenheit sitzen, wenn ein spezifizierter Passant dort ankommt.
$\overline{occ}$	Anzahl der Passanten, die auf einer spezifizierten Sitzgelegenheit sitzen, wenn ein spezifizierter Passant von dort losgeht.
$occ_j(\tau)$	Menge der Passanten, die im Diskursweltzustand τ Sitzgelegenheit j benutzen.
$Passanten$	Menge der Passanten.
p_i	Passant i.
$pos(j)$	Position des Wegpunktes j.
$pos_i(\tau)$	Position des Passanten i im Diskursweltzustand τ.
$q_i(\tau)$	Bewegungszustand von Passant i im Diskursweltzustand τ.
r_i	Dauer, die Passant i auf Sitzgelegenheiten verweilt.
$\hat{\mathbf{s}}$	Safety-Bewertungsvektor mit den Komponenten $\hat{s}_i = \hat{s}(i, B)$.
$\hat{s}(i, B)$	Safety-Bewertung des Weges für Passant i bei Belegung B.
$s(j, i, B)$	Safety-Funktion (zusammengesetzt aus $s_{(a)}$ und $s_{(b)}$, s. u.).
$s_{(a)}(\cdot)$	Safety-Funktion für erfordernisgenaue Verfügbarkeit. Für den Platzhalter $(\cdot)$ einsetzbare Argumente: (j, i, B) , $\Big(z_i(\tau)\Big)$ bzw. (z).

$s_{(b)}(\cdot)$	Safety-Funktion für Hinsetz- und Aufstehassistenz. Für $(\cdot)$ einsetzbar: (j, i, B) oder $(\underline{occ}, \overline{occ})$.
$SSOs$	Menge der adaptiven Sitzgelegenheiten (als SSOs).
t_i^{max}	Maximale komfortable Gehdauer von Passant i.
v_i	Gehgeschwindigkeit von Passant i.
w_{li} , w_{re}	Wendestellen der Safety-Funktion $s_{(a)}$.
$x_i(\tau)$	Normalisierte Distanz seit der letzten Pause von Passant i.
$y_i(\tau)$	Normalisierte Distanz bis zur nächsten Sitzgelegenheit für p_i.
$z_i(\tau)$ bzw. z	Maß für erfordernisgenaue Verfügbarkeit einer Sitzgelegenheit.
γ^{Auf} , γ^{Hin}	Safety-Bewertung für Aufsteh- bzw. Hinsetzassistenz.
κ	Kapazität der Sitzgelegenheiten.
$\lambda_{(a)}$, $\lambda_{(b)}$	Gewichtungsfaktoren für $s_{(a)}$ bzw. $s_{(b)}$.
$\xi_\kappa(B)$	Kapazitätsprüffunktion: $\xi_\kappa(B) = 1$, wenn keine Sitzgelegenheit überbelegt wird.
τ	Ein Diskursweltzustand.
τ_{init} , τ_{ziel}	Ein Start- bzw. ein Zielzustand.
T	Logischer Vorgängerzustand von τ.

1 Einleitung

1.1 Gegenstand

Gegenstand der vorliegenden Arbeit ist der urbane Raum, maßgeblich charakterisiert durch seine städtebaulichen Objekte (wie z. B. Parkbänke oder Straßenlaternen). Die betrachteten Nutzer[1] des urbanen Raums sind Passanten, d. h. Fußgänger.[2] Die Arbeit ordnet sich in die Zielstellung für *Smart Cities* ein: Im Kern steht die Verbesserung des urbanen Lebens durch Informationstechnologie (IT), bezogen auf Wohlfahrt und Inklusion (Brandt et al., 2016; Neirotti et al., 2014).

Städtebauliche Objekte werden durch Ausstattung mit Informationsverarbeitungskapazitäten, Sensorik und Aktuatorik in neuartige sog. *smarte* städtebauliche Objekte (SSOs) transformiert (vgl. z. B. Okada et al., 2016; Kinch et al., 2014; Poulsen et al., 2013). Unter Aktuatorik werden hier sowohl Bauteile gefasst, die Translations- oder Rotationsbewegungen erzeugen (vgl. Rayes und Salam, 2017, S. 71 f.), als auch nicht-bewegliche Komponenten, die ihren wahrnehmbaren Zustand ändern, um auf ihre Umgebung einzuwirken; z. B. durch visuelle Signale. Um zweckmäßig mit der urbanen Umgebung zu interagieren, sind die SSOs mit einem *intelligenten Verhaltensmodell* versehen. SSOs können dann eine über simple Stimulus-Response-Funktionalität hinausgehende *Adaptivität des urbanen Raums* realisieren.

Besonders stark ausgeprägte Anforderungen an eine Adaptivität des urbanen Raums stellen die Erfordernisse von Passanten mit alterstypischen[3] Beeinträchtigungen – wobei im Rahmen der vorliegenden Arbeit körperliche Beeinträchtigungen fokussiert werden. Es lässt sich formulieren, dass die alterstypischen Beeinträchtigungen in körperlicher Hinsicht die Toleranz gegenüber einem nicht individuell angepassten urbanen Raum reduzieren.

[1] Soweit nicht anders angegeben, bezieht sich in der vorliegenden Arbeit der verwendete grammatikalische Genus bezogen auf Personen stets gleichzeitig sowohl auf weibliche als auch auf männliche Personen, unabhängig davon, ob die feminine oder die maskuline Form verwendet wird.

[2] Im Rahmen dieser Arbeit sind Personen stets Passanten.

[3] Mit „alterstypisch“ ist hier „für ein Alter ab ca. 65 Jahren typisch“ gemeint.

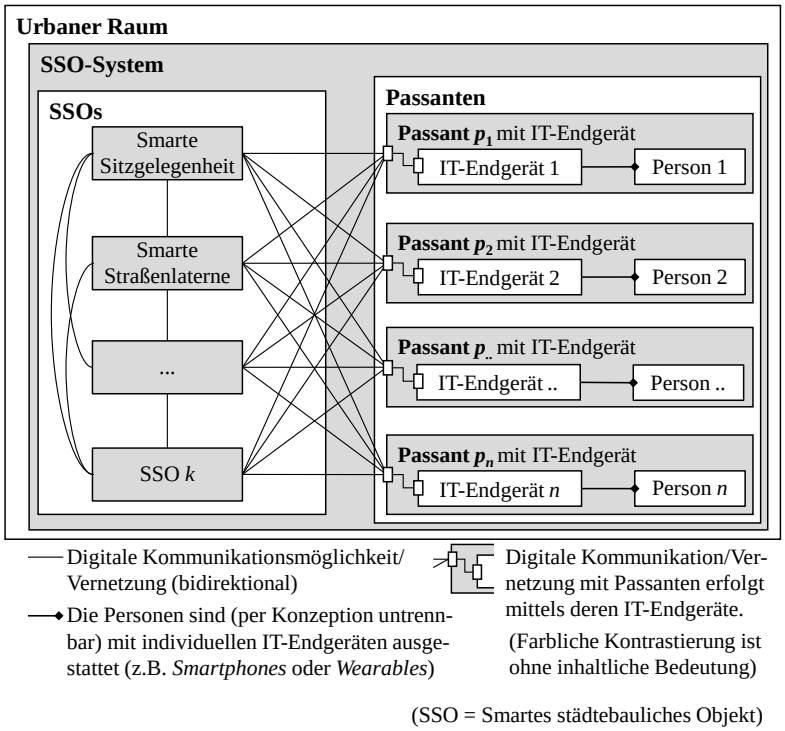


Abbildung 1.1: System smarter städtebaulicher Objekte

Abbildung 1.1 stellt das als Gegenstand der Arbeit betrachtete System smarter städtebaulicher Objekte dar. Das SSO-System besteht dabei stets sowohl aus SSOs als auch den Passanten, da sie konzeptionell nicht unabhängig voneinander betrachtet werden können.[4] Die Darstellung als Vollvermaschung ist als logische Vernetzung zu verstehen, die technisch aber auch über einen zentralen Knoten erfolgen kann. Passanten sind mittels IT-Endgeräte, wie *Smartphones* oder *Wearables*, vernetzt, die per Annahme eineindeutig, untrennbar einzelnen Personen zugeordnet sind.

1.2 Problemstellung

Die Problemstellung bezieht sich auf die Frage, wie mittels SSOs *Safety* für Passanten mit alterstypischen, körperlichen Beeinträchtigungen erhöht werden kann. Safety stellt dabei den durch die Funktionalität der SSOs realisier-

[4] Das konzeptionelle Problem dahinter ist in der Multiagentenforschung diskutiert (vgl. Ferber, 2001, S. 32; Kirn, 1996, S. 115, 153 ff.).

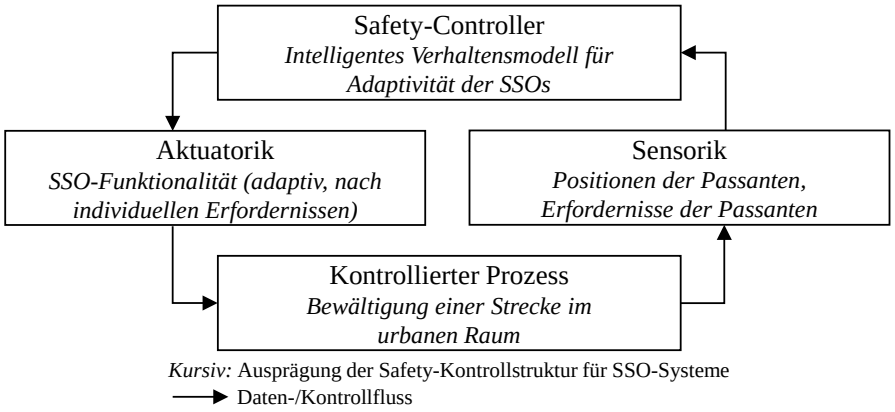


Abbildung 1.2: Safety-Kontrollstruktur (nach Leveson, 2011, S. 66, 93)

ten Nutzen dar, welcher hier auf die *sichere Benutzbarkeit* des urbanen Raumes abzielt. Dazu wird Safety als Zielkriterium für die Konstruktion Software-intensiver sozio-technischer Systeme in den Mittelpunkt gestellt (vgl. Leveson, 2011, S. 211). Der urbane Raum mit Passanten und städtebaulichen Objekten ist ein sozio-technisches System, welches durch die SSOs Software-intensiv wird.

Gemäß Leveson (ebd., S. 66) liegt eine Safety-Kontrollstruktur zugrunde, in welcher ein sog. Safety-Controller einen Prozess kontrolliert. Der Safety-Controller erfasst bestimmte Daten des kontrollierten Prozesses mittels Sensoren und steuert Aktuatoren, um auf den kontrollierten Prozess einzuwirken. Abbildung 1.2 stellt diese Kontrollstruktur im Allgemeinen dar und macht durch die kursiv gesetzten Ergänzungen kenntlich, wie sich der Gegenstand der Arbeit dort einordnet. Die Problemstellung der Arbeit bezieht sich im Kern auf die Konstruktion eines Safety-Controllers mit Methoden der Künstlichen Intelligenz (KI).

Leveson (ebd.) bietet einen Ansatz, um systematisch einen Gestaltungsspielraum – bzw. Konstruktionslösungsraum – zur Erhöhung von Safety zu identifizieren. Als zentraler Ansatzpunkt dient die Konzeption, dass ein Safety-Vorfall *genau dann* vorliegt, *wenn* ein kritischer Personenzustand und ein kritischer Umweltzustand zusammenkommen (vgl. ebd., S. 184). Kritischer Personen- und Umweltzustand stehen dabei in dualistischer Beziehung zueinander, weil das eine nicht ohne das andere zu definieren ist. Der Lösungsraum ergibt sich dann aus den Möglichkeiten, die kritischen Personenzustände oder die kritischen Umweltzustände zu verhindern. Die Forschungsfrage für die vorliegende Arbeit lautet:

Wie ist ein intelligentes Verhaltensmodell für SSOs zu konstruieren, um Safety-orientierte Adaptivität des urbanen Raumes zu bewirken?

1.3 Perspektive

Die Problemstellung wird aus einer *Engineering*-Perspektive betrachtet. Das angestrebte Gesamtergebnis besteht in der Herstellung Safety-orientierter Adaptivität des urbanen Raumes. Zu diesem Zweck weist die Engineering-Aufgabe drei Teile auf:

(1.) Als notwendige Voraussetzung für das angestrebte Gesamtergebnis müssen einzelne städtebauliche Objekte mittels Ansätzen des Internet of Things (IoT) und der KI zu Adaptivität befähigt werden. Moderne Sensor- und Aktuatortechnologien schaffen diese Voraussetzungen. Dennoch sind städtebauliche Objekte gegenwärtig überwiegend statische Objekte. Die informationstechnische Befähigung zu Adaptivität städtebaulicher Objekte stellt eine wesentliche Innovation gegenüber dem Status quo dar.

(2.) Als weitere notwendige Voraussetzung für das angestrebte Gesamtergebnis müssen die einzelnen städtebaulichen Objekte zu einem intelligenten Verbund vernetzt werden. Die Voraussetzung für die Vernetzung wird ebenfalls durch IoT-Technologien geschaffen (Hammi et al., 2018; Zanella et al., 2014): SSOs stellen dann Objekte im IoT dar.

(3.) Der im Kern dieser Arbeit stehende Beitrag liegt in der Entwicklung eines maximal Safety-sicherstellenden Koordinationsverfahrens für den intelligenten SSO-Verbund, basierend auf wohlfahrtsökonomischen Modellen. Denn unter der zu treffenden Annahme, dass die Nachfrage zur Nutzung spezifischer SSO-Funkionalität regelmäßig deren Verfügbarkeit übersteigt, ergibt sich ein Koordinationsproblem aufgrund begrenzter Ressourcen (Ferber, 2001, S. 433). Hierzu sind wohlfahrtsökonomische Rationalitätskalküle adäquat abzubilden. Kern ist die prozedurale *Erzeugung* entsprechender Rationalität als normative Konzeption intelligenten Verhaltens (vgl. Russell und Norvig, 2003, S. 1 f.).[5]

Das SSO-System bringt intelligentes Verhalten auf Verbundebene hervor, indem es die einzelnen SSOs nach gewissen Rationalitätskalkülen koordiniert. Ferber (2001, S. 436) beschreibt, dass für Koordinationsprobleme i. Allg. nicht bekannt ist, wie diese gelöst werden und dass dies oftmals heuristische Methoden erfordert. Die vorliegende Arbeit liefert hierzu einen Beitrag.

[5] In der „reinen“ Ökonomik kann mitunter von prozeduralen Fragen zum Finden rationaler Lösungen abstrahiert werden (vgl. Boutilier et al., 1997).

1.4 Aufbau der Arbeit

- *Kapitel 1* legt den Forschungsansatz für die vorliegende Arbeit dar.
- *Kapitel 2* analysiert die Ausgangssituation, das Problem, Anforderungen und Lösungsansätze.
- *Kapitel 3* entwickelt das Verhaltensmodell für intelligentes Verbundverhalten der SSOs.
- *Kapitel 4* validiert das Verfahren durch eine Computer-Simulation und vergleicht ein Szenario vor und nach Einführung der entwickelten Lösung.
- *Kapitel 5* diskutiert die Validierung, den Beitrag für Smart Cities und die KI-Verortung.
- *Kapitel 6* fasst die Arbeit zusammen und gibt einen Ausblick.

2 Stand der Forschung

2.1 Objekte im Internet of Things

IoT-Technologien sind Treiber für Veränderungen. Die Entwicklung zum IoT vollzog sich von RFID über drahtlose Sensornetze und wurde stetig von der Etablierung angepasster Kommunikationsprotokolle und Standards für spezielle technische Anforderungen begleitet (Li et al., 2015; Atzori et al., 2010). Die Potentiale der hinter IoT stehenden Technologien werden bereits seit den frühen Entwicklungsstadien für unterschiedliche Anwendungsfälle aufgegriffen (Xu et al., 2014; F.-J. Wu et al., 2011). Hierbei liegen Effizienzsteigerung (weniger Ressourcenverbrauch bei gleicher Ausbringung bzw. höhere Ausbringung bei gleichem Ressourcenverbrauch) oder gesteigerte Effektivität (zielgenauere Erfüllung von Bedürfnissen) als ein klassisches ökonomisches Motiv zugrunde. Darüber hinaus können aber auch neue Geschäftsmodelle und *neuartige Dienste* entstehen.

Für die Generierung neuartiger Dienste bilden die *Objekte* des IoT die zentralen Bausteine. Objekte des IoT sind mit Sensorik, Aktuatorik und Informationsverarbeitung ausgestattete physische Objekte, die ihre Umgebung wahrnehmen und auf sie einwirken können (vgl. Rayes und Salam, 2017, S. 4; Mattern und Flörkemeier, 2010, S. 109 f.). Abbildung 2.1 stellt das Grundmodell für IoT-Objekte dar. Die Sensorik stellt dabei eine Eingabeschnittstelle für Informationsaufnahme aus der Umgebung und die Aktuatorik eine Ausgabeschnittstelle für die Ausführung von Aktionen in der Umgebung dar. Vermittelst der Umgebung besteht hier eine zyklische Wechselwirkung zwischen Ein- und Ausgabe. Unter dem Aspekt, dass die Objekte durch Sensorik und Aktuatorik mit Ihrer Umgebung in einer Wechselwirkung stehen, findet sich diese Grundstruktur in ähnlicher Form in der Multiagentenforschung (vgl. z. B. Kirn, 1996, S. 150). Kirn (2002) beschreibt das Grundmodell mit den drei Aktivitäten für die Informationsaufnahme (s. u. Ausdruck 2.1), Informationsverarbeitung (s. u.

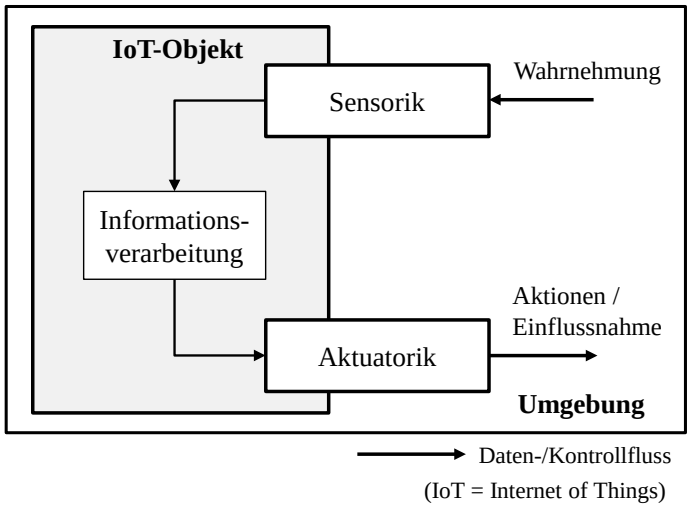


Abbildung 2.1: Grundmodell für IoT-Objekte

Ausdruck 2.2) und Ausführung von Aktionen (s. u. Ausdruck 2.3) formal:

$$sense : SensoryInputs \rightarrow Perceptions \tag{2.1}$$

$$reason : InternalStates \times Perceptions \rightarrow InternalStates \tag{2.2}$$

$$act : Commitments \rightarrow Actions \tag{2.3}$$

SensoryInputs ist dabei die Menge der Eingaben über die Sensorik, welche in die Menge der Wahrnehmungen *Perceptions* abgebildet wird. $InternalSates \subseteq (EnvironmentalModel \times Goals \times Commitments)$ stellt dabei einen internen Informationszustand dar, der durch das Umgebungsmodell *EnvironmentalModel*, den Zielstellungen *Goals* und den geplanten, noch nicht ausgeführten Aktionen in der *Commitments*-Menge gegeben ist. *Actions* ist die Menge der durch die Aktuatorik ausführbaren Aktionen.

Die Funktion *reason*, also die Informationsverarbeitung, soll Kognition für sog. Software-Agenten implementieren, um ihr „Verhalten planen und steuern (Cognition) [...] zu können“ (Kirn, 1996, S. 149 f.). Die Kognition basiert auf Schlussfolgerungsfähigkeiten (vgl. Wooldridge, 2009, S. 65 ff.). Objekte im IoT lassen sich prinzipiell auf die gleiche Weise formalisieren, v. a., wenn diese über kognitive Kapazitäten verfügen sollen. Atzori et al. (2010) beschreiben das IoT nicht nur als Konvergenz einer Internet-bezogenen und Objekt-bezogenen Sicht, sondern auch mit einer Semantik-bezogenen Sicht. Die Semantik-bezogene Sicht stellt Anforderungen an Schlussfolgerungskapazitäten und erfordert somit Kognitionskapazitäten im Sinne des Ausdrucks

2.2.

Auch Xu et al. (2014) beschreiben, dass IoT-Objekte zu smarten, intelligenten Objekten in einem engeren Sinne werden, indem sie mit Schlussfolgerungskapazitäten ausgestattet werden. López et al. (2011, auch im Weiteren) nehmen für smarte Objekte eine Typisierung anhand von fünf Kapazitäten vor, die jeweils mit einem Buchstaben abgekürzt angegeben werden. „I“ steht für Identität und Speicherung von Daten (mindestens der Identitätskennung). „S“ steht für Sensorwahrnehmungen der Umwelt. „A“ steht für Ausführung von Aktionen durch Aktuatorik. „D“ steht für Entscheidungsfähigkeit (engl.: *decision making*). „N“ steht für Netzwerkkonnektivität. IN-Objekte nach dieser Typologie sind dann z. B. Objekte, die mit einer (auf einem eingebetteten Datenspeicher abgelegter) Identität und Netzwerkkonnektivität ausgestattet sind. Dies sind die absoluten Mindestanforderungen an IoT-Objekte. Reine IN-Objekte sind allerdings fraglich, da sie mangels Sensorik oder Aktuatorik von ihrer Umgebung abgekoppelt sind. ISN- oder IAN-Objekte wären hingegen zweckmäßig, wenn der Zweck eines ISN-Objekts darin besteht, aufgenommene Daten über das Netzwerk zu übertragen oder wenn IAN-Objekten die auszuführenden Aktionen über das Netzwerk übermittelt werden. Die eigentliche Intelligenz für smarte Objekte in dieser Typologie kann allerdings in der Entscheidungsfähigkeit gesehen werden, also in der Kapazität „D“. Entscheidungen werden hierbei betrachtet als das Ergebnis eines Schlussfolgerungsprozesses für gegebene Daten bzw. für eine gegebene formale Beschreibung einer Ausgangssituation, um eine Menge von Aktionen zur Einflussnahme auf die Umgebung zu wählen.

Die im IoT realisierte Funktionalität lässt sich gemäß Xu et al. (2014, auch im Weiteren) in vier Schichten strukturieren: (i) Sensor- und Aktuatorschicht, (ii) Netzwerkschicht, (iii) Schnittstellenschicht, (iv) Dienste-Schicht. Die (i) Sensor- und Aktuatorschicht ist direkt mit der existierenden Sensorik- oder Aktuatorik-Hardware integriert, um Daten zu erfassen und auf die Umgebung einzuwirken. Die (ii) Netzwerkschicht stellt die Basisunterstützung für den digitalen Datentransfer dar und kann dabei auch Datenaggregationen oder *Quality-of-Service*-Verwaltung und -Kontrolle enthalten. Die (iii) Schnittstellenschicht bezieht sich auf die Schnittstellen für Interaktionen zwischen Geräten und trägt dazu bei, dass verschiedene Geräte mit unterschiedlichen Interaktionsprotokollen idealerweise nach dem *Plug-&-Play*-Prinzip miteinander integriert werden können. Die (iv) Dienste-Schicht stellt eine *Middleware* dar, welche die Dienste bereitstellt, um Benutzerbedürfnisse zu befriedigen, also schließlich Nutzen zu stiften.

Die Problemstellung der vorliegenden Arbeit lässt sich wie folgt in die soeben beschriebene Konzeptionen für IoT-Objekte einordnen: SSOs sind mit

Sensorik, Aktuatorik und Informationsverarbeitungskapazitäten ausgestattete physische Objekte; die Umgebung ist der urbane Raum; der Safety-Controller soll *reason* gemäß Ausdruck 2.2 durch KI-basierte Informationsverarbeitung implementieren, sodass die SSOs durch planvolles Verhalten kontrollierend auf die Bewältigung einer Strecke einwirken. Der Safety-Controller stellt damit die Entscheidungskapazität „D“ in der Typologie von López et al. (2011) dar, indem er KI-basierte Verfahren für praktisches Schlussfolgern implementiert (vgl. Wooldridge, 2009, S. 65 ff.) und somit ein intelligentes Verhaltensmodell für die SSOs. Die Entscheidungskapazität kann dann gemäß der Strukturierung von Xu et al. (2014) auf der Dienste-Schicht laufen, welche Safety als Nutzen stiftet.

2.2 Der urbane Raum als Anwendung des Internet of Things

Bevor die für die vorliegende Problemstellung und den Lösungsansatz der Arbeit ausschlaggebenden Anforderungen identifiziert, beschrieben und analysiert werden, sei ein Überblick darüber gegeben, wie IoT-Technologien in den urbanen Raum Einzug halten. IoT-Innovationen stellen auch im Kontext von Smart City als entscheidender Treiber einen Ausgangspunkt für Veränderungen dar (Hammi et al., 2018; Zanella et al., 2014). Die Nutzung von IoT-Technologien, um städtebauliche Objekte zu SSOs zu transformieren, die für Passanten direkten Nutzen stiften, ist dabei allerdings vergleichsweise wenig verbreitet. Städtebauliche Objekte werden zwar als Träger für Sensoren verwendet, es wird ihnen aber meistens keine Möglichkeit gegeben, mittels Aktuatorik auf ihre Umgebung einzuwirken. Der urbane Raum als IoT-System unterscheidet sich in diesem Punkt ganz erheblich von IoT-Systemen in bspw. der industriellen Fertigung. Eine Erweiterung städtebaulicher Objekte um Einwirkungsmöglichkeiten in den urbanen Raum stellt einen wesentlichen, weitgehend noch zu leistenden Entwicklungsschritt dar.

Bei prototypisch erforschten SSOs stechen aufgrund ihrer technischen Eignung Lichtsysteme hervor. Zum einen sind Lichtsysteme im urbanen Raum nahezu omnipräsent und bieten sich schon aus diesem Grund als besonders sichtbare städtebauliche Objekte für eine Smartifizierung an. Entscheidender dürfte aber sein, dass Lichtsysteme ohnehin mit Strom versorgt werden müssen, der dann für informationstechnische Erweiterungen oder Sensoren mitgenutzt werden kann. Darüber hinaus bieten die Leuchtmittel von Lichtsystemen eine direkt nutzbare Möglichkeit für visuelle Effekte als Aktuatoren i. w. S. Leucht-

signale als nicht-physische Aktuatoren können visuell kodierte Informationen darstellen – z. B. als adaptive Wegweiser zu freien Sitzgelegenheiten (vgl. Hubl et al., 2018).

Poulsen et al. (2013, auch im Weiteren) implementierten bspw. ein Lichtsystem, das adaptiv auf Bewegungsmuster von Passanten und auf Wetterbedingungen, wie die Windgeschwindigkeit, reagieren kann. Mittels eines Windsensors wird der Windgeschwindigkeitsvektor bestimmt und in eine wellenförmige Visualisierung umgerechnet. Das smarte Beleuchtungssystem visualisiert dann das Ausbreiten der Wellen in Windrichtung über die Straßenlaternen hinweg. Je höher die Windgeschwindigkeit ist, desto schneller breiten sich die Wellen aus und desto näher liegen sie beieinander. In Bezug auf Bewegungsmuster von Passanten implementierten Poulsen et al. (ebd.) eine Art *Heatmap*-Visualisierung mit den Straßenlaternen des smarten Beleuchtungssystem. Je stärker der Bereich um eine Straßenlaterne frequentiert gewesen ist, desto höher ist die Intensität der Straßenlaterne. Die Farbe ist rot. Die Frequentierung wird mithilfe von *Computer-Vision*-Technologie ermittelt. Außerdem wird in einer weiteren Anwendung für vorbeigehende Passanten der umliegende Bürgersteig mit einem weißen Lichtkegel in einem Radius von 5 m beleuchtet. Darüber hinaus können die Farben der Leuchtmittel über ein Smartphone von Passanten manuell an deren individuelle Farbvorlieben angepasst werden.

Farbliche Anpassung von Beleuchtung kann neben dem Bedienen individueller Vorlieben auch eine Wirkung auf die Stimmung haben. Huldtgren et al. (2015, auch im Weiteren) untersuchten diese Möglichkeit explizit, um die Stimmungslage von Senioren zu erhöhen. Die Idee dahinter ist, dass, sobald eine bestimmte negative Stimmungslage bei Senioren erkannt wird, wahl- und fallweise ein aktivierendes oder entspannendes Licht konfiguriert werden kann. Aktivierendes Licht ist dabei hell und weiß, entspannendes Licht eher gedämmt. Das Ziel ist, dass einer negativen Stimmungslage durch angepasste Beleuchtung etwas entgegengewirkt werden kann. Dabei schlussfolgern Huldtgren et al. (ebd.) aus ihren im Rahmen der Untersuchung geführten Gesprächen, dass es Senioren bei der Nutzung eines solchen adaptiven Beleuchtungssystems offenbar wichtig ist, die Kontrolle über das System zu behalten. Diese Untersuchung bezog sich dabei auf eine Nutzung im eigenen Zuhause der Senioren. Die Anwendung lässt sich aber auch auf den urbanen, außerhäuslichen Bereich übertragen. Hierbei kann dann auch eine automatische Erkennung der Stimmungslage mittels Kameras und Sensoren relevanter sein.

Bei automatischer Erkennung und automatischer Ausführung entsteht eine zyklische Kontrollstruktur (vgl. Abb. 1.2 u. 2.1). Cunha und Fuks (2015, auch im Weiteren) machen dazu den Vorschlag, Lampen direkt mit Sensoren, LEDs und Mikrocontrollern auszustatten. Dadurch könnten insbesondere Menschen

mit alterstypischen Beeinträchtigungen adäquat überwacht und ggf. direkt unterstützt oder Hilfe gerufen werden. Lampengehäuse eignen sich dabei gut für die Unterbringung von Sensoren, die darin verborgen werden können. Dies soll Akzeptanzhürden von Senioren entgegenwirken. In diesem Zusammenhang sei darauf hingewiesen, dass es nach Auffassung der vorliegenden Arbeit entscheidender ist, dass ein innovatives System anhand *objektiver* Kriterien nachweislichen Nutzen bringt. Wenn ein System bspw. nachweislich medizinische Notfälle verhindert oder deren Folgen mildert, somit einen hohen Nutzen hat, dann sollte die Frage, ob Sensoren sichtbar oder versteckt sind, nicht die entscheidende Frage sein.

Eine Klasse eher herkömmlicher IT-Objekte, die sich aber im urbanen Raum platzieren und dort als IoT-Objekte auffassen lassen, stellen öffentliche, interaktive Informationstafeln dar. Im Prinzip sind dies große Bildschirme, die im öffentlichen Raum, z. B. an zentralen Stellen in einer Stadt, installiert werden. Benutzer können dann mit ihnen interagieren, um bestimmte Informationen zu erhalten. Cremonesi et al. (2015), Müller et al. (2010) und Vogel und Balakrishnan (2004) untersuchen dazu Möglichkeiten für individuelle Interaktionen, obwohl die Bildschirme in der Öffentlichkeit stehen. Dabei verfolgen sie jeweils den gleichen Grundansatz, dass sich die Individualität der Interaktionen und Informationen abhängig von der Entfernung der Passanten unterscheidet. Individuelle Interaktion wird bspw. ermöglicht, wenn sich ein Passant sehr nahe am Bildschirm befindet und ggf. nur auf einem einzelnen Bildschirmausschnitt. Für weiter entfernte Passanten können allgemeinere Informationen dargestellt werden. Interaktive öffentliche Bildschirme stellen zwar eine naheliegende Möglichkeit dar, den urbanen Raum zu smartifizieren, doch handelt es sich bei Bildschirmen nicht um typische städtebauliche Objekte im Sinne von Stadtmobiliar, sondern eher um mögliche Komponenten für zu SSOs transformierte städtebauliche Objekte.

Als ein System, das als komplementäre Komponente für SSOs verwendet werden könnte, stellen Traunmueller und Schieck (2013, auch im Weiteren) ein *Space Recommender System* vor. Dieses System kann Passanten individuell Routen unter Berücksichtigung von Kriterien empfehlen, welche die Attraktivität einer Route zum Schlendern und Bummeln betreffen. Die Empfehlungen basieren dabei auf Bewertungen von Routen, die andere Benutzer über eine soziale Netzwerkplattform geben können. Die Informationsbeschaffung mittels explizit geforderten Eingaben anderer Benutzer wird auch als *Crowdsourcing* bezeichnet (vgl. auch Abschnitt 2.5.2.2). Die Kriterien könnten auf Barrierefreiheit, Ausruhmöglichkeiten oder Verfügbarkeit von SSOs ausgerichtet werden.

Es bestehen weitere Möglichkeiten, mithilfe von IoT-Technologien Passanten mit alterstypischen Beeinträchtigungen im urbanen Raum Assistenzfunktionen

zu bieten. Foell et al. (2014, auch im Weiteren) entwarfen ein sog. Mikro-Navigationssystem, das Passanten bei der Nutzung von ÖPNV assistieren soll. Zielgruppe sind hierbei insbesondere Passanten, die Schwierigkeiten mit der ÖPNV-Nutzung haben. Dies können bspw. unerfahrene ÖPNV-Nutzer, Touristen, allgemein Passanten mit Schwierigkeiten sich im betreffenden Gebiet zu orientieren oder eben Passanten mit alterstypischen Beeinträchtigungen sein. Das System ist internetbasiet und informiert die Benutzer über ein tragbares Endgerät, z. B. über ein Smartphone, bspw. darüber, ob sie sich gerade im richtigen Bus befinden oder wie lange es noch dauert, bis sie aussteigen müssen.

Kumar et al. (2017, auch im Weiteren) entwickelten ein Assistenzsystem für Passanten mit beeinträchtigtem Sehvermögen. Dieses System kann beim Finden eines geeigneten Weges unterstützen und mittels Ultraschallsensoren Hindernisse auf dem Weg erkennen. Die Ultraschallsensoren sind dabei per Bluetooth mit einem Smartphone verbunden. Unter Nutzung der Smartphone-Kamera und einer auf künstlichen neuronalen Netzen basierenden Software kann das System außerdem bekannte Gesichter erkennen. Dies soll Passanten mit beeinträchtigtem Sehvermögen dabei unterstützen, Personen zu identifizieren, denen sie begegnen. Es ist allerdings zu beachten, dass nicht alle hier vorgestellten Assistenzsysteme für den urbanen Raum SSOs sind. Im letztgenannten System werden bspw. sogar überhaupt nicht *Objekte* smartifiziert, d. h. mit Sensoren, Aktuatoren oder Informationsverarbeitungskapazitäten ausgestattet. Für eine Einordnung beschreiben Skowron et al. (2019) eine Taxonomie, welche den Konstruktionslösungsraum nach den Dimensionen „SSO“, „Hilfestellung“ und „Personen“ aufschlüsseln.

Es sei an dieser Stelle darauf hingewiesen, dass öffentliche Sitzgelegenheiten als SSOs später (in Kap. 2.5) beschrieben werden. Denn in der nun folgenden Problemanalyse stellen sich zu SSOs transformierte Sitzgelegenheiten als einen Lösungsansatz für die Herstellung von Adaptivität des urbanen Raumes zur Erhöhung seiner sicheren Benutzbarkeit für Passanten mit alterstypischen Beeinträchtigungen heraus. Die dedizierte Analyse des Stands der Forschung zu Sitzgelegenheiten als SSOs erfolgt daher, nachdem die Probleme von Passanten bei der Benutzung des urbanen Raums analysiert sind.

Außerdem sei darauf hingewiesen, dass für SSOs als Objekte des öffentlichen Raumes besondere Anforderungen bestehen, die im Kapitel 2.6 genauer analysierte werden und für IoT-Innovationen für den urbanen Raum bspw. im Gegensatz zur Umsetzung in der Fertigungsindustrie zu beachten sind: Aufgrund der intendierten öffentlichen Zugänglich- und Benutzbarkeit städtebaulicher Objekte muss für unterschiedliche, ggf. gegenseitig konfliktäre individuelle Benutzerbedürfnisse eine legitime kollektive Lösung gefunden werden. Dies ist

insbesondere dann relevant, wenn ein SSO prinzipiell von mehreren Passanten gleichzeitig genutzt werden kann und die Objekte eine gewisse Adaptivität realisieren sollen. Die Frage ist dann: An wen bzw. woran sollen sich die SSOs adaptieren?

2.3 Safety-Konzeption

Safety wird als Terminus in Abgrenzung zu *Security* verwendet (Albrechtsen, 2003; Menz et al., 2015; Hillenbrand, 2012, S. 42). Beides wird ins Deutsche mit „Sicherheit“ übersetzt, wobei möglicherweise eine intendierte explizite Differenzierung verloren geht. Gemäß den Oxford Dictionaries ist Safety „[t]he condition of being protected from or unlikely to cause danger, risk, or injury“ (Oxford Dictionaries online, 2020a) und Security „[t]he state of being free from danger or threat“ (Oxford Dictionaries online, 2020b). In der Konnotation der Umgangssprache wird Security häufig auf kriminelle Vorfälle und Safety häufig auf Verletzungsrisiken für Menschen bezogen (Albrechtsen, 2003, S. 3).

Darüber hinaus weist Albrechtsen (ebd.) auf eine definitorische Differenzierung hin, nach der mit dem Begriff „Safety“ Schutz vor zufälligen, ungewollten Vorfällen gemeint ist, während mit dem Begriff „Security“ Schutz vor absichtlich herbeigeführten Vorfällen gemeint ist. Eine (andere) begriffliche Abgrenzung vornehmend, definieren Menz et al. (2015) Safety als Schutz der Umgebung eines Systems vor dem Fehlverhalten des Systems und Security als Schutz eines Systems vor Angriffen von außen. Hillenbrand (2012, S. 42) nimmt die gleiche konzeptionelle Unterscheidung vor und bezeichnet Safety als Betriebssicherheit und Security als Angriffssicherheit.

Es wird offenkundig, dass dem Begriff „Safety“, gerade bei Verwendung als Terminus, eine andere Bedeutung zukommen kann als dem Begriff „Security“. Die deutsche Übersetzung „Sicherheit“ ist somit möglicherweise ein Homonym für unterschiedliche Sachverhalte (ebd.). Aber auch bei ausschließlicher Verwendung des Begriffs „Safety“, können unterschiedliche Sachverhalte gemeint sein. Denn zufällige Vorfälle können sich bspw. sowohl auf Risiken für das System als auch auf Risiken für die Systemumwelt beziehen und umgekehrt können Risiken für die Systemumwelt sowohl zufällig als auch absichtlich, bzw. aufgrund der Systemspezifikation, entstehen.[6]

[6] Ein Exkursbeispiel dafür liefert die sog. *Soft Close Automatic*, die eine Autotür auch dann zuzieht, wenn diese mit einem Finger einen Spalt weit offengehalten wird. Dies erfüllt die (ursprüngliche) Spezifikation, ist aber nicht sicher. (Das Verbrauchermagazin „Plusminus“ der Arbeitsgemeinschaft der öffentlich-rechtlichen Rundfunkanstalten der Bundesrepublik Deutschland (ARD) berichtete Anfang 2018 über einen Fall, bei dem

Die genannten Safety-Auffassungen (von Menz et al., 2015 und Hillenbrand, 2012), die sich auch in der VDI-Richtlinie zur Technikbewertung unter dem Begriff der Sicherheit widerspiegeln (VDI, 2000), legen eine konzeptionelle Trennung zwischen dem intendierten nutzenstiftenden Zweck eines Systems und Sicherheitsaspekten in der Art nahe, dass ein System bei der Erfüllung seines intendierten Zwecks Risiken *hervorbringt*. Eine solche konzeptionelle Trennung ist für die vorliegende Arbeit allerdings nicht zielführend. Denn hier ist der intendierte Zweck gerade die Erhöhung von Safety. Die folgende Entwicklung des dieser Arbeite zugrunde liegenden Safety-Begriffs zeigt, wie Safety so konzipiert werden kann, dass Safety als intendierter Zweck in die Systementwicklung eingeht.

Die Safety-Konzeption geht auf einen *Safety-Engineering*-Ansatz nach Leveson (2011) zurück. Demnach ist das oberste Safety-Ziel: „Freedom from accidents (loss events)“ (ebd., S. 467); wobei ein Unfall aufgefasst wird als: „An undesired or unplanned event that results in a loss, including loss of human life or human injury, property damage, environmental pollution, mission loss, etc.“ (ebd., S. 181). Ausgangspunkt für Maßnahmen gemäß der Safety-Konzeption ist die Analyse von Gefahren als: „A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss).“ (ebd., S. 184). Hieraus ergibt sich die Einsicht, dass Unfälle prinzipiell auf zwei Arten verhindert werden können. Erstens, indem verhindert wird, dass ein System in einen gefährlichen Zustand kommt; zweitens, indem bestimmte schlimmstmögliche Umweltbedingungen vermieden werden.

Gefährliche Systemzustände und schlimmtsmögliche Umweltbedingungen bilden einen Dualismus: Ob ein Systemzustand als gefährlich gilt, hängt von den entsprechenden als schlimmstmöglich definierten Umweltbedingungen ab und v. v. Gefährlicher Systemzustand und schlimmstmögliche Umweltbedingungen können zwar nicht unabhängig voneinander definiert werden, aber sie können unabhängig voneinander eintreten. Dies kann konstruktiv zur Verhinderung von Unfällen ausgenutzt werden. Denn in einigen Fällen lässt sich der Eintritt gefährlicher Systemzustände nicht vermeiden. Dann kann aber immer noch versucht werden, zu verhindern, dass der gefährliche Systemzustand auf die entsprechenden schlimmstmöglichen Umweltbedingungen trifft.

Mithilfe des Safety-Engineering-Ansatzes kann eine Erklärungstheorie für Unfälle unterstellt werden (ebd., S. 76 ff.): Ausgangspunkt der Theorie ist, dass Unfälle deshalb geschehen, weil sogenannte Safety-*Constraints* nicht erfolgreich aufrechterhalten werden. Safety-Contraints sind bestimmte Bedingungen, die durch *passive* oder *aktive* Controller erzwungen werden sollen. Passive Con-

dadurch ein Finger abgetrennt wurde.)

troller sollen Safety-Constraints dadurch aufrechterhalten, dass sie physisch vorhanden sind (Leveson, 2011, S. 76 f.). Wenn darüber hinaus aber auch eine *Steuerung* hinsichtlich bestimmter Kriterien erfolgen soll, dann sind zusätzlich *aktive* Kontrollelemente notwendig.

Aktive Controller sollen Safety-Constraints aufrechterhalten durch (ebd., S. 77):

- Überwachen der relevanten Umweltzustände,
- Erheben der Werte notwendiger Systemvariablen,
- Interpretieren der erfassten Werte,
- Auswählen geeigneter Aktionen zur Vermeidung von Unfällen und
- Ausführen der ausgewählten Aktionen.

Aktive Controller können prinzipiell auch manuell ausgeführt werden, sind aber typischerweise stark durch IT geprägt. Vor allem für die Automatisierung der Interpretation der erfassten Werte und der Auswahl geeigneter Aktionen kann eine über einfaches „Stimulus-Response“ hinausgehende Informationsverarbeitung im Sinne der Automatisierung einer komplexen, d. h. intellektuell nicht handhabbaren,[7] Entscheidung sachgerecht sein. Die vorliegende Arbeit fokussiert die Auswahl geeigneter Aktionen als ein für sich stehendes Problem, welches mit Methoden der KI gelöst werden soll. Diese KI stellt dann den (aktiven) Controller dar.

Gemäß der Theorie nach Leveson (ebd., S. 92) ist mindestens eine der folgenden Aussagen wahr, wenn es zu einem Unfall kommt, d. h. es muss versucht werden das Eintreten aller dieser folgende Sachverhalte zu verhindern:

- Eine erforderliche Kontrollaktion stand nicht zur Verfügung.
- Eine erforderliche Kontrollaktion stand zwar zur Verfügung, aber zur falschen Zeit (zu früh, zu spät, zu kurz).
- Die zur Verfügung gestellte Kontrollaktion hat selbst ein Safety-Constraint verletzt.
- Eine adäquate Kontrollaktion stand zwar zur Verfügung, wurde aber nicht ausgeführt.

Für die vorliegende Arbeit tritt an die Stelle des Unfall-Begriffs ein sich daran anlehnender Begriff des *Safety-Vorfalls* gemäß Definition 1 (s. u.). Safety-Vorfälle gelten per definitionem als eingetreten oder nicht. Sie können als notwendige Vorbedingungen für Unfälle betrachtet werden, unabhängig davon,

[7] „In fact, complexity can be defined as intellectual unmanageability.“ (Leveson, 2011, S. 4)

ob es tatsächlich zu sog. Verlustereignissen kommt (Tod, Verletzung, Schaden usw.; vgl. ebd., S. 181).

Als Systeme, welche in einen gefährlichen Zustand geraten können, seien Passanten betrachtet. Zwar beziehen sich in der Safety-Konzeption von Leveson (ebd.) Systeme i. d. R. auf technische Systeme. Dennoch ist die Betrachtung von Passanten als Systeme mit der Safety-Konzeption vereinbar: Die Safety-Konzeption basiert auf einem systemtheoretischen Fundament (ebd., S. 61 ff.). Mit dem systemtheoretischen Systembegriff ist es konzeptionell möglich, auch Menschen als Systeme oder Subsysteme zu betrachten (vgl. Stachowiak, 1973, S. 73; Ropohl, 2009, S. 107 ff.). Statt von Gefahren als Systemzustände wird nachfolgend von kritischen Personenzuständen gesprochen. Statt von schlimmstmöglichen Umweltzuständen soll fortan von kritischen Umweltzuständen gesprochen werden. Umweltzustände beziehen sich im Rahmen dieser Arbeit auf die urbane Umgebung mit städtebaulichen Objekten und den aus Sicht eines einzelnen Passanten anderen Passanten.[8] Letzten Endes bewährt sich die Begriffsinterpretation dadurch, dass sie eine konstruktive Nutzung der Safety-Konzeption für die vorliegende Arbeit ermöglicht. Ziel ist die Vermeidung von Safety-Vorfällen, wobei gelte:

Definition 1 (Safety-Vorfall) *Ein Safety-Vorfall liegt genau dann vor, wenn ein kritischer Personenzustand und ein kritischer Umweltzustand zusammenkommen.*

2.4 Adaptivitätserfordernis auf Individuenebene: Motorische Beeinträchtigungen im Altersgang

2.4.1 Kraft

Mit zunehmendem Alter wird die Safety des urbanen Raumes i. S. v. sicherer Benutzbarkeit für Fußgänger in körperlicher Hinsicht typischerweise durch abnehmende motorische Leistungsfähigkeit beeinträchtigt. Die OECD (2001, S. 52) fasst dies wie folgt zusammen:

> The mobility of older pedestrians may be progressively impaired by general physical changes experienced in the ageing process such as loss of agility, flexibility and endurance, reduced cardio-respiratory capacity, postural mechanisms and deteriorating balance. These factors lead to slower walking, as a result of taking shorter steps and precautions to maintain balance. [...]

[8] Vergleiche Ferber (2001, S. 32): „Umgekehrt ist die Umwelt eines Agenten all das, was der Agent nicht selbst ist."

> The percentage of older people with varying levels of sensory, cognitive or motor disabilities, associated or not with a specific pathology, increases with age. Furthermore, a large number of older people are taking medication that frequently affects their ability to walk safely. [...] (OECD, 2001, S. 52)

Der Rückgang der motorischen Leistungsfähigkeit mit fortschreitendem Alter äußert sich unter anderem in einer langsameren Gehgeschwindigkeit, sowohl für die normale als auch für die maximale Gehgeschwindigkeit (Bohannon, 1997). Die normale Gehgeschwindigkeit ist die vom gehenden Subjekt als komfortabel empfundene Gehgeschwindigkeit. Die Reduktion der normalen Gehgeschwindigkeit könnte zwar auch auf eine willkürliche Entscheidung der Subjekte zurückzuführen sein, im Alter etwas langsamer zu gehen. Die Reduktion der maximalen Gehgeschwindigkeit ist allerdings ein deutlicher Indikator dafür, dass für die Reduktion der Gehgeschwindigkeit ein Rückgang der motorischen Leistungs*fähigkeit* (und nicht des Leistungswillens) *ursächlich* ist.

Dies wird durch eine parallel zur Messung der Gehgeschwindigkeiten durchgeführte Messung der Muskelstärken an den unteren Gliedmaßen bekräftigt. Bohannon (ebd.) konnte eine statistisch signifikante Korrelation zwischen der maximalen isometrischen Kraftausübung[9] an den für die Bewegung der unteren Gliedmaßen beteiligten Muskeln und der maximalen wie auch der normalen Gehgeschwindigkeit zeigen.

Die Korrelationen alleine liefern keine Erklärung, die erklärt *wie* es zur geringeren motorischen Leistungsfähigkeit kommt. Das Alter als „erklärende" Variable kann hierzu bspw. wenig bis gar nichts beitragen. Auch wenn mit Kausalerklärungen Antworten auf „Warum?"-Fragen gegeben werden können, so ist es vielmehr die Beantwortung von „Wie?"-Fragen, durch welche eine Erklärung im engeren kausalen Sinne informativ wird. Die Korrelation der Gehgeschwindigkeit mit den Kraftausübungen in der Muskulatur lenkt die Erklärungsansätze aber in Richtung bestimmter biomechanischer Ursachen.

Winter et al. (1990) untersuchten Gangveränderungen im Alter durch Messung biomechanischer Parameter. Sie beobachteten unter ausschließlich gesunden Personen, dass die Kraft, mit der sich ältere Personen beim Gehen abstoßen, geringer ist als bei jüngeren Personen und dass damit eine kürze Schrittweite sowie eine längere Zweibeinstandzeit einhergeht. Winter et al. (ebd.) sehen aufgrund ihrer Messungen drei *gleichwertige* kausale Erklärungsmöglichkeiten dafür: (1) Ältere Menschen könnten versuchen durch Verlängerung der Zweibeinstandzeit und der Verkleinerung des Winkels, mit dem die Ferse auf den Boden trifft, ihre Stabilität zu erhöhen. Kleinerer Auftrittswinkel der

[9] Bei *isometrischer* Kraftausübung eines Muskels ändert sich im Gegensatz zu *dynamischer* Kraftausübung seine Länge nicht (sondern lediglich seine Spannung). Die von Bohannon (1997) verwendete Messmethode ist bei Andrews et al. (1996) beschrieben.

Ferse wird durch kürze Schrittlänge erreicht, die wiederum durch schwächeres abstoßen erreicht wird. (2) Ältere Menschen könnten sich auch einfach stabiler fühlen, wenn sie kleinere Schritte machen oder langsamer gehen. Kleinerer Auftrittswinkel der Ferse und längere Zweibeinstandzeit wären dann eine Folge, aber nicht der Grund für das Gangverhalten. (3) Kleinere Schrittlänge und längere Zweibeinstandzeit könnte auch durch schwächere Muskelkraft bedingt sein.

Für die dritte Erklärungsmöglichkeit sprechen neben den oben beschriebenen Beobachtungen von Bohannon (1997) auch die Ergebnisse von Pisciottano et al. (2014). Denn auch Pisciottano et al. (ebd.) fanden, dass der stärkste Faktor für die Leistung in Gehfähigkeitstests[10] die maximale Muskelstärke der unteren Gliedmaßen ist. Da diese Studie unter gesunden Frauen im Alter von 65 Jahren oder mehr durchgeführt wurde, können auch hier, wie bei Winter et al. (1990), pathologische Ursachen weitestgehend ausgeschlossen werden.

Wenn nun, wie in den Studien beobachtet, die Gehfähigkeit im Alter abnimmt, pathologische Ursachen aber ausgeschlossen werden können, dann gibt es offenbar auch nicht-pathologische Ursachen für die Reduzierung der Gehfähigkeit im Alter. Dies unterstreicht, dass es typische Beeinträchtigungen der motorischen Leistungsfähigkeit gibt, die nicht an der Ausprägung bestimmter Krankheiten hängen, sondern tatsächlich am natürlichen Alterungsprozess.

Eine der bedeutendsten im natürlichen Verlauf des Alterns auftretende physiologischen Veränderungen ist der Verlust von Skelettmuskelmasse, bezeichnet als Sarkopenie (Evans und W. W. Campbell, 1993; Rosenberg, 1997). Die damit verbundenen Beeinträchtigungen äußern sich unter anderem im oben beschriebenen Phänomen geringerer Muskelkraft (Evans und W. W. Campbell, 1993).

Borkan et al. (1983) beobachteten, dass der Fettanteil zwischen und in der Muskulatur bei den Älteren signifikant größer ist als bei den Jüngeren. Während in jüngeren Jahren Fett vor allem subkutan, d. h. im Gewebe direkt unter der Haut, vorkommt, verlagern sich die Fettvorkommen in höherem Alter hinein in die Muskulatur bzw. deren Zwischenräume (ebd.).

Die Anzahl und zum Teil auch die Größe der Muskelfasern nimmt im Alter ab (Lexell et al., 1988). Da die Muskelfasern im Prinzip die Funktionalität der Muskulatur bestimmen, bedeutet deren Abnahme unmittelbar eine Beeinträchtigung der Muskulatur, also v. a. der Fähigkeit, Kraft auszuüben (vgl. Rinkenauer, 2008, S. 146 f.). Für Lexell et al. (1988) kann die Abnahme der Muskelfasern entweder auf irreparable Schäden an den Fasern selbst zurückzu-

[10] Messung bei Pisciottano et al. (2014) durch: *Timed Up and Go Test*, *Berg Balance Test* und *Dynamic Gait Index*.

führen sein oder aber darauf, dass die Fasern nicht mehr innerviert, d. h. nicht mehr an das Nervensystem angeschlossen sind und dadurch degenerieren. Mit steigendem Alter durchlaufen Muskelfasern den Schlussfolgerungen von Lexell et al. (1988) zufolge einen Prozess mit Denervierungen und Reinnervationenen, d. h. im Altersgang werden Muskelfasern, die vom Nervensystem abgetrennt sind, wieder daran angeschlossen. Wenn jedoch die Reinnervationsfähigkeit in höherem Alter so gering wird, dass abgetrennte Muskelfasern nicht mehr innerviert werden können, dann gehen diese Fasern schließlich verloren und an deren Stelle wird Fett und fibröses Gewebe in das betroffene Muskelareal eingelagert (ebd.). Dies ist nicht nur eine Erklärung zu der oben beschriebenen Feststellung von Borkan et al. (1983), dass die Muskulatur von älteren Personen signifikant weniger Muskelgewebe enthält als die von jüngeren, sondern offenbar auch die Hauptursache des altersbedingten Muskelschwundes (Lexell et al., 1988).

Damit konsistent zeigten Doherty et al. (1993), dass die auch bei gesunden, aktiven Senioren reduzierte Kraftausübungsfähigkeit auf den Verlust von die Muskulatur innervierenden Motoneuronen zurückgeführt werden kann und schlossen daraus, dass die reduzierte Kraftausübungsfähigkeit mittelbare Folge des Verlustes an Muskelmasse ist (ebd.).

Lynch et al. (1999) untersuchten altersbezogene Effekte auf die Muskelqualität. Die Muskelqualität bestimmten sie dabei durch das maximal erzeugbare Drehmoment sowohl bei konzentrischer als auch bei exzentrischer Muskelkontraktion[11] und setzten diese ins Verhältnis zur Muskelmasse (ebd.). Das maximale Drehmoment nahm dabei in den Beinen stärker ab als in den Armen (ebd.). Die unterschiedlich starke Abnahme in Armen und Beinen bezüglich der Muskelqualität war allerdings nur bei Frauen zu beobachten (ebd.). Daraus lässt sich schlussfolgern, dass die Fähigkeit ein hohes Drehmoment zu erzeugen, d. h. eine hohe Kraft auszuüben, nicht nur an der Masse des Muskelgewebes alleine hängen kann. Tatsächlich scheinen die Muskelfasern im Alter auch einfach schwächer zu werden (Frontera et al., 2000).

2.4.2 Beweglichkeit

Allerdings nimmt im Alter nicht nur die Muskelkraft ab, sondern auch die Beweglichkeit. Das oben Beschriebene bestätigend maßen Vandervoort et al. (1992) für die Dorsalflexion im Sprungelenk, d. h. für das Beugen (*Flexion*) des Fußes in Richtung Fußrücken (*dorsal*), ebenfalls geringere willkürliche

[11] Bei konzentrischer Muskelkontraktion verkürzt sich der Muskel, bei exzentrischer Muskelkontraktion wirkt der Muskel seiner unkontrollierten Verlängerung entgegen. Das Drehmoment entsteht an einem Gelenk.

Kraftausübungsfähigkeit mit höherem Alter. Darüber hinaus maßen Vandervoort et al. (ebd.) gleichzeitig ein höheres Gegenmoment für die Drehbewegung des Fußes sowie einen geringen Umfang der möglichen Fußbeugung mit höherem Alter. Hier kommen also drei physiologische Beeinträchtigungen zusammen: (1.) Ältere Menschen können weniger Kraft ausüben, (2.) der mechanische Widerstand bei der Ausführung von Bewegungen ist größer und (3.) der Bewegungsumfang reduziert sich. Diese drei Effekte wurden durch die Anlage der Untersuchung so erhoben, dass sie als voneinander unabhängig betrachtet werden können. Denn Vandervoort et al. (ebd.) verwendeten bspw. nicht das Drehmoment als Maß für die Kraftausübung, welches dann direkt auch vom Gegenmoment abhinge, sondern die isometrische Muskelkontraktion.

Weitere biomechanische Studien belegen solche Beeinträchtigungen auch für andere Muskel- bzw. Gelenkregionen. Nonaka et al. (2002) zeigten bspw. durch anthropometrische Vermessungen verschiedener Körperhaltungen, dass sich der Umfang passiver Beweglichkeit der Knie- und Hüftgelenke im Alter typischerweise reduziert. Für die passive Beweglichkeit muss das Subjekt selbst nicht aktiv Kraft aufwenden, um die Bewegung auszuführen. Dadurch wird tatsächlich die Beweglichkeit erhoben und nicht bloß die Fähigkeit, die Kraft für eine Bewegung aufzubringen. Auch Ronsky et al. (1995), die dreidimensionale kinematische Messungen der Gelenkbewegungen bei älteren Menschen durchführten, kommen im Vergleich zu vorangegangen Studien mit denselben gemessenen Variablen bei jüngeren Menschen zu der Schlussfolgerung, dass die Gelenkbeweglichkeit, vor allem an den Knien, mit dem Alter typischerweise abnimmt. Ursächlich dafür können Veränderungen der Gelenkflüssigkeit (Wachtel et al., 1995) oder der Gelenkknorpelschichten sein (Rinkenauer, 2008, S. 146). Dass die beschriebenen physiologischen Veränderungen tatsächlich *nicht-pathologische* Ursachen für die Beeinträchtigung der Mobilität zu Fuß älterer Menschen sind, wird bspw. durch die Studien von Kerrigan et al. (1998) bestätigt.

Auch die Reduktion der Beweglichkeit ist indirekt auf den Verlust der Muskelmasse und der in dieser Folge eintretenden Infiltration mit nicht-muskulärem Gewebe zurückzuführen. Denn eingehendere Untersuchungen zeigen, dass die Einlagerungen nicht-muskulären Gewebes in die Muskulatur höhere Muskelsteifheit impliziert (Gosselin, Adams et al., 1998; Gosselin, Martinez et al., 1994). Diese Infiltrationen scheint dabei ab einem gewissen Alter[12] beschleunigt zu geschehen (Rice et al., 1989) und betrifft vor allem die Muskelregionen der unteren Extremitäten (ebd.). Dieser Befund spielt für die Mobilität zu Fuß eine herausragende Rolle, da er bedeutet, dass die Muskelregionen, die beim

[12]Rice et al. (1989) nennen hierzu das Alter ab 70-75 Jahren.

Gehen wohl am meisten beansprucht werden, durch den natürlichen Alterungsprozess verhältnismäßig stark beeinträchtigt werden.

Dass die genannten Beeinträchtigungen nicht mit spezifischen Erkrankungen zusammenhängen, wurde bereits einige Male erwähnt. Außerdem ist die ebenfalls mehrfach erwähnte Infiltration der Muskulatur mit nicht-muskulärem Gewebe im Alter keine Eigenart der menschlichen Physiologie, denn sie tritt bspw. auch bei Ratten auf und wurde dort eingehender untersucht (Gosselin, Adams et al., 1998; Gosselin, Martinez et al., 1994; Thomas et al., 1992; Zimmerman et al., 1993). Auslöser dafür ist ein allgemeiner Muskelschwund im Altersgang, dem zwar durch Training entgegengewirkt werden kann (Evans und W. W. Campbell, 1993), aber auch ohne pathologische Befunde nicht aufzuhalten zu sein scheint. Es scheint also ein naturgegebenes, gleichsam zwingendes Prinzip des Alterns zu existieren.

Eine mögliche biologische Erklärung liefert Hayflick (1965), indem er zeigte, dass die Anzahl der Zellteilungen normaler menschlicher Zellen nach oben begrenzt ist. Dies erklärt bspw. die abnehmende Reinnervationsfähigkeit von Muskelfasern: Die dafür nötigen Nervenzellen können nicht mehr durch Zellteilung produziert werden. Als möglichen Grund dieser Zellteilungsgrenze wiederum beschreibt Olovnikov (1973) die Verkürzung an den sog. Telomeren, also an den Chromosom-Enden, bei jeder Zellteilung. Gut zwanzig Jahre nach der Aufdeckung der natürlichen Zellteilungsgrenze stellte Hayflick (1985) selbst eine Reihe weiterer biologischer Theorien des Alterns zusammen, welche bspw. die Rolle freier Radikale, von Zellabfallprodukten, exogen induzierter Zellmutationen oder fehlerhafter Zellreplikation betrachten. Eine neuere Übersicht findet sich bei Weinert und Timiras (2003).

Altern ist zwar auf der einen Seite individuell, sodass sowohl sehr alte Menschen anzutreffen sind, denen kaum Beeinträchtigungen anzumerken sind, als auch relativ junge Senioren, die kaum noch in der Lage sind, Aktivitäten ohne Hilfsmittel oder ohne Unterstützung auszuführen. Auf der anderen Seite ist der beschriebene Mechanismus, welcher zur Degenerierung der motorischen Leistungsfähigkeit führt, nach gegenwärtigem Stand des Wissens unumkehrbar. Bei Erreichen eines gewissen Alters wirkt sich dies dann grundsätzlich negativ auf die Mobilitätskapazitäten aus, denn: “All body movements are produced by contractions of skeletal muscles. Consequently, any impairment in the functional properties of skeletal muscle results in some degree of immobility.“ (Brooks und Faulkner, 1994, S. 432).

2.4.3 Sensomotorik

Zu den funktionalen Eigenschaften der Skelettmuskulatur gehören neben Kraftausübungsfähigkeit und Beweglichkeit auch sog. sensomotorische Kapazitäten. Sensomotorische Kapazitäten beziehen sich auf die Wechselwirkung des Bewegungsapparates mit ständigen sensorischen Wahrnehmungen und sind sowohl für zielgerichtete willkürliche Bewegungen, für reflexhafte Ausgleichsbewegungen, aber auch für das Halten von Körperteilen in bestimmten Positionen essentiell (Rinkenauer, 2008, S. 145 f.).

Bspw. muss der Bewegungsapparat beim zufälligen und plötzlichen Verschieben der Standfläche reflexartig die Positionen und Ausrichtungen der Körperteile anpassen, um ein neues Gleichgewicht zu finden, damit der Körper nicht umfällt. Dazu testeten Peterka und Black (1990) die muskuläre Reaktion bei Probanden, die auf einer nach vorne und hinten beweglichen Fläche standen. Bei 214 Probanden, dessen Lebensalter zwischen 7 und 81 Jahren etwa gleichverteilt gewesen ist, erhielten sie über das Alter hinweg einen leichten, aber statistisch signifikanten Anstieg der Latenz in der elektrischen Aktivität in den Beinmuskeln (ebd.). Das heißt, die muskuläre Reaktion auf die Verschiebung der Standfläche erfolgt in höherem Alter offenbar mit größerer Verzögerung als im jüngeren Alter. Auch wenn Peterka und Black (ebd.) auf ein relativ geringes Ausmaß des altersbezogenen Unterschiedes in der Reaktionsgeschwindigkeit hinweisen, zeigen die im folgenden referierten Untersuchungen, dass sich im Alter motosensorische Veränderungen durchaus deutlich bemerkbar machen.

Bei Untersuchungen der Präzision, mit der einfache Armbewegungen ausgeführt werden, stellten Darling et al. (1989) eine mit dem Alter zunehmende Variabilität der Bewegungstrajektorien fest. Probanden sollten durch Bewegung ihres Unterarms eine Projektion ihrer Handposition mit einem etwa 50 cm entfernten visualisierten Balken, dessen Breite bezogen auf die Ellbogenposition etwa $4° - 5°$ betrug, horizontal in Übereinstimmung bringen (ebd.). Zwar können ältere Menschen die Positionierung mit der gleichen Präzision vornehmen, wie jüngere, allerdings müssen die Älteren die Bewegungen dann langsamer ausführen (ebd.). Im Umkehrschluss bedeutet dies, dass die Bewegungen älterer Personen offenbar weniger präzise sind, wenn sie diese mit der gleichen Geschwindigkeit ausführen (müssen) wie jüngere.

An dieser Stelle soll eine kurze Reflexion zur Erkenntnismethodik eingebracht werden. Denn Darling et al. (ebd.) führten ihre Studie mit einer Stichprobengesamtgröße von 15 durch. Darin befanden sich 6 junge Probanden im Alter zwischen 21 und 24 Jahren und 9 älteren Probanden im Alter zwischen 68 und 95 Jahren. Die Frage ist, ob die 6 jungen Probanden stellvertretend für die jungen Menschen der Grundgesamtheit gelten können und entsprechend, ob

die 9 älteren Probanden stellvertretend für die älteren Menschen der Grundgesamtheit gelten können. Solch eine Frage ergibt sich in erster Linie dann, wenn eine gesuchte Kausalerklärung von sog. zweiter Art ist und daher einer induktiven Logik unterliegt.[13]

> Der Satz „X, weil Y“ ist eine korrekte kausale Erklärung der zweiten Art genau dann, wenn Y vor X aufgetreten ist und wenn das Auftreten von Y das Auftreten von X wahrscheinlicher gemacht hat bzw. X unwahrscheinlicher gewesen wäre, wenn Y nicht stattgefunden hätte. [Fußnote entfernt] (Beckermann, 1977, S. 43)

Beckermann (ebd., S. 44) bezeichnet die Kausalerklärung der zweiten Art auch als schwache kausale Erklärung. X ist in der allgemeinen Formulierung Platzhalter für ein Explanandum, d. h. für eine zu erklärende Variable. Y ist Platzhalter für ein Explanans, d. h. für eine erklärende Variable. Durch die Aufteilung von Darling et al. (1989) in die zwei oben beschriebenen Experimentalgruppen „junge Probanden“ (welche als Kontrollgruppe definiert ist, siehe ebd., S. 151) und „ältere Probanden“, wird die Tatsache, ob eine Person alt ist oder nicht als Explanans nahegelegt. Dies soll dann X = Bewegungspräzision erklären.

Die anvisierte kausale Erklärung der zweiten Art lautet dann: „Wenn eine Person alt ist, dann ist es wahrscheinlicher, dass die Präzision ihrer Bewegung geringer ist (als bei jungen Personen).“ Bei empirischen Experimenten werden Schlussfolgerungen mittels induktiver Wahrscheinlichkeitsrechnung gezogen, wobei die Wahrscheinlichkeiten in der Regel durch relative Häufigkeiten geschätzt werden. Da relative Häufigkeiten bei großen absoluten Häufigkeiten gegen die in der Wahrscheinlichkeitsrechnung zugrunde gelegten theoretischen Wahrscheinlichkeiten und Wahrscheinlichkeitsverteilungen konvergieren, sollten die Stichproben aus rein mathematischen Gründen nicht zu klein sein: Relative Häufigkeiten können eine Interpretation für das Axiomensystem der Wahrscheinlichkeitstheorie, d. h. im axiomatischen Sinne ein Modell der Wahrscheinlichkeitstheorie, sein. Aus dem Axiomensystem leiten sich die in der Wahrscheinlichkeitsrechnung angewendeten Lehrsätze (Theoreme) ab. Nach dem Gesetz der großen Zahlen ist die Interpretation durch relative Häufigkeiten im mathematischen Sinne für gegen unendlich gehende Stichproben gültig.

Neben den mathematischen Argumenten liefern van Eersel et al. (2019) einen aktuellen wissenschaftstheoretischen Beitrag zu Problemen induktiv-statistischer Schlussfolgerungen aus empirischen Experimenten. Sie beziehen sich dabei vor allem auf humanwissenschaftliche Experimente, denen durch ihre Anlage oft zu eigen ist, dass kein hinreichendes mechanistisches Wissen in den

[13] Zur induktiven Logik siehe auch: Carnap (1959).

Schlussfolgerungsprozess eingebracht werden kann (ebd., S. 224). Empirisch valide Experimente fördern zwar in der Regel probabilistische Relationen zwischen Ursache und Wirkung zu Tage, aber oft nicht, *wie* diese miteinander verbunden sind (ebd., S. 222).

Genauere Betrachtung der Studie von Darling et al. (1989), sowie der meisten anderen hier zitierten biomechanischen Studien, kann die eben genannten Vorbehalte aber dadurch entkräften, dass die Studien den mechanistischen Zusammenhängen zwischen Ursache und Wirkung auf der Spur sind: „Control of the phasic activity of the antagonist muscles appears abnormal in the elderly in comparison to young adult subjects. Whether this reflects a deficiency in the planning of movements resulting in inappropriate commands to the antagonist muscles cannot be answered in the present study. Changes in the motor neuron population and firing rate variability and mechanical properties of motor units with age may also result in greater variability in muscle force output even if the motor commands to the agonist and antagonist muscles are not more variable in elderly subjects.“ (ebd., S. 157)

Wenn für das Explanans Y anstelle des Alters nun, wie vorgeschlagen, die Population eines Muskelareals mit Motoneuronen, deren Feuerrate oder andere mechanische Eigenschaften eingesetzt werden, so resultiert ein stärker *mechanistischer* Erklärungszusammenhang.

Für „streng“ mechanistische Zusammenhänge können Kausalerklärungen der ersten Art gegeben werden, die Beckermann (1977, S. 43 f.) auch als starke kausale Erklärungen bezeichnet und für die sich die Regeln der deduktiven Logik anwenden lassen (wobei auch die deduktive Logik eine Überprüfung erfordert, ob die Prämissen erfüllt sind – hier z. B., ob die Zahl der Motoneuronen eines Individuums reduziert ist –, bevor ein Schluss auf einen konkreten Fall angewendet werden kann, vgl. Stegmüller, 1958, S. 7.):

> Der Satz „X, weil Y“ ist eine korrekte kausale Erklärung der ersten Art genau dann, wenn Y eine hinreichende Bedingung oder Element eine[r] Menge von hinreichenden Bedingungen für X und unter den gegebenen Umständen auch eine notwendige Bedingung für X war, d.h. wenn X aufgrund von Y (und gegebenenfalls einigen anderen Bedingungen) auftreten „mu[ss]te“ und wenn X nicht aufgetreten wäre, falls Y nicht der Fall gewesen wäre. [Fußnote entfernt, Orthografie angepasst] (Beckermann, 1977, S. 43)

Bezüglich der Motoneuronenpopulation im Alter, stellten M. J. Campbell et al. (1973) fest, dass die Zahl der sog. motorischen Einheiten abnimmt. Die motorischen Einheiten bestehen dabei aus Muskelfasern sowie dem sie innervierenden Motoneuron (Rinkenauer, 2008, S. 147), d. h. eine Abnahme der motorischen Einheiten bedeutet eine Reduktion in der Zahl der Motoneuronen. M. J. Campbell et al. (1973) beobachteten gleichzeitig eine größere Querschnittsflä-

che der motorischen Einheiten, was sie als Indikation dafür betrachten, dass die motorischen Einheiten Muskelfasern (re-) innerviert haben könnten. Das heißt, es stehen im Alter nur noch weniger, aber größere motorische Einheiten zur Verfügung. Die Kraftausübung kann nicht mehr so fein reguliert werden (Rinkenauer, 2008, S. 148), was offenbar eine Konsequenz der Reorganisation an den motorischen Einheiten ist (Galganski et al., 1993).

Bezüglich der Feuerraten der Motoneuronen, konnten Roos et al. (1999) zumindest für isometrische Kraftausübung der Quadriceps-Muskeln[14] keine Abnahme mit dem Alter nachweisen.

Bezug nehmend auf andere mechanische Eigenschaften der motorischen Einheiten nimmt im Alter die Dicke der Spindelkapseln durch Kollageneinlagerung zu und die mittlere Zahl der intrafusalen Fasern, d. h. der Fasern innerhalb der Spindeln, ab (Swash und Fox, 1972). Die Muskelspindeln geben Rückmeldung über Muskeldehnung und -spannung und sind somit zur Einstellung der Körper- bzw. Gelenkstellung erforderlich (Rinkenauer, 2008, S. 148 f.). Swash und Fox (1972) halten es dabei für möglich, dass hinter der abnehmenden mittleren Zahl intrafusaler Fasern und den beobachteten Änderungen, die mit der Denervierungserklärung konsistent sind, eine ähnliche neuronale Ursache steht. Die hiermit verbundenen Beeinträchtigungen betreffen direkt die sogenannte Propriozeption, d. h. die Eigenwahrnehmung des Körpers, durch Sensoren in den Gelenken und Muskeln (Rinkenauer, 2008, S. 148).

Propriozeptive Rückkopplung ist sowohl für hohe Genauigkeit der Bewegungstrajektorien als auch für hohe Genauigkeit der Bewegungsendpunkte notwendig (Berardelli et al., 1996). Adamo et al. (2007) führten hierzu eine Untersuchung durch, bei der junge und ältere Probanden ihren Unteram in einer horizontalen Bewegung um genau 10°, 30° und 60° ausscheren sollten. Dabei gab es drei Modi: (1.) Der rechte Unterarm wurde passiv aus- und wieder zurückgeschert und die Probanden sollten den Unterarm danach um den gleichen Winkel aktiv, also selber, wieder ausscheren. (2.) Der rechte Unterarm wurde passiv ausgeschert, in dieser Position belassen und die Probanden sollten danach selber den linken Unterarm um den gleichen Winkel ausscheren. (3.) Der rechte Unterarm wurde passiv aus- und wieder zurückgeschert und die Probanden sollten danach selber den linken Unterarm um den gleichen Winkel ausscheren. Die Nachahmungsbewegung der Referenzausscherung dauerte bei den älteren Probanden länger und wies gleichzeitig mehr Geschwindigkeitsspitzen als bei den jungen Probanden auf (ebd., S. 1305). Adamo et al. (ebd., S. 1307) kommen unter Berücksichtigung ähnlicher Studien für die unteren Gliedmaßen, zu der Schlussfolgerung, dass diesem Phänomen eine allgemeine

[14] Oberschenkelmuskulatur

Beeinträchtigung der Propriozeption im Alter zugrunde liegt.

Da viele scheinbar einfache Aktionen, wie das Hinsetzen auf einen Stuhl oder das Greifen von Objekten, präzise koordinierte Bewegungen erfordern, haben Beeinträchtigungen der Fähigkeit mittels Propriozeption die Stellung der eigenen Körperglieder zu bestimmen, erheblich negatives Ausmaß für die Alltagskompetenzen älterer Menschen (Rinkenauer, 2008, S. 149).

2.4.4 Gleichgewicht

Der auch in der Beinmuskulatur feststellbare Rückgang der propriozeptiven Leistung im Alter spielt auch für den sicheren Stand und damit mutmaßlich auch für den sicheren Gang eine Rolle (Hurley et al., 1998). Hurley et al. (ebd.) ließen jüngere und ältere Probanden (1.) auf *beiden* Beinen mit *offenen* Augen, (2.) auf *beiden* Beinen mit *geschlossenen* Augen und (3.) auf *einem* Bein mit *offenen* Augen stehen. Während in der 1. Situation ältere und junge Probanden vergleichsweise gleichstabil stehen konnten, war der Stand in der 2. und 3. Situation bei den älteren Personen erheblich instabiler als bei den jungen.

Die geringere Standstabilität in der 2. Situation ist dabei auf die beschriebenen Beeinträchtigungen der Propriozeption an Muskel- und Gelenksensorik zurückführbar. Hytönen et al. (1993) zeigten bereits früher, dass für ältere Menschen eine visuelle Rückkopplung zur Haltungskontrolle sehr wichtig ist. Erst bei Wegfall der visuellen Rückkopplung werden offenbar etwaige propriozeptive Defizite deutlich. Dadurch, dass im Alter die Sehkraft typischerweise schwächer wird (Lüder und Böckelmann, 2011), lässt diese Rückkopplungsmöglichkeit allerdings auch nach. Die geringere Standstabiltät der älteren Probanden in der oben beschriebenen 3. Situation von Hurley et al. (1998) legt allerdings darüber hinaus nahe, dass die Fähigkeit, das Gleichgewicht zu halten noch von weiteren Faktoren determiniert wird.

Bei der Untersuchung altersbedingter Veränderungen der Fähigkeit, das Gleichgewicht zu halten, wiesen G. Wu (1998) für die Haltungskontrolle auf einer beweglichen Fläche einen Zusammenhang zur Vibrationswahrnehmungsfähigkeit an den Füßen sowie zur Stärke der Sprunggelenksmuskulatur[15] nach. Dieser Zusammenhang gilt sogar für das Halten des Kopfes (ebd.).

Wie bis hierhin bereits dargelegt wurde, geht das Altern mit mehreren beobachtbaren physiologischen Veränderungen einher. Es könnte sein, dass abnehmende Vibrationswahrnehmungsfähigkeit an den Füßen und geringe Sta-

[15] Genauer für die sog. Plantarflexion, d. h. für das Beugen (*Flexion*) des Fußes in Richtung Sohle (*plantar*)

bilität zwar gemeinsam zu beobachten sind, aber durch einen dritten Faktor gleichzeitig „erzeugt“ werden. Eine solche Scheinkorrelation läge bspw. für die Vibrationswahrnehmungsfähigkeit vor, wenn der Abbau des Nervensystems sowohl zu schwächerer Muskulatur als auch zu geringerer taktiler Sensibilität führt, aber nur die schwächere Muskulatur tatsächlich kausal für die Körper- und Kopfhaltung verantwortlich ist. G. Wu (1998) geben dazu an, dass die Stärke der Sprunggelenksmuskulatur die signifikanteste Korrelation mit den Haltungsbewegungen als Reaktion einer plötzlichen Bewegung der Standfläche aufweist. Gleichwohl ist der Zusammenhang zwischen Vibrationssensibilität an den Füßen und der Haltungskontrolle statistisch signifikant und auch in anderen Studien bestätigt (Era et al., 1996; Lord et al., 1991). Der Zusammenhang ist durchaus plausibel, weil sich Änderungen der Standfläche an der Fußsohle bemerkbar machen. Stevens und Choo (1996) zeigten auf, dass die Füße (und Hände) im Vergleich zu den anderen Körperregionen sogar besonders stark von der im Alter typischerweise zurückgehenden taktilen Sensibilität betroffen sind (und vermuten, dass dies auf sinkenden Blutfluss im peripheren Kreislauf zurückgehen könnte oder auf die Zerstörung der Rezeptoren durch mechanische und thermische Einwirkungen, denen die Extremitäten im Laufe des Lebens stärker ausgesetzt sind als die eher zentral liegenden Körperregionen).

Dass die Stärke der Muskulatur eine ausschlaggebende Bedeutung für die Haltungskontrolle hat, ist aber alleine auch dadurch schon begründet, dass die Körperhaltung des Menschen mittels ungefähr 750 Skelettmuskeln des Bewegungsapparates reguliert wird (Barin, 1989).[16]

Der eigentliche Gleichgewichtssinn befindet sich bei Menschen im sog. vestibulären Labyrinth (Sloane, Baloh et al., 1989) im Innenohr. Dieses ist vergleichbar mit Inertialsensorik, welche die eigene Körperlage durch Sensoren für Winkel- und lineare Beschleunigungen erfasst (Valko et al., 2012). Sloane, Baloh et al. (1989) berichten, dass das vestibuläre System im Alter von Zellverlusten, synaptischen und elektrophysiologischen Veränderungen sowie Verlusten und Veränderungen in der Mikroumgebung der neuronalen Zellen betroffen sind. Gleichzeitig berichten sie aber auch, dass das vestibuläre System bei „normalem“ Altern (d. h. ohne spezielle pathologische Ausprägungen) nur geringfügigem Funktionalitätsverlust unterworfen ist (ebd.). Bei den meisten älteren Menschen mache es sich zwar bemerkbar, dass sie langsamer gehen und den Körper vorsichtiger drehen müssen sowie einen unsichereren Stand als in jüngeren Jahren haben, allerdings beklagten sich diese selten über vestibuläre Beschwerden (ebd.). Sloane, Baloh et al. (ebd.) verweisen darauf, dass

[16] Barin (1989) evaluierte für den Bewegungsapparat ein mathematisches Modell, welches Grundlage für „streng“ mechanistische Erklärungen darstellen kann.

Schwindelgefühl und Ähnliches eher auf spezifische pathologische Indikationen oder auf die Einnahme bestimmter Medikamente zurückgeht und kein „rein alters*bedingtes*[17]“ Phänomen darstellt (siehe auch: Sloane, Coeytaux et al., 2001).

Mit einer alterskorrelierten Prävalenz von ca. 30% bei Über-60-Jährigen ist Schwindel ein bedeutendes Problem (vgl. Fernández et al., 2015; Sloane, Coeytaux et al., 2001), gerade auch im Hinblick auf das Ziel, die Safety für Passanten mit alterstypischen Beeinträchtigungen zu erhöhen.

2.4.5 Pathologische Beeinträchtigungen

Abschließend soll noch auf einige pathologisch bedingte Beeinträchtigungen der Mobilität zu Fuß im Alter eingegangen werden. Dies sind solche, die nicht als „normal“ gelten, sondern als krankhaft. Menschen in höherem Alter werden im Gegensatz zu den zuvor erläuterten Beeinträchtigungen nicht notwendigerweise[18] davon betroffen. Gleichwohl handelt es sich bei den im Folgenden beschriebenen pathologischen Beeinträchtigungen um alterskorrelierte Phänomene.

Schwindelanfälle wurden bereits oben als ein solches pathologisches Phänomen charakterisiert. Sloane, Coeytaux et al. (2001) beschreiben drei Ausprägungen von Schwindelgefühl wie folgt: (1.) Die betroffene Person hat das Gefühl, dass sie oder ihre Umgebung sich bewegt bzw. dreht (*Dreh- oder Schwankschwindel*). (2.) Die betroffene Person hat das Gefühl in Ohnmacht zu fallen (*Präsynkope* bzw. *Ohnmachtsschwindel*). (3.) Die betroffene Person hat das Gefühl von Ungleichgewicht (a) primär in den unteren Extremitäten, (b) hauptsächlich beim Stehen oder Gehen oder (c) beim Hinsetzen oder Hinlegen (*Lageschwindel*).

Die Ätiologie, d. h. mögliche Ursachen, klassifizieren Fernández et al. (2015, S. 3) in den sechs Kategorien (engl.) „Peripheral vestibular“, „Central nervous system“, „Cardiovascular“, „Medications“, „Multimodal balance disorder“ und „Others“ mit insgesamt 25 Faktoren (über alle Kategorien hinweg). Dies zeigt, dass die hinter diesem Krankheitsbild liegenden Faktoren aus medizinischer Sicht sehr differenziert sein können. Da diese Arbeit aber keinen medizinischen Fokus hat, soll im Hinblick auf pathologische Beeinträchtigungen die phänomenologische Beschreibungen genügen.

[17] Wie in Abschnitt 2.4.2 angesprochen, bedingt das Alter *an sich* freilich nichts, wohl aber biologische Mechanismen, die zu sog. Seneszenz (Kuilman et al., 2010) führen.

[18] Der Begriff der Notwendigkeit ist hier zu relativieren: Denn die Notwendigkeit der zuvor erläuterten Beeinträchtigungen gilt nach gegenwärtigem Stand des Wissens und eben nur für Menschen, die überhaupt ein gewisses Alter erreichen.

Ein weiterer alterskorrelierter pathologischer Befund ist Anämie, d. h. sog. Blutarmut (Anía et al., 1997). Die physiologisch-funktionalen Konsequenzen von Blutarmut sind eine geringere Sauerstoffversorgung der „Bedarfsträger“ im Körper (Muskeln, Organe) durch die geringe Konzentration des sauerstoffbindenden und -transportierenden Hämoglobins im Blut. Eine umfassende schematische Übersicht der *hämodynamischen* Veränderungen bei Anämie bieten Metivier et al. (2000). Generell kommt es bei reduzierter Sauerstoffversorgung aufgrund einer Erkrankung des Herz-Kreislaufsystems oder der Atemwege zur Reduzierung der aeroben Leistungsfähigkeit, d. h. für Leistung, bei der die notwendige Energiebereitstellung durch Oxidation mit Sauerstoff erfolgt, was z. B. gerade für länger andauernde motorische Leistung der Fall ist (vgl. Rinkenauer, 2008, S. 157 f.).

Herz-Kreislauf-Erkrankungen geht häufig Arteriosklerose, d. h. eine Arterienverschlusskrankheit, voraus (Saß et al., 2010). Arteriosklerose und damit auch Herz-Kreislauf-Erkrankungen weisen dabei alterskorrelierte Prävalenz auf (Saß et al., 2010; Debus et al., 2013). Die periphere arterielle Verschlusskrankheit (PAVK) ist dabei besonders prävalent im Alter (Debus et al., 2013; Criqui et al., 1997). Bei der PAVK entstehen durch mangelnde Durchblutung der Beinmuskulatur schon beim Gehen einer Strecke von weniger als 200 m mäßige bis starke Schmerzen an den Beinen (Ploenes, 2019; Espinola-Klein, 2011). Nach einer Pause gehen die Schmerzen aufgrund der damit verbundenen muskulären Entlastung aber auch wieder zurück (Ploenes, 2019; Espinola-Klein, 2011).[19]

Abschließend seien noch Arthrose, Osteoporose und Rheuma als alterskorrelierte Erkrankungen des Bewegungsapparates genannt (Saß et al., 2010). Bei Arthrose handelt es sich um eine Erkrankung der Gelenke, die Schmerzen und gelenkfunktionsspezische Einschränkungen verursacht (Zacher und Gursche, 2001). Hiervon können alle Bewegungen betroffen sein, also auch das Gehen, Hinsetzen usw. Bei Osteoporose handelt es sich um eine Erkrankung des Skeletts, die mit Abbau von Knochenmasse sowie höherer Knochenbrüchigkeit einhergeht (Minne et al., 2002). Osteoporotische Frakturen sind nicht nur sehr schmerzhaft, es können auch Beeinträchtigungen durch Osteoporose-bedingte Mikrofrakturen entstehen (ebd.). Aber auch ohne Frakturbeschwerden kann die mit Osteoporose verbundene Sturzangst zur Meidung außerhäuslicher Aktivitäten führen (Saß et al., 2010).

Bei Rheuma handelt es sich um einen Formenkreis ebenfalls mitunter schmerzhafter und einschränkender Erkrankungen des Bewegungsapparates (ebd.). Da

[19] Die periphere arterielle Verschlusskrankheit wird auch als „Schaufensterkrankheit“ bezeichnet (Ploenes, 2019), weil betroffene Passanten dazu neigen, während der Pausen in die Schaufenster zu schauen.

diese Arbeit aber, wie bereits weiter oben in diesem Abschnitt angemerkt, keinen medizinischen Fokus hat, sollen die rheumatische Erkrankungen hier nicht weiter differenziert werden.

2.4.6 Schlussfolgerungen

Trotz medizinischen und technischen Fortschritts ist das Altern mit Beeinträchtigungen der Motorik verbunden. Abbildung 2.2 fasst den Sachverhalt zusammen. Den dargestellten Sachverhalt ergänzend weisen Shepard (1999) und das U.S. Department of Transportation (1997, S. B-3 u. B-5) darauf hin, dass im Alter durch den allgemein geringeren Flüssigkeitshaushalt auch die Temperaturregulierungsfähigkeit beeinträchtigt ist. Dies kann zu verhältnismäßig schneller Erschöpfung beitragen.

Die in Tabelle 2.1 formulierten Folgen alterstypischer Beeinträchtigungen für die Benutzbarkeit des urbanen Raums betreffen zwei Probleme:

(a) Erhöhtes Pausenerfordernis, vorzugsweise im Sitzen, mit geringer Toleranz bei Nicht-Erfüllung,

(b) Hinsetz- und Aufstehschwierigkeiten bei herkömmlichen Sitzgelegenheiten.

Diese Probleme können durch *Verfügbarkeit geeigneter Sitzgelegenheit* gemindert werden. Die weitere Analyse des Stands der Forschung bezieht sich daher auf das Ziel, mittels eines Systems aus zu SSOs transformierten, bzgl. (a) und (b) *adaptiven* Sitzgelegenheiten zur Problemlösung beizutragen.

2.5 Adaptive Sitzgelegenheiten und ähnliche Anwendungen

2.5.1 Öffentliche Sitzgelegenheiten mit Sensorik und Aktuatorik

Als öffentliche dedizierte Sitz- und Pausengelegenheiten dienen im urbanen Raum v. a. Parkbänke. Dekel et al. (2005, S. 227) beschreiben Parkbänke als eine der „ubiquitärsten“ Elemente des urbanen Raums. Die Charakterisierung als ubiquitär kann dabei durchaus in Anlehnung an Weiser (1991) verstanden werden, der in Bezug auf „ubiquitous computing“ meinte, dass IT auf der einen Seite immer allgegenwärtiger würde und gleichzeitig auf der anderen Seite immer weniger sichtbar. Diese Entwicklung kann mittlerweile in weiten

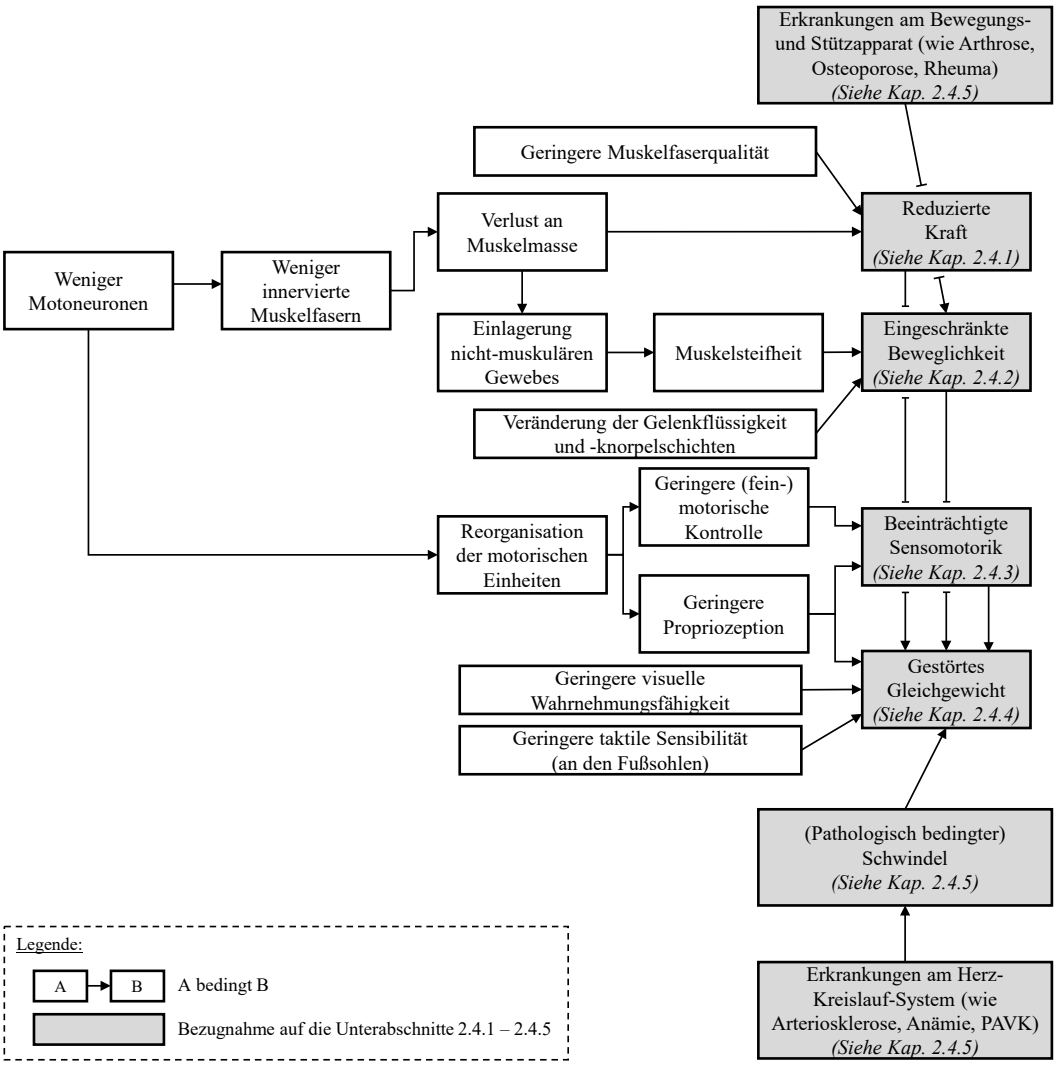


Abbildung 2.2: Zusammenfassung alterstypischer motorischer Beeinträchtigungen und ihrer Ursachen

Beeinträchtigung	**Folgen hinsichtlich Erfordernis von** **(a) → Ruhepausen** **(b) → speziellen Sitzgelegenheiten**
Reduzierte Kraft *(siehe Kap. 2.4.1)*	(a) Reduzierte Belastbarkeit und dadurch leichtere Ermüdbarkeit beim Gehen (b) Schwierigkeiten, den Körperschwerpunkt kontrolliert vertikal zu ändern
Eingeschränkte Beweglichkeit *(siehe Kap. 2.4.2)*	(b) Schwierigkeiten, in die Knie zu gehen
Beeinträchtigte Sensomotorik *(siehe Kap. 2.4.3)*	(b) Beeinträchtigte Haltungskontrolle durch beeinträchtigte körperliche Selbstwahrnehmung und inadäquate Motorik für reflexartige Ausgleichsbewegungen
Gestörtes Gleichgewicht *(siehe Kap. 2.4.4)*	(b) Sturz-/Verletzungsrisiko bei jeder willkürlichen oder notwendigen Veränderung des Körperschwerpunkts
Schwindel *(siehe Kap. 2.4.5)*	(a) (Mitunter plötzliche) Beeinträchtigung des sicheren Ganges und Standes (b) Unsicherheit/Verletzungsgefahr beim Hinsetzen auf eine (normal-) tiefe Sitzfläche
Erkrankungen am Herz-Kreislauf-System *(siehe Kap. 2.4.5)*	(a) Vergleichsweise rasche konditionelle Erschöpfung (und bei PAVK: Auftretende Schmerzen beim Gehen)
Erkrankungen am Bewegungs- und Stützapparat *(siehe Kap. 2.4.5)*	(a) Schmerzen bei den Gehbewegungen (b) Schmerzen bei den Hinsetz- und Aufstehbewegungen

Tabelle 2.1: Folgen alterstypischer motorischer Beeinträchtigungen in Bezug auf die sichere Benutzbarkeit des urbanen Raumes

Teilen als bestätigt erachtet werden und auch die vorliegende Arbeit ordnet sich durch die Konzeption von SSOs darin ein.

Die Charakterisierung von Parkbänken als ubiquitär im o. g. Sinne trifft aber auch deshalb durchaus zu, weil Parkbänke auf der einen Seite zahlreich im urbanen Raum vorkommen, während ihnen auf der anderen Seite im Allgemeinen eher geringe Salienz zukommt, d. h. möglicherweise überhaupt nicht bewusst wahrgenommen werden. Letzteres gilt vor allem, wenn ein Passant gerade keinen Bedarf nach einer Sitzgelegenheit hat. Aber auch wenn durch „Smartifizierung“ einer einzelnen Sitzgelegenheit nur ein scheinbar geringer direkt wahrnehmbarer Nutzen geschaffen wird, so können sich auch kleine Effekte kumulieren und zu einer deutlich verbesserten Benutzbarkeit des urbanen Raums führen. Dies gilt insbesondere für die intelligente Vernetzung adaptiver Sitzgelegenheiten.

Trotz der Bedeutung, die Sitzgelegenheiten im urbanen Raumes haben können, ließen sich in der Literatur nur wenige dedizierte Beiträge finden, welche die Möglichkeiten IT-gestützter, sich irgendwie adaptiv verhaltender Sitzgelegenheiten eruieren. Dekel et al. (2005, S. 226 ff.) berichten über Prototypen für sog. „Musical Chairs“ und eine „Intimate Bench“. Die „Musical Chairs“ (ebd., S. 226 f., auch im Weiteren) sind für den urbanen Raum bestimmte Sitzblöcke, die mit visuellen und auditiven Interaktionselementen ausgestattet sind. Wenn sich ein Passant auf einen dieser Sitzblöcke niederlässt, startet eine audio-visuelle Sequenz. Auch dazu sind die Sitzflächen mit Sensoren ausgestattet, die mit einem Mikrocontroller verbunden sind, der wiederum Leuchten und Lautsprecher steuert.

Die eigentlich intendierten Interaktionen beginnen aber erst, wenn zwei oder mehr Passanten beteiligt sind. Wenn ein Passant bereits sitzt und ein weiterer hinzukommt, der sich auf einen Musical Chair in der Nähe setzt, dann erzeugen die Sitzblöcke, die zwischen diesen beiden Personen liegen, audio-visuelle Animationen. Wenn nun ein dritter Passant hinzukommt und sich auf einen der dazwischenliegenden Sitzblöcke setzt, dann unterbricht dieser gleichsam den Animationsfluss. Die Animationen werden dann absichtlich nur noch zwischen dem neu hinzugekommenen und dem als allererstes dagewesenen Passanten fortgesetzt. Die Idee dahinter ist, dass die zweite Person sich jetzt bewegen muss und sich auf einen neuen Block hinsetzen muss, damit sie wieder in die Animationen einbezogen ist.

In den von Dekel et al. (ebd.) geschilderten Beobachtungen, sind Passanten beim ersten Hinsetzen zwar zunächst von den audio-visuellen Effekte überrascht, testen daraufhin aber doch durch gezieltes Platzwechseln die audio-visuellen Reaktionen spielerisch aus. Üblicherweise formt sich dann allmählich eine Menschentraube um die Musical Chairs und ein zweiter und dritter

Passant steigen schließlich in die spielerischen Interaktionen mit ein.

Bei der Intimate Bench (ebd., S. 227 f., auch im Weiteren) handelt es sich um eine gewöhnlich aussehende Parkbank, auf dessen Sitzfläche Sensoren angebracht sind, die mit einem Mikrocontroller verbunden sind. Der Mikrocontroller steuert dann Leuchten, die in die Rückenlehne der Parkbank eingelassen sind. Wenn sich ein Passant auf die Intimate Bench setzt, dann wird dies durch die Sitzflächensensoren erkannt und diese Information durch den Mikrocontroller verarbeitet. Abhängig von der Sitzposition auf der Sitzfläche, werden dann verschiedene Leuchtmuster aktiviert.

Das eigentliche Ziel ist, Passanten, die auf der Bank Platz nehmen, dazu zu bringen, miteinander zu kommunizieren und sich näherzukommen. Wenn zwei Passanten mit großem Abstand zueinander auf der Bank sitzen, dann formen die Leuchten in der Rückenlehne bunte Dreiecke und Herzen. Dadurch soll eine Interaktion zwischen den beiden Passanten in Gang gesetzt werden. Dekel et al. (ebd.) berichten, dass Betrachter der Intimate Bench von dem plötzlichen Aufleuchten der Formen in der Rückenlehne sehr angetan gewesen sind, vor allem, weil die aufleuchtenden Formen in deaktiviertem Zustand gar nicht sichtbar sind.

Okada et al. (2016, auch im Weiteren) stellen einen Prototypen für Parkbänke als sog. „Social Things“ vor. Als Social Things werden Objekte definiert, die (1.) eine gewissen Autonomie gegenüber strikter Benutzerkontrolle haben sollen, eigene Ziele verfolgen und eine eigene Gesellschaft haben, (2.) bei Interaktionen mit anderen Objekten und Benutzern lernen sollen, und Kontext verstehen müssen sowie (3.) dynamisch kommunizieren, um sich gegenseitig mit komplexeren vernetzten Diensten bei der Erreichung ihrer Ziele zu helfen. Das Interaktionsnetzwerk soll dabei auf physischer Nähe bzw. Annäherung basieren.

Benutzer tragen ein kleines Schlüsselanhänger-ähnliches IT-Endgerät mit Netzwerkfunktionalität, das persönliche mobile Social Things repräsentiert. Mit Parkbänken als Prototyp untersuchten Okada et al. (ebd.), wie öffentliche Objekte neue Erfahrungen im Zusammenhang mit dem Konzept der Social Things liefern. Dazu wurde der Korpus einer Parkbank mit Leuchten ausgestattet, die ihre Farbe und Helligkeit anpassen können. Dazu können z. B. Informationen für eine Farbe im persönlichen Endgerät gespeichert sein und von dort an die mit dem Netzwerk verbundenen Leuchten der Parkbank übertragen werden.

Die durchgeführten Testszenarien mit einer Parkbank und drei mobilen Endgeräten zeigten schließlich die technische Machbarkeit autonomer – d. h. hier: ohne aktiv durch Menschen gesteuerter – Interaktionen. Für den Nachweis rein annäherungsbasierter Interaktionen blinkten sowohl die Leuchten im Korpus

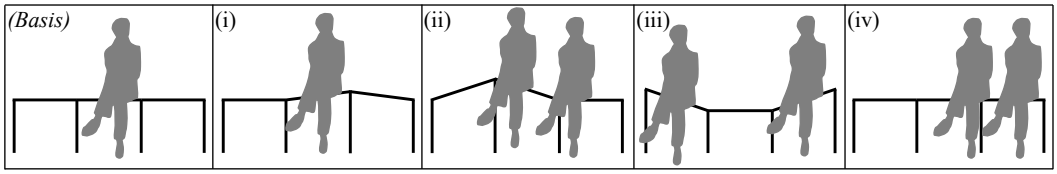


Abbildung 2.3: Konfigurationen der „coMotion“-Bank (nach Kinch et al., 2014)

der Parkbank als auch in das Schlüsselanhänger-ähnliche Endgerät integrierte LEDs jeweils in 0,5-Sekunden-Intervallen, wenn eine Bewegung innerhalb von 30 cm Entfernung detektiert wurde, und in 2-Sekunden-Intervallen, wenn eine Bewegung innerhalb von 150 cm Entfernung detektiert wurde. Für Interaktionen mit Nachrichtenaustausch leuchtete jeweils eine bestimmte von drei in die Parkbank integrierte Leuchten in der LED-Farbe des dieser Leuchte zugeordneten detektierten Schlüsselanhänger-ähnlichen Endgeräts.

Ein Prototyp für eine Parkbank, die ihre Form verändern kann, ist „coMotion“ (Kinch et al., 2014, auch im Weiteren). Die Sitzfläche dieser Bank besteht aus drei gleich großen Teilen. Vier Paare von Linearaktuatoren bzw. Hubsäulen bilden die Standbeine und sind jeweils an den Grenzbereichen der Teilsitzflächen angebracht. Die Gesamtsitzfläche kann so zu einer Mulde geformt werden, wenn die ganz äußeren Standbeine hochgefahren werden oder zu einem Hügel, wenn die inneren Standbeine hochgefahren werden. Jeder Linearaktuator kann prinzipiell einzeln gesteuert werden. Zusätzlich verfügt die Bank über sechs Kraftsensoren, die ermitteln sollen, ob und wo genau Passanten auf der Sitzfläche sitzen.

Für die Form der Sitzfläche sind einige Konfigurationen vorpogrammiert. Jede dieser Form ist auf eine bestimmte Intention ausgerichtet. Abbildung 2.3 stellt vier beispielhafte Konfigurationen schematisch dar: (i.) Wenn ein einzelner Passant auf der Bank sitzt, hebt sich die Sitzfläche ganz leicht an. Die Intention dahinter ist, Passanten mit der Formveränderbarkeit der Bank vertraut zu machen. (ii.) Wenn mehrere Passanten relativ mittig auf der Bank sitzen, dann kann es sein, dass dadurch die komplette Sitzfläche belegt wird, obwohl eigentlich noch genug Platz für weitere Passanten wäre. Die Bank formt dann einen hohen Hügel und macht somit deutlich, dass die sitzenden Passanten die Mitte frei machen könnten, indem sie zu einer Seite rücken. (iii) Wenn sich zwei Passanten in maximalem Abstand zueinander auf die Außenseiten der Bank setzen, dann hebt die Bank die Außenseiten an, sodass die Passanten näher zueinander rücken. (iv) Die Bank ändert ihre Form absichtlich nicht, wenn Passanten bereits so auf der Sitzfläche verteilt sind, dass noch genug Platz für

weitere Passanten darauf vorhanden ist.

Kinch et al. (ebd.) stellten zu „coMotion“ die Hypothesen auf, dass die Formveränderung der Bank eine nachhaltige, längerfristige Gesprächsinteraktion zwischen fremden Passanten induzieren kann. In der Tat konnten Kinch et al. (ebd.) beobachten, dass die Formveränderung der Bank durchaus für Gesprächsstoff auch zwischen fremden Passanten sorgte. Da die Gespräche allerdings recht flüchtig waren und ausschließlich das unerwartete Verhalten der Bank zum Inhalt hatten, betrachten Kinch et al. (ebd.) diese Hypothese letztlich nicht als bestätigt.

Eine Hypothese, dass die aktuelle Situation und allgemeinen Gefühlslagen der Passanten einen Einfluss auf die Nutzung und Interpretation von „coMotion“ haben, sehen Kinch et al. (ebd.) deutlicher bestätigt. Sie untersuchten das Verhalten jeweils im Foyer eines Konzertsaals, in einem Supermarkt sowie auf einem Flughafen. Im Foyer des Konzertsaals wurden die unerwarteten Formveränderungen eher positiv und mit Humor aufgenommen, während sie im Einkaufszentrum und auf dem Flughafen eher auf Zurückhaltung stieß. Dies erklären sich Kinch et al. (ebd.) damit, dass Passanten, die ein Konzert besuchen, innerlich schon auf vergnüglichen Zeitvertreib eingestellt sind und die Formveränderung der Bank als eine Art Darbietung aufnehmen. Auf einem Flughafen oder im Einkaufszentrum überwiegt eher ein gewisses Stressempfinden. Eine Bank, die unerwartet ihre Form verändert, wird dann eher so wahrgenommen, dass es sich dabei gar nicht um eine Sitzgelegenheit handelt und man durch das Daraufsetzen etwas Falsches gemacht hat (ebd.). Eine naheliegende Schlussfolgerung ist, dass Passanten über Nutzung (und intendierten Nutzen) eines neuartigen SSOs aufgeklärt werden sollten.

In den vier oben referierten Beiträgen werden informationstechnische Potentiale für urbane Sitzgelegenheiten vordergründig dafür genutzt, eher spielerisch die soziale Atmosphäre positiv zu beeinflussen. Eine Anwendung zur Erhöhung der Safety für Passanten mit alterstypischen Beeinträchtigungen ist dabei aber nicht zielgebend gewesen. Hier besteht noch Forschungsbedarf.

Abschließend soll aber noch ergänzend ein Beispiel aufgeführt werden, welches zeigt, dass zumindest hinsichtlich ergonomischer[20] Aspekte von Sitzgelegenheiten, Ansätze erforscht werden, die im Hinblick auf eine Adaptivität genutzt werden könnten – auch wenn dies nicht dediziert für Sitzgelegenheiten des urbanen Raums ausgelegt ist:

Benocci et al. (2011, auch im Weiteren) entwickelten mit einem *k-Nearest-Neighbor*-Klassifikator ein simples maschinelles Lernverfahren, mit dem Büro-

[20] *Hinweis:* Ergonomie ist die Befassung mit Gebrauchstauglichkeit als Eigenschaft (Adler et al., 2010, S. 13).

stühle die Haltung darauf sitzender Personen erkennen sollen. Dafür werden Bürostühle mit fünf Kraftsensoren, einem Magnetometer und einem Accelerometer ausgestatt. Mit den Kraftsensoren misst der Bürostuhl, ob und wie eine Person sitzt. Der Magnetometer wird als Kompass verwendet, um die Rotation zu ermitteln. Der Accelerometer misst die Linearbeschleunigung entlang der drei räumlichen Achsen. Zusätzlich trägt die sitzende Person ein *Wearable* im Bereich des Brustbeins, das ebenfalls mit einem 3-Achsen-Accelorometer ausgestattet ist. Mittels dieser Sensoren kann der Klassifikator die Sitzhaltung der Personen mit hoher Genauigkeit bestimmen ($Accuracy = 92,7\%$, Benocci et al., 2011).

Darüber hinaus wurden auch die Atmungstiefe und -frequenz sowie EKG-Daten der sitzenden Personen mittels eines als Brustgürtel getragenen Polygraphen gemessen. Benocci et al. (ebd.) benutzten diese Daten als psychometrische Indikatoren für Stress u. Ä. Solche physiologischen Daten könnten aber auch in Bezug auf die Problemstellung der vorliegenden Arbeit interessant sein, etwa um einen Erschöpfungsgrad abzuschätzen. Der Schwerpunkt von Benocci et al. (ebd.) lag allerdings weniger in der Interpretation physiologischer Daten als in der Entwicklung und Erprobung einer Software-Hardware-Architektur, die eine Echtzeitanforderung von weniger als 15 ms Latenz erfüllt.

Je nachdem, auf welche Art und Weise eine Sitzgelegenheit adaptiv sein soll, kann die Erkennung der Sitzhaltung eine relevante Anforderung sein. In der vorliegenden Arbeit könnte dies bspw. in Bezug auf das im Abschnitt 2.4.6 geschlussfolgerte Problem (b) zutreffen, wenn dadurch die Gebrauchstauglichkeit für das Hinsetzen und Aufstehen erhöht wird.

2.5.2 „Smart Parking“ als strukturell ähnliche Anwendung

In Bezug auf das im Abschnitt 2.4.6 geschlussfolgerte Problem (a) bietet der Smart-City-Anwendungsfall *Smart Parking* gewisse Struktur- bzw. Modellanalogien. Das Erfordernis nach Sitzgelegenheiten für Pausen, ist mit der Notwendigkeit nach Parkplätzen zum Parken vergleichbar. Dabei kann ein System digital vernetzter, mit Sensorik und Aktuatorik ausgestatteter Parkplätze ein Analogon zu einem System vernetzter adaptiver Sitzgelegenheiten darstellen[21] und digital vernetzte Autos ein Analogon für die Passanten (mit ihren IT-Endgeräten). Die Benutzung eines smarten Parkplatzes kann analog zur

[21] Interessanterweise hat der Wortteil „Park“ in *Park*bank – als öffentliche dedizierte Sitzgelegenheiten – und *Park*platz jeweils eine andere Bedeutung (erstere kommt von „Park“ als Grünanlage, zweitere von „parken“ als Abstellen eines Vehikels). Möglicherweise gibt es einen gemeinsamen Ursprung für die Bedeutung von „Park“. Dies zu prüfen sei gerne dem interessierten Leser überlassen.

Benutzung einer adaptiven Sitzgelegenheit aufgefasst werden. Für beides gilt, dass ein Park- bzw. Sitzplatz während einer Benutzung temporär gebunden, davor und danach aber prinzipiell für alle verfügbar ist. Der Ort eines Parkplatzes kann dabei genauso entscheidend sein, wie der Ort einer Sitzgelegenheit. Für einen Parkplatz ist es wünschenswert, dass er in der Nähe eines bestimmten Zielorts liegt und für eine Sitzgelegenheit ist es erstrebenswert, dass sie möglichst nah an dem Ort liegt, wo ein Passant ein Pausenerfordernis hat. Unterschiede bestehen darin, dass Parkplätze i. Allg. nicht der Regeneration dienen und dass jedes Fahrzeug normalerweise nur einmal parken muss, während Passanten mit körperlichen Beeinträchtigungen mitunter mehrere Pausen auf mehreren Sitzgelegenheiten benötigen. (Bezüglich der Regeneration könnten Ladestationen für elektrisch betriebene Fahrzeuge ein Analogon sein.)

Im Folgenden werden vier exemplarische Beiträge analysiert, welche verschiedene Ansätze und Schwerpunkte für smarte Parkplatzsysteme repräsentieren. Dazu wird jeweils die Eignung als Modellanalogon für adaptive Sitzgelegenheiten reflektiert.

2.5.2.1 Parkplätze als drahtloses Sensornetzwerk

Yang et al. (2012, auch im Weiteren) entwarfen ein smartes Parkplatzsystem aus informations*technischer* Perspektive. Das Ziel ist es, Autofahrern die Suche nach einem freien Parkplatz zu erleichtern. Ihre IT-Architektur umfasst eine Smartphone-App, eine Datenbank, mehrere eingebettete sowie einen zentralen Web-Server und die drahtlosen Sensornetzwerke.

Abbildung 2.4 stellt die Architektur von Yang et al. (ebd.) grafisch dar. Dort ist zu entnehmen, dass jeder Parkplatz mit einem sog. Sensorknoten ausgestattet ist, also mit einer relativ kompakten, drahtlos vernetzbaren Sensoreinheit. Diese Knoten können dann die Informationen kodieren, ob ein Parkplatz belegt ist oder nicht. Sensortechnisch wird dies mittels Lichtsensoren realisiert. Die Parkplätze sind in Zonen eingeteilt, wobei jede Zone mit einem auf einem Mini-PC eingerichteten Web-Server ausgestattet ist. Diese Web-Server werden als eingebettete Web-Server bezeichnet, da sie auf ähnlicher Hardware laufen, wie die Sensorknoten. Die Sensorknoten sind drahtlos mit dem eingebetteten Web-Server verbunden, der wiederum drahtlos mit einem zentralen Web-Server verbunden ist. Der zentrale Web-Server setzt die volle Funktionalität üblicher Web-Server um und besitzt höhere Leistungsstärke als die eingebetteten Web-Server. Der zentrale Web-Server verfügt über eine Standard-Datenbank und ist für entfernte Aufrufe mit dem Internet verbunden. Die Möglichkeit der entfernten Aufrufe über das Internet wird von Endanwendungen genutzt, die auf Smartphones von Autofahrern laufen können (oder auf PCs o. Ä.).

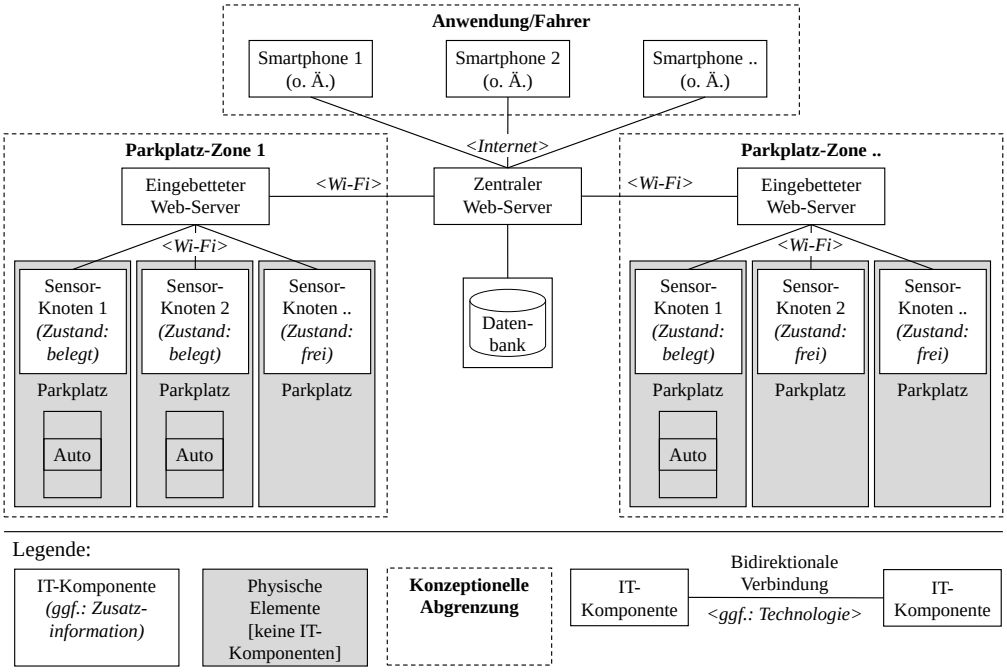


Abbildung 2.4: Architektur für Parkplätze als drahtlose Sensornetzwerke (nach Yang et al., 2012)

Die Gesamtfunktionalität der Architektur kann anhand der Aufgaben ihrer Komponenten wie folgt beschrieben werden: Die *Sensorknoten*, überwachen fortlaufend den Zustand eines Parkplatzes durch Erhebung der entsprechenden Lichtsensordaten und senden den Zustand an den eingebetteten Webserver ihrer Parkplatz-Zone. Die *eingebetteten Web-Server* horchen auf Änderungen der Parkplatzzustände und senden die Parkplatzzustände mit der Position der Parkplatz-Zone an den zentralen Web-Server. Der *zentrale Web-Server* empfängt die Informationen von den eingebetteten Web-Servern, speichert sie in seiner Datenbank und erstellt eine Visualisierung. Die Visualisierung erfolgt dabei in sehr kurzfristigen Zyklen, von Yang et al. (2012) als in Echtzeit bezeichnet. Zusätzlich sendet der zentrale Web-Server die Informationen über die Parkplatzzustände an die Smartphones o. Ä. der Fahrer. Die *Smartphones o. Ä.* als Endgeräte der Fahrer verbinden sich mit dem zentralen Web-Server, empfangen so die Informationen über die Parkplatzzustände und visualisieren diese, sodass die Fahrer verfügbare Parkplätze sehen können.

In Hinblick auf die informationstechnische Realisierung liefert der Beitrag von Yang et al. (ebd.) einen nützlichen Ansatz. Insbesondere die Hierarchisierung mittels der Parkplatz-Zonen, eingebetteter Web-Server und des zentralen Web-Servers kann als Muster einer IT-Architektur für ein System adaptiver Sitzgelegenheiten dienen. Eine adaptive Parkbank, d. h. eine Sitzgelegenheit mit mehreren Sitzplätzen, kann dabei in der Architektur wie eine Parkplatz-Zone abgebildet werden. Jede adaptive Parkbank verfügt dann je Sitzplatz über einen Sensorknoten, welcher den Belegungszustand kodiert, und einen eingebetteten Web-Server. Die Sensorknoten senden den Belegungszustand dann an den eingebetteten Web-Server der adaptiven Parkbank. Auf dem zentralen Web-Server laufen dann die Informationen zusammen, die auch für eine aktive Verfügbarkeitssteuerung genutzt werden könnten.

Der Prototyp von Yang et al. (ebd.) stellt die technische Machbarkeit einer effektiven, IT-basierten Unterstützung zum Auffinden von Parkplätzen in den Vordergrund. Autofahrer können „von Weitem“ sehen, ob ein Parkplatz zu einem momentanen Zeitpunkt frei ist oder nicht. Darin ist allerdings keine aktive Verfügbarkeitssteuerung enthalten. Parkplätze können in keinster Weise reserviert werden. Ein Leitsystem, das den Weg zu einem geeigneten freien Parkplatz weisen könnte, ist nicht enthalten.

2.5.2.2 Parkplatzsuche mittels „Crowdsourcing“

Chen et al. (2012, auch im Weiteren) entwickelten ein System, welches ein koordiniertes Leitsystem enthält und dabei auf *User-Generated Content* zurückgreift, d. h. auf manuelle Eingaben menschlicher Benutzer. Diesen Rückgriff auf

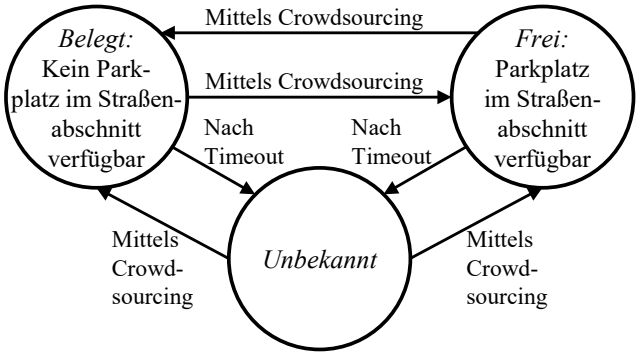


Abbildung 2.5: Zustandsdiagramm für Parkplatzsuche mittels „Crowdsourcing“ (nach Chen et al., 2012)

manuelle Eingaben bezeichnen Chen et al. (2012) als „Crowdsourcing“. Das System beschafft sich relevante Informationen unter Mithilfe von Benutzer-„Massen“ (die „Crowd“). Jeder Benutzer in der sog. Crowd muss mit einem relativ kleinen Aufwand dazu beitragen, dass ein umfassendes Bild über die Parkplatzsituation entstehen kann. Dieses umfassende Bild wird dann für die Benutzer nutzbringend verwendet. Die Funktionsweise besteht im Wesentlich darin, dass die Fahrer, die das System nutzen und gerade geparkt haben, über eine Internet-Applikation mittels eines mobilen Endgeräts gefragt werden, ob sich im Straßenabschnitt, wo sie parken, noch weitere freie Parkplätze befinden.

Abbildung 2.5 stellt die möglichen Informationszustände zur Parkplatzverfügbarkeit eines Straßenabschnitts als Zustandsdiagramm dar (ebd.). Ein Straßenabschnitt gilt als frei, wenn bekannt ist, dass dort noch mindestens ein Parkplatz verfügbar ist. Wenn bekannt ist, dass kein Parkplatz auf einem Straßenabschnitt verfügbar ist, dann gilt der Abschnitt als belegt. Im Initialzustand ist allerdings für alle Straßenabschnitte die Parkplatzverfügbarkeit unbekannt. Dieser Zustand wechselt erst entsprechend durch Crowdsourcing-Informationen, d. h. durch die manuelle Eingabe eines Benutzers oder zusätzlich mittels sog. Crowdsourcing-Sensordaten, was im nächsten Absatz beschrieben wird. Außerdem ist eine Dauer definiert, nach dessen Ablauf der Zustand wieder auf „unbekannt“ wechselt, wenn zwischenzeitlich keine weitere Crowdsourcing-Information über den betreffenden Straßenabschnitt eingeht. Wenn aber solche Information innerhalb des *Timeout*-Fensters eingeht, dann wechselt der Zustand entsprechend der Information zwischen „belegt“ und „frei“

Über manuelle Eingaben hinaus kann das System auch durch Inferenz automatische Schlussfolgerungen über die Verfügbarkeit von Parkplätzen ziehen.

Wenn ein Auto der Parkplatzempfehlung des Systems folgt, aber im anvisierten Straßenabschnitt nicht parkt, sondern langsam weiterfährt, dann kann davon ausgegangen werden, dass auf dem betreffenden Straßenabschnitt doch kein Parkplatz frei ist. Außerdem kann für alle Straßenabschnitte, die das selbe Auto mit langsamen Tempo danach durchfährt, davon ausgegangen werden, dass dort auch kein Parkplatz frei ist. Denn das langsame Durchfahren, ohne zu parken, ist ein Indikator für erfolglose Parkplatzsuche. Umgekehrt kann automatisch geschlussfolgert werden, dass ein Parkplatz wieder frei ist, wenn ein Auto von diesem Parkplatz wegfährt. In diesem Sinne bezeichnen Chen et al. (ebd.) Sensordaten über Geschwindigkeit und Position eines Fahrzeuges als sog. Crowdsourcing-Sensordaten.

Die Kombination manueller Eingaben mit automatischer Inferenz kann sich als besondere Stärke herausstellen, da sie für eine wechselseitige Plausibilitätsprüfung als Korrektiv dienen kann. Hier entsteht allerdings die Schwierigkeit, wie Widersprüche aufzulösen sind. Manuelle Eingaben können mitunter falsch sein – z. B. wenn ein Fahrer freie Parkplätze im Straßenabschnitt übersah oder in Eile versehentlich den falschen Eingabeknopf drückte – oder fehlen. Automatische Inferenz kann allerdings ebenso unsachgemäß sein – z. B. wenn sich ein Fahrer gegen einen zugewiesenen freien Parkplatz entscheidet und den entsprechenden Straßenabschnitt dennoch langsam durchfährt. Dieses Problem der Auflösung etwaig widersprüchlicher Informationen wird im Ansatz von Chen et al. (ebd.) nicht thematisiert.

Der Fokus liegt bei Chen et al. (ebd.) auf einem Verfahren für eine koordinierte Parkplatzsuche. Zwar könnten, ähnlich wie beim zuvor beschriebenen Ansatz von Yang et al. (2012), die Crowdsourcing-Informationen dazu genutzt werden, die Verfügbarkeit von Parkplätzen auf den Straßenabschnitten zu visualisieren und den Fahrern letztlich die Entscheidung selbst zu überlassen, wo sie basierend auf diesen Informationen parken möchten. Allerdings ist dies Chen et al. (2012) zufolge mit der Problematik behaftet, dass es den Benutzern schwerfallen könnte, während der Fahrt sämtliche mögliche Parkalternativen zu erfassen und stattdessen immer nur eine bestimmte Teilmenge im Hinblick auf deren Verfügbarkeit beobachten. Dadurch könnten die Fahrer andere, günstig gelegene Parkmöglichkeiten übersehen. Außerdem soll verhindert werden, dass sich zwei Fahrer gleichsam im Wettbewerb um denselben Parkplatz befinden, was passieren kann, wenn für Benutzer die gleichen verfügbaren Parkplätze angezeigt werden. Der Algorithmus zur koordinierten Parkplatzsuche von Chen et al. (ebd.) kann informal wie folgt beschrieben werden (ohne Parkplatzwiederfreigabe):

– Bei Parkplatzanfrage eines Benutzers:

    - Suche die Menge der Straßenabschnitte mit verfügbaren Parkplätzen, die am nächsten zum Zielort des Benutzers liegt.
      (*Hinweis:* Dass mehrere Straßenabschnitte gleich weit von einem Ort entfernt sein können, wird plakativ, wenn der Ort das Zentrum einer Kreuzung ist.)
    - Wähle unter den zuvor herausgesuchten Straßenabschnitten jenen, welcher am nächsten zur aktuellen Position des Benutzers liegt.
    - Reduziere die Kapazität des Straßenabschnitts um eins.
- Bei erfolgtem Parken:
    - Wähle den Straßenabschnitt, auf dem der Benutzer geparkt hat.
    - Frage den Benutzer nach der Parkplatzverfügbarkeit auf dem Straßenabschnitt.
    - Passe die Information zur Parkplatzverfügbarkeit entsprechend der Benutzerantwort an.
- Bei langsamen Durchfahren eines Straßenabschnitts:
    - Wähle den Straßenabschnitt, den der Benutzer langsam durchfährt (ohne zu parken).
    - Setze den Zustand dieses Abschnittes auf „belegt“, bzw. seine Kapazität auf den Zahlenwert Null.
    - Suche die Menge der benachbarten Straßenabschnitte mit verfügbaren Parkplätzen.

Bei solch einem System kann es zu einem sog. Trittbrettfahrer-Phänomen kommen, wenn Benutzer zwar die Empfehlungen nutzen, selbst aber keine aktiven Eingaben machen, d. h. Nutzen aus dem System ziehen, ohne den für die Nutzengenerierung eigentlich erforderlichen Aufwand zu betreiben. Dies kann dann funktionieren, wenn genügend andere Benutzer durch ihre aktiven Eingaben ein hinreichendes Bild der Parkplatzsituation liefern. Chen et al. (2012) untersuchten dieses Trittbrettfahrerproblem für ihren Ansatz und kommen zu dem Schluss, dass der Ansatz, gemessen an der Parkplatzsuchzeit, robust gegenüber Trittbrettfahrern ist, auch wenn nur 15% der Benutzer durch ihre Eingaben aktiv zum Informationsbild über die Parkplatzsituation beitragen. Weil die Trittbrettfahrer dann auch die koordinierte Parkplatzsuche nutzen, steigt der Anteil der eingesparten Parkplatzsuchzeit sogar bei relativ kleinem Anteil an Benutzern, die durch aktive Eingaben beitragen. Auch wenn dies auf den ersten Blick positiv überraschend erscheint, kann dies recht einleuchtend begründet werden: Je mehr Benutzer die koordinierte Parkplatzsuche nutzen, desto effektiver ist das Verfahren. Es reicht aus, dass je Straßenabschnitt nur

ein Benutzer (korrekte) Angaben über die Parkplatzverfügbarkeit macht. Darüber hinaus, wird Parkplatzwiederfreigabe durch automatische Inferenz ohnehin ohne Benutzereingabe erkannt.

Ein Ansatz wie der von Chen et al. (ebd.) wäre auch für adaptive Sitzgelegenheiten als SSOs umsetzbar. Allerdings ist angestrebt, dass adaptive Sitzgelegenheiten den Belegungszustand von Sitzplätzen kennen, ohne auf die „Crowd“ angewiesen zu sein, z. B. durch Lichtsensoren, wie im zuvor beschriebenen Ansatz von Yang et al. (2012). Die vorliegende Arbeit abstrahiert mit der Referenz auf prinzipielle Machbarkeit von der sensortechnischen Umsetzung und betrachtet die Informationsverarbeitungsprobleme. Der Algorithmus von Chen et al. (2012) ist zwar erwiesenermaßen effektiv im Vergleich zu unkoordinierter Parkplatzsuche, stellt aber eine Art *Greedy*-Verfahren dar, weil er lediglich in jedem Schritt *lokale* Optima wählt. Es ist hier fraglich, inwieweit eine Greedy-Strategie auch für ein Verfügbarkeitsproblem von Sitzgelegenheiten im urbanen Raum sinnvoll ist. Obwohl der Algorithmus von Chen et al. (ebd.) die durchschnittliche Distanz von Parkplatz zu gewünschtem Zielort gegenüber unkoordinierter Parkplatzsuche reduzieren kann, soll für die Problemstellung der vorliegenden Arbeit darüber hinaus explizit berücksichtigt werden, dass die Allokation einer Sitzgelegenheit zu einem Passanten anderen Passanten eine Pause verwehren könnte, die diese ggf. dringender nötig gehabt hätten.

2.5.2.3 Parken als stochastische Prozesse

In den zuvor beschriebenen Ansätzen wurden keine Prognosen über die Parkdauer – ein Analogon für die Sitzdauer – gemacht, sondern lediglich bei Bedarf die momentane Verfügbarkeit von Parkplätzen abgerufen. Damit wird dort nicht das Problem betrachtet, abschätzen zu müssen, wie lange ein Fahrzeug parken wird. Mit sachgerechten Prognosen können Parkplätze aber so koordiniert werden, dass die Zielfunktionen besser erfüllt werden, als ohne Berücksichtigung, dass ein Parkplatz auch wieder frei wird. Ein Minimalbeispiel kann dies verdeutlichen.

Ein Fahrzeug befinde sich auf einer eindimensionalen Strecke mit der Länge 4000 m an der Streckenmarke 2000 m und fragt einen Parkplatz an. Es gebe lediglich zwei Parkplätze an den Streckenmarken 1000 m und 3000 m. Der gewünschte Zielort des Fahrzeuges sei genau bei 1000 m. Der Parkplatz bei der Streckenmarke 1000 m sei aber belegt. Aus diesem Grund fährt das Fahrzeug zum Parkplatz an der Streckenmarke 3000 m. Kurz bevor das Fahrzeug diesen Parkplatz erreicht, wird der andere Parkplatz bei der Streckenmarke 1000 m frei. Wenn dies vorher bekannt gewesen wäre, hätte das Fahrzeug auch direkt zum Parkplatz an der Streckenmarke 1000 m fahren können und hätte bei

Ankunft einen freien Parkplatz vorgefunden.

Yan et al. (2011, auch im Weiteren) entwickelten ein Modell, welches stochastische Prognosen über Parkplatzverfügbarkeiten enthält. Sie modellieren dazu Parken als sog. Geburts- und Todesprozesse, eine bestimmte Anwendung von Markov-Ketten. Die Geburt repräsentiert dabei das Ereignis, dass ein Fahrzeug auf einen Parkplatz fährt und diesen für eine nicht genau bekannte Zeitspanne belegt. Das Verlassen des Parkplatzes stellt das Todesereignis dar. Die Geburts- und Todesraten werden über Verkehrsdetektoren als stochastische Größen gemessen. Hierzu legen Yan et al. (ebd.) eine IT-Architektur zugrunde, die aus vernetzen Sensoren und Recheneinheiten besteht.

Die Recheneinheiten führen auf der Grundlage der durch die Sensoren erhobenen Größen Kalkulationen durch und dienen als Controller für eine mögliche aktive Verfügbarkeitssteuerung. Das Ziel des Controllers liegt in der Prognose des Erlöses aus der zeitweise gegen Entgelt zur Verfügungstellung von Parkplätzen. Dies soll dann als Basis für spezielle Parkplatzangebote oder zielgerichtete Rabatte dienen. Durch die Einführung zweier Parkplatztarife ergibt sich dabei ein zusätzliches Erlösmaximierungsproblem. Der sog. *Economy*-Tarif ist günstiger als der sog. *Business*-Tarif, ist aber von geringerer Qualität, z. B., weil die Parkplätze weiter weg vom Eingangstor einer Parkplatzanlage sind (denkbar wäre auch eine Diskriminierung in der Parkplatzgröße oder Bewachung). Der grundlegende stochastische Modellansatz ist wie folgt (ebd.):

Sei $X(t)$ die Anzahl der Parkplätze, die zum Zeitpunkt t belegt sind. Es sei $\{X(t); t \geq 0\}$ ein Geburts- und Todesprozess. Ausdruck 2.4, s. u., formuliert die Wahrscheinlichkeit dafür, dass zum Zeitpunkt $t + h, h > 0$, die Anzahl der belegten Parkplätze $i+1, i \in \mathbb{N}_0$ beträgt, unter der Bedingung, dass die Anzahl der belegten Parkplätze zum Zeitpunkt t i betrug. Damit wird die Übergangswahrscheinlichkeit dafür formuliert, dass im Zeitschritt h genau ein Parkplatz mehr belegt wird. λ ist dabei die Geburtenrate, gegeben durch die Schätzung der relativen Häufigkeit[22], mit der ein Auto auf einen Parkplatz fährt. Ausdruck 2.5 formuliert entsprechend, dass im Zeitschritt h genau ein Parkplatz frei wird. $i\mu$ ist die Sterberate für i parkende Autos, d. h. eine Schätzung der relativen Häufigkeit, mit der ein Auto einen Parkplatz verlässt, wenn i Autos parken. Yan et al. (ebd.) nehmen an, dass die Sterberate proportional zur Anzahl der parkenden Autos ist. $o(h)$ ist ein Restglied, das infinitesimal wird,

[22] Die Schätzung relativer Häufigkeiten bildet ein logisch-analytisches Wahrscheinlichkeitsmaß (Carnap, 1959, S. 45 ff.).

wenn h gegen unendlich geht.

$$P\Big\{X(t+h) = i+1 \Big| X(t) = i\Big\} = \lambda \cdot h + o(h) \ , \tag{2.4}$$

$$P\Big\{X(t+h) = i-1 \Big| X(t) = i\Big\} = i\mu \cdot h + o(h) \ . \tag{2.5}$$

Mithilfe dieses Ansatzes bestimmen Yan et al. (ebd.) die Wahrscheinlichkeit $p_i(t)$, dass die Anzahl parkender Autos zum Zeitpunkt t i beträgt (durch Lösen eines entsprechenden Differenzen-Differentialgleichungssystems). In weiteren Schritten diskriminieren Yan et al. (ebd.) die Geburtenrate für die Economy-Klasse mit λ_1 gegen die Geburtenrate in der Business-Klasse mit λ_2, die Sterberaten mit μ_1 und μ_2, die Parkplatzkapazitäten der beiden Buchungsklassen mit N_1 und N_2 sowie die Tarife mit f_1 und f_2. Yan et al. (ebd.) modellieren zwei stochastische Geburts- und Todesprozesse und bestimmen für jeden der vier folgenden Fälle den erwarteten Erlös: (1.) Die Economy-Klasse und die Business-Klasse haben freie Parkplätze, (2.) die Economy-Klasse ist vollständig belegt, die Business-Klasse aber nicht, (3.) die Business-Klasse ist vollständig belegt, die Economy-Klasse aber nicht, (4.) beide Klassen sind vollständig belegt. Auf dieser Basis können Sensitivitätsanalysen durchgeführt werden, welche Auswirkungen zielgerichtet induzierte Parameteränderungen auf den erwarteten Gesamterlös haben können.

Der stochastische Geburts- und Todesprozess-Ansatz ist insofern für die Problemstellung der vorliegenden Arbeit interessant, als dass Yan et al. (ebd.) eine Möglichkeit aufzeigen, wie unsichere Belegungszeiten öffentlicher Sitzgelegenheiten modelliert werden können, unabhängig davon, ob dann diskriminierte Buchungsklassen betrachtet werden. Der Ansatz liefert eine allgemein parametrisierte Berechnungsvorschrift, wie Belegungszeiten für Sitzgelegenheiten prognostiziert werden könnten. Obwohl eine dedizierte Optimierungsvorschrift fehlt, kann der Ansatz als Grundlage für die Bestimmung optimierter Parameterwerte dienen.

Menschliches Verhalten ist prinzipiell nicht immer deterministisch beschreib- und prognostizierbar, weshalb stochastische Modelle ein angemessenes Werkzeug bieten können. Die durch stochastische Modelle modellierte Unsicherheit kann allerdings auch durch sachgerechte Annahmen eliminiert werden. Dabei können Annahmen auch dann als sachgerecht betrachtet werden, wenn sie eine Diskursweltkomplexität derart reduzieren, dass sie idealtypische Aussagen ermöglichen. Die Aussagen sind stets unter der Bedingung der Annahmen zu interpretieren. Eine solche idealtypische Annahme im Smart-Parking-Beispiel wäre, dass jeder Fahrer genau auf dem Parkplatz parkt, zu dem er alloziert ist und dass kein anderer Fahrer dort parkt, oder dass die Parkdauern genau be-

kannt sind. Dies sind Annahmen, die zwar in der Realität nicht immer exakt zutreffen mögen, aber eine aussagekräftige Analyse prinzipieller Zusammenhänge ermöglichen.

Für die Analyse der Wirksamkeit eines steuernden Verfahrens, was auch Ziel der vorliegenden Arbeit ist, kann es vorteilhaft sein, zunächst enge Annahmen zu definieren, die dann nach und nach erweitert bzw. durch stochastische Modellierung gleichsam gelockert werden können. Es kann bspw. unter Annahmen davon ausgegangen werden, dass die Parkzeiträume der Fahrer exakt bekannt seien. So werden idealtypische Analysen ermöglicht, die aber bei sachgerechter Interpretation unter der Bedingung der Annahmen zu aussagekräftigen Aussagen führen können, z. B.: Unter der angenommenen Idealbedingungen, dass für zwei Autos die Parkzeiträume exakt bekannt seien, und sich diese Parkzeiträume überschneiden, kann geschlossen werden, dass ein Parkplatz nicht ausreicht, um die Nachfrage nach Parkplätzen zufriedenstellend zu befriedigen. Die entscheidende Frage ist dann, ob die angenommen Parkzeiträume sachgerecht sind. Eine analoge Frage stellt sich aber auch bei Ersetzen der Parkzeiträume durch Zufallsvariablen. Der Unterschied ist lediglich, dass Zufallsvariablen mehr anpassbare Parameter zulassen, etwa durch die Art der Verteilung, ggf. durch die Intervallgrenzen oder durch die „eigentlichen“ Parameter einer stochastischen Verteilung, wie Mittelwert und Varianz. Im Übrigen ist die Betrachtung von Mittel- bzw. Erwartungswerten ebenso idealisiert und bedarf einer reflektierten Interpretation. Eine solche idealisierte Interpretation könnte sein, dass für alle Fahrer der gleiche Parkzeitraum angenommen wird, gegeben durch den Mittelwert aller Parkzeitraumschätzungen. Der hier vorgestellte stochastische Ansatz soll als eine *mögliche* Modellerweiterung, aber nicht als grundlegender Ansatz für die vorliegende Arbeit verstanden werden.

2.5.2.4 Parkplatzallokation als Optimierungsproblem

Geng und Cassandras (2013, auch im Weiteren) formulieren für das Problem der Parkplatzallokation einen normativ-steuernden Ansatz mit einer expliziten Ziel- bzw. Optimierungsfunktion. In diesem Zusammenhang betonen sie die besondere Notwendigkeit für eine normative Steuerung der Parkplatzsuche, die für jeden Fahrer individuell ist. Denn einfache Parkleitsysteme, die allen Fahrern die gleichen Informationen darüber anzeigen, wo es noch freie Parkplätze gibt, können dazu führen, dass alle Parkplatzsuchende zum gleichen Parkplatz fahren. In der Folge kann es dann zu punktueller Verkehrsverstopfung kommen. Solch ein Parkleitsystem hätte dann negative Effekte auf den Verkehr.

Im mathematisch formulierten Ansatz von Geng und Cassandras (ebd.) ist

X eine Matrix mit $x_{ij} = 1$, gdw. Fahrer i zur Parkplatzressource j zugeordnet ist, sonst 0. Eine Parkplatzressource kann dabei ein einzelner Parkplatz oder eine Parkplatzanlage mit mehreren Einzelparkplätzen sein. k stellt einen sog. Entscheidungspunkt dar. Die Entscheidungspunkte sind die Zeitpunkte, zu denen jeweils eine Lösung für das Minimierungsproblem berechnet wird. $W(k)$ ist die Menge der Fahrer, die zum Entscheidungspunkt k in einer Warteliste für einen Parkplatz stehen. $R(k)$ ist die Menge der Fahrer, die zum Entscheidungspunkt k in einer Liste für reservierte Parkplätze eingetragen sind. $\Omega_i(k)$ ist die Menge der Parkplatzressourcen, die für den Fahrer i zum Entscheidungspunkt k möglich sind. J_{ij} sind die Kosten, die dem Fahrer i entstehen, wenn er zur Parkplatzressource j alloziert wird. Diese Kosten hängen auch vom Entscheidungspunkt k ab. Die Kostenfunktion J gewichtet monetäre Kosten, die von der Reservierungsdauer, Belegungsdauer und Fahrzeit zur Parkplatzressource abhängen, gegen die Distanz der Parkplatzressource zum eigentlichen gewünschten Bestimmungsort des Fahrers. Die Zielfunktion lautet:

$$\min_X \underbrace{\sum_{i \in W(k) \cup R(k)} \sum_{j \in \Omega_i(k)} x_{ij} \cdot J_{ij}(k)}_{\text{Parkkostenterm}} + \underbrace{\sum_{i \in W(k)} \left(1 - \sum_{j \in \Omega_i(k)} x_{ij}\right)}_{\text{Strafterm}} . \tag{2.6}$$

Der Parkkostenterm summiert gemäß der Kostenfunktion J für eine Parkplatzallokation X die Kosten aller Fahrer in der Warte- und in der Reservierungsliste, die zu einer Parkplatzressource alloziert wurden. Der Strafterm addiert für die nicht allozierten Fahrer konzeptionelle Strafkosten, sodass implizit möglichst viele Fahrer zu einer Parkplatzressource j alloziert werden. Wenn ein Fahrer i zu keiner Parkplatzressource j alloziert ist, dann ist $\sum_{j \in \Omega_i(k)} x_{ij} = 0$ und es werden folglich Strafkosten von 1 addiert. Da jeder Fahrer nur zu maximal einer Parkplatzressource j alloziert werden kann, können die Strafkosten nicht negativ werden. Außerdem können die Parkkosten für einen Fahrer per Konstruktion niemals die Strafkosten für die Nichtallokation des Fahrers übersteigen. Denn die Kostenfunktion für die Parkkosten ist durch

$$J_{ij}(k) := \lambda_i \cdot \frac{M_{ij}(k)}{M_i} + (1 - \lambda_i) \cdot \frac{D_{ij}}{D_i} \tag{2.7}$$

gegeben, wobei $\lambda_i \in [0; 1]$ ein Gewichtungsfaktor für den Fahrer i ist, $M_{ij}(k)$ die monetären Kosten sind, die zum Entscheidungspunkt k für Fahrer i anfallen, wenn er auf der Parkplatzressource j parkt, und D_{ij} ein Kostenmaß für die Distanz vom Parkplatz j zum eigentlich gewünschten Bestimmungsort von i ist. M_i und D_i sind jeweils maximale Kosten, um die Kosten zu normalisieren.

Die Kostenfunktion nimmt also Werte zwischen 0 und 1 an.

Für die durch den Ausdruck 2.6 gegebene Zielfunktion muss eine Lösung unter Berücksichtigung der folgenden Nebenbedingungen bestimmt werden, für die hier eine verbale Formulierung genügen soll: (1) Jeder Fahrer in der Warteliste darf *höchstens* zu einer Parkplatzressource alloziert werden. (2) Jeder Fahrer in der Reservierungsliste muss *genau* zu einer Parkplatzressource alloziert werden. (3) Keine Parkplatzressource darf überbelegt werden. (4) An keinem Entscheidungspunkt darf eine Allokation gemacht werden, die für einen Fahrer schlechter ist, als die Allokation beim vorherigen Entscheidungspunkt.

Die Art und Weise der mathematischen Formulierung legt stets einen exakt-optimierenden Lösungsansatz nahe. Dies ist allerdings nicht immer ohne Einschränkungen möglich. Für Optimierungsprobleme mit „echt"-kombinatorischer Komplexität – d. h. für Probleme, deren Anzahl zu überprüfender bzw. vergleichender Lösungen exponentiell oder sogar faktoriell mit der Anzahl kombinierbarer Elemente wächst – kann auch mit moderner IT ab einer gewissen Anzahl kombinierbarer Elemente keine exakte optimale Lösung in hinnehmbarer Zeit bestimmt werden. Auch Geng und Cassandras (2013) reduzieren daher die Komplexität ihres Parkplatzallokationsproblems durch Abgrenzen von Gebieten, Zusammenfassen von Parkplätzen und dadurch, dass Fahrer nicht in die Berechnung einer optimalen Allokation mit einbezogen werden, die noch weiter weg von einem Gebiet sind, als ein bestimmter Schwellwert vorgibt. Dennoch kommen nach dieser Reduktion exakt-optimierende Lösungsverfahren der linearen (gemischt mit diskreter) Optimierung zum Einsatz. Ein exakt-optimierendes Verfahren kann stets darin bestehen, alle Lösungsmöglichkeiten zu überprüfen und dann die gemäß Zielfunktion beste zu wählen.

Die vorliegende Arbeit wird keine dezidierten Verfahren der linearen Optimierung anwenden. Der Grund dafür liegt darin, dass sich nicht zwingend auf lineare Zielfunktionen und Nebenbedingungen beschränkt werden soll. Der Ansatz von Geng und Cassandras (ebd.) wäre dessen ungeachtet in vielen Details anzupassen und nicht direkt übernehmbar. Zum Bsp. soll in der vorliegenden Arbeit explizit berücksichtigt werden, wie lange Sitzgelegenheiten belegt sind, weil eine Sitzgelegenheit zu unterschiedlichen Zeitpunkten von verschiedenen Passanten genutzt werden kann, und dass ein Passant möglicherweise mehrere Sitzpausen braucht, d. h. zu mehreren Sitzgelegenheiten alloziert werden muss.

Die Arbeit von Geng und Cassandras (ebd.) bietet allerdings interessante Einsichten, die auch für die Problemstellung der vorliegenden Arbeit relevant sind. Der Ansatz, das Allokationsproblem zu bestimmten Entscheidungspunkten zu lösen, kann dafür übernommen werden, ein Lösungsverfahren in einer nicht-geschlossenen Diskurswelt anzuwenden. Damit ist insbesondere gemeint, dass neue, bei der Bestimmung einer Allokation nicht berücksichtige Passanten,

in die Diskurswelt eintreten können. Auch die Nebenbedingung, dass zu einem Entscheidungspunkt eine Allokation für keinen Passanten schlechter werden darf als die zuvor bestimmten Allokationen erscheint plausibel. Diese ist aber diskutabel, wie das folgendes Minimalbeispiel zeigt: Es sei ein Passant p_1 und eine Sitzgelegenheit S, auf der nur Platz für genau einen Passanten ist. Zum Entscheidungspunkt 1 sei p_1 zu S alloziert. Nun trete ein zweiter Passant p_2 zur Diskurswelt hinzu. Auch wenn p_2 eine Sitzpause viel dringender nötig hat, als p_1 dürfte unter der genannten Nebenbedingung p_2 nicht zu S, also zum einzigen Sitzplatz in der Diskurswelt, alloziert werden.

Eine weitere relevante Einsicht besteht in der von Geng und Cassandras (ebd.) beschriebenen Möglichkeit für Parkplatzgarantien. Sobald ein Parkplatz reserviert wurde, muss auch sichergestellt werden, dass dieser nur von dem Fahrer benutzt werden kann, für den dieser reserviert ist. Selbiges gilt für Sitzgelegenheiten. Geng und Cassandras (ebd.) schlagen als weiche Reservierungsmöglichkeit visuelle Hinweise vor, z. B. Lichtsingale, ähnlich wie Ampeln, oder textuell. Als harte Reservierungsmöglichkeit, bei der physisch verhindert werden kann, dass unbefugte Fahrer einen reservierten Parkplatz nutzen, schlagen Geng und Cassandras (ebd.) absenkbare Poller, kleine Schranken oder sonstige Barrieren vor. Beide Möglichkeiten wären auch für Sitzgelegenheiten einsetzbar.

2.6 Adaptivitätsanforderung auf Verbundebene: Wohlfahrtskriterien für den urbanen Raum

2.6.1 Nutzen und utilitaristische Wohlfahrt

Die Funktionalität eines SSOs als öffentliches Gut soll (gesamt-) gesellschaftlich legitimiert sein. Dies erfordert die Berücksichtigung und ggf. eine Abwägung unterschiedlicher Interessen, was in funktionale Software-Anforderungen überführt werden muss. Durch diese explizit durchzuführende Abwägung sollte kein Passant übervorteilt werden. In Bezug auf Adaptivität von SSOs mündet dies in der Frage in welcher Weise Adaptivität anhand individueller Bedürfnisse ausgerichtet werden „darf“. Denn die Maximierung der Bedarfserfüllung eines Individuums kann zulasten der Bedarfserfüllung eines anderen Individuums gehen. Die folgende Erörterung zeigt dies an einem Beispiel für die Verfügbarkeit geeigneter Sitzgelegenheiten.

Das Übervorteilungsverbot ist ein Gerechtigkeitsgebot. Gerechtigkeit kann allerdings unterschiedlich aufgefasst und auf unterschiedlichen Ebenen geltend gemacht werden. Eine Lösung kann kollektiv gerecht erscheinen, auch wenn sie

individuell ungerecht erscheint und v. v. Es gebe bspw. in einer Diskurswelt nur eine einzige Sitzgelegenheiten und drei Passanten. Alle drei Passanten benötigen eine Sitzpause an dieser Sitzgelegenheit. Passant p_1 benötigt 20 Minuten Regeneration, während die Passanten p_2 und p_3 nur jeweils 5 Minuten benötigen. p_1 erreicht die Sitzgelegenheit in 5 Minuten, p_2 in 10 und p_3 in 15 Minuten. Es kann als kollektiv ungerecht erscheinen, dass p_1 die Sitzgelegenheit für p_2 und p_3 blockiert. Auf der anderen Seite kann es als individuell ungerecht erscheinen, wenn p_1 die Sitzgelegenheit verwehrt wird.

Gerechtigkeitsaspekte für Allokationen als Koordinationslösungen sind in der Ökonomik schon lange Gegenstand formaler Modelle mit normativem Charakter[23]. Diese können formativ für die Formulierung funktionaler Software-Anforderungen herangezogen werden. Ausgangspunkt ist dabei das gedankliche Konstrukt des Nutzens. Nutzen stellt dabei im Grunde genommen nichts weiter als eine Quantifizierung einer Bedürfniserfüllung dar. In der Ökonomik wird Nutzen als ein numerisches Maß formalisiert, welches Individuen zu maximieren versuchen:

> Whether or not utility is some kind of glow or warmth, or happiness is here irrelevant; all that counts is that we can assign numbers to entities or conditions which a person can strive to realize. Then we say the individual seeks to maximize some function of those numbers. (Alchian, 1953, S. 31)

Eine aktuelle Arbeit der Ökonomik bezeichnet Nutzen auch als Indizes des Wohlbefindens (Piacquadio, 2017). In der selben Arbeit wird dafür argumentiert, dass die Quantifizierung von Nutzen durch eine *gemeinsame Kardinalisierungsfunktion* erfolgen kann. Die Argumentation beruht darauf, dass die Gesellschaft die gemeinsame Kardinalisierungsfunktion wählt, um konfligierende Interessen zu aggregieren und soziale Wohlfahrt zu messen (ebd.). Diese Funktion wird dann für alle betrachteten Individuen in gleicher Weise angewendet (ebd.). Genau in diesem Sinne wird im Kap. 3.2 eine gemeinsame Kardinalisierungsfunktion für die Problemstellung der vorliegenden Arbeit konstruiert werden.

Als geläufigste gemeinsame Kardinalisierungsfunktion kann eine direkte Quantifizierung durch Geld gelten. Die Zahlungsbereitschaft von x GE drückt dann einen Nutzen von x aus. Im vorliegenden Fall soll allerdings Geld gerade nicht an die Stelle einer Kardinalisierungsfunktion treten. Für städtebauliche Objekte als öffentliche Objekte sollen keine privilegierten Nutzungsrechte dadurch ableitbar sein, dass ein Passant über mehr Geld verfügt als ein anderer. Auch wenn es gute Gründe dafür geben mag, bestimmten Passanten aufgrund ihrer Beeinträchtigungen ein bevorzugtes Nutzungsrecht einzuräumen, so ist dieses

[23] Vgl. auch im Weiteren: Hubl (2018)

nicht durch Geld oder Zahlungsbereitschaften gerechtfertigt. Stattdessen kann die Gesellschaft ein gemeinsames akzeptiertes Maß für die Beeinträchtigungen legitimieren, das auf objektiv feststellbaren Merkmalen beruht. Dieses Maß kann dann als kardinale, interpersonal vergleichbare Nutzenfunktion für alle Passanten zugrunde gelegt werden.

Basierend auf der Nutzenkonzeption erarbeitete die Ökonomik einige Gerechtigkeitskriterien, die nun vor dem Hintergrund dieser Arbeit vorgestellt werden sollen. Denn nach Auffassung der vorliegenden Arbeit muss eine Smart City auch gerecht sein. Das heißt, die Indizes des Wohlbefindens, sollen in solch einer Weise über die Passanten verteilt sein, die als gerecht bezeichnet werden kann. Die grundlegendste Forderung für ökonomische Gerechtigkeit ist die nach Pareto-Effizienz. In einem Pareto-Optimum kann kein Individuum seinen Nutzen erhöhen, ohne gleichzeitig den Nutzen eines anderen zu reduzieren. Die folgenden Ausführungen werden allerdings zeigen, dass dies *keine hinreichende* Bedingung für eine Lösung darstellt, die als gerecht zu bezeichnen ist.

Zur Veranschaulichung diene folgendes Beispiel: Es gebe drei Sitzgelegenheiten mit jeweils einem Sitzplatz. Von einem definierten Nullpunkt aus, befinden sich diese in 10 m, 80 m und 100 m Entfernung. Es gebe zwei Passanten, die sich beide am definierten Nullpunkt befinden und gerne eine Sitzpause machen möchten. Der Nutzen beider Individuen sei durch die gemeinsame Kardinalisierungsfunktion $u(x) = \frac{10}{x}$ gegeben, wobei x die noch zurückzulegende Distanz bis zur Sitzgelegenheit ist. Demnach beträgt der Nutzen der ersten Sitzgelegenheit in $x = 10\,\text{m}$ Entfernung 1, der zweiten in $x = 80\,\text{m}$ Entfernung $0,125$ und der dritten in $x = 100\,\text{m}$ Entfernung $0,1$. Wenn der Passant p_1 nun zur ersten und der Passant p_2 zur zweiten Sitzgelegenheiten alloziert wird, dann kann sich keiner der beiden Passanten besser stellen, ohne den anderen schlechter zu stellen. p_2 müsste mit p_1 tauschen um sich besser zu stellen, wodurch p_1 allerdings schlechter gestellt würde. Diese Allokation ist also Pareto-effizient.

Ökonomen und v. a. Wohlfahrtsökonomen interessieren sich für aggregierte Indizes des Wohlbefindens. Aus diesem Grund besteht die Zielfunktion oftmals in der Maximierung der als soziale Wohlfahrt bezeichneten Summe der einzelnen erreichten Indizes. Um dieses Wohlfahrtskriterium schärfer von weiteren Kriterien für Wohlfahrt abzugrenzen, sei die Summe der individuell realisierten Nutzen hier als *utilitaristische Wohlfahrt* bezeichnet (denn alle Wohlfahrtskriterien könnten, je nach Interpretation, als „sozial“ attribuiert werden). Im oben beschriebenen Beispiel würde eine Allokation nach diesem Kriterium genau zur gleichen Allokation führen, also dass ein Passant zur ersten und der andere zur zweiten Sitzgelegenheit alloziert würde. Die utilitaristische Wohlfahrt betrüge dann $1,125$. Obwohl diese Zielfunktion den gesellschaftlichen Gesamtnutzen

maximiert, kann im Beispiel aus Gerechtigkeitsgesichtspunkten auch dagegen argumentiert werden: Möglicherweise wäre die Allokation gerechter, bei der ein Passant zur zweiten und der andere zur dritten alloziert wird, weil dann beide in etwa noch die gleiche Distanz zurückzulegen hätten.

Myerson (1979, 1984) analysierte eine andere Auffassung ökonomischer Gerechtigkeit. Wenn allen Individuen, die an einem Allokationsmechanismus teilnehmen, garantiert werden kann, dass jeder die gleiche Chancen hat, seinen maximalen Nutzen zu realisieren, dann ist der Mechanismus gerecht. Diese Idee ist auch als Zufallsdiktatur bekannt, weil sie gedanklich durch die Vorstellung ersetzt werden kann, dass ein Diktator zufällig eine Allokation wählt. Die Problematik besteht dann darin, jedem Individuum die gleichen Chancen zu garantieren, ohne dass einer die anderen durch Falschangabe seiner Präferenzen übervorteilen kann. Unter der Maxime einer gemeinsamen Kardinalisierungsfunktion entfällt diese Problematik freilich, weil die gemeinsame Kardinalisierungsfunktion die Präferenzen für alle Individuen gleich quantifiziert, welche somit als wohlbekannt vorausgesetzt werden können. Im o. g. Beispiel könnte der Diktator eine Münze werfen und abhängig vom Ausgang des Münzwurfs einen Passanten zur ersten und den anderen zur zweiten Sitzgelegenheit allozieren. Hier wird deutlich, dass dieser Ansatz zwar ex ante als gerecht bezeichnet werden kann, aber nach Ausführung des Allokationsmechanismus als genauso gerecht oder ungerecht bezeichnet werden kann, wie die oben beschriebenen Lösungen.

2.6.2 Neidfreiheit als Wohlfahrtsziel

Eine im Hinblick auf Ex-post-Gerechtigkeit brauchbarere Formulierung der Ökonomik stammt von Varian (1974). Wenn ein Passant p_1 das präferiert, was ein anderer Passant p_2 erhält, dann beneidet p_1 p_2. Eine gerechte Allokation ist dann eine, die neidfrei und Pareto-effizient ist. Nach Varian (ebd.) wird dies wie folgt formalisiert:

Sei eine Allokation $\mathbf{x}$ ein Vektor aus Sitzplätzen, dessen Dimensionalität genau der Anzahl der Passanten entspricht. Das i-te Element im Vektor enthält den Sitzplatz, der zu Passant p_i alloziert ist. Sei also x_i der Sitzplatz, der unter der Allokation $\mathbf{x}$ zu Passant p_i alloziert ist. Dann sei $u_i(x_i)$ der Nutzen, den Passant p_i durch die Allokation zu x_i erfährt. Wenn $\mathbf{y}$ eine andere Allokation ist, bei der Passant p_i zu y_i alloziert ist, dann bedeutet $x_i \succeq_i y_i$, dass $u_i(x_i) \geq u_i(y_i)$. Das heißt, dass p_i gemäß der Kardinalisierungsfunktion u die Sitzgelegenheit x_i gegenüber der Sitzgelegenheit y_i bevorzugt oder indifferent ist. In diesem Fall ist die Kardinalisierungsfunktion durch i parametrisiert, also im Grunde genommen individualisiert. Es kann sich dennoch um eine *ge-*

meinsame Kardinalisierungsfunktion handeln, wenn die Parametrisierung mit i ebenfalls auf gesellschaftlichem Konsens beruht und eben nicht indiziert, dass es sich um *private* Nutzenfunktionen handelt. Private Nutzenfunktionen sind solche, die außer dem betreffenden Individuum selbst niemandem bekannt sind. Durch eine Parametrisierung einer gemeinsamen Kardinalisierungsfunktion könnten demnach individuelle Informationen, wie Alter, Art der Beeinträchtigungen usw. mitberücksichtigt werden, ohne dass die Nutzenfunktion selbst privat ist.

$x_i \succeq_i x_j$ bedeutet, dass $u_i(x_i) \geq u_i(x_j)$ und d. h., dass Passant p_i bei der Allokation $\mathbf{x}$ den Sitzplatz, zu dem er alloziert ist, dem Sitzplatz vorzieht (oder ihm gegenüber indifferent ist), zu dem ein anderer Passant p_j alloziert ist. Eine Allokation gelte nun als gerecht[24], gdw.:

$$\nexists \mathbf{y} : (y_i \succeq_i x_i, \forall i \wedge \exists j : y_j \succ_j x_j) \text{ und} \tag{2.8}$$

$$x_i \succeq_i x_j, \forall i, j \ . \tag{2.9}$$

Ausdruck 2.8 formuliert, dass es keine Allokation gibt, bei der ein Passant bessergestellt wäre, ohne einen anderen Passanten schlechter stellen zu müssen. Unter dieser Bedingung ist $\mathbf{x}$ Pareto-effizient. Ausdruck 2.9 formuliert, dass unter der Allokation $\mathbf{x}$ kein Passant p_i einen anderen Passanten p_j um seinen Sitzplatz beneidet.

Im oben beschriebenen Beispiel existiert keine Allokation, die beide Bedingungen erfüllt. Wenn es aber eine zusätzliche Sitzgelegenheit in 100 m Entfernung gäbe, dann existierte zumindest die neidfreie Allokation, bei der beide Passanten zu einer Sitzgelegenheit in 100 m Entfernung alloziert werden. Diese Allokation wäre allerdings dennoch nicht Pareto-effizient. Es könnten sich sogar beide Passanten besser stellen, ohne den anderen schlechter zu stellen. Das heißt auch, dass die Allokation beider Passanten zu einer Sitzgelegenheit in 100 m Entfernung nicht stabil wäre. Das wiederum heißt, dass es keine Garantie für eine im Sinne von Varian (ebd.) gerechte Allokation gibt.

2.6.3 Wohlfahrtsegalitarismus

Einem Gerechtigkeitskriterium kann sich aber auch auf eine etwas andere Weise genähert werden. Dazu sei erneut der Ausdruck 2.9 zu Neidfreiheit betrachtet, aber mit der Kardinalisierungs- bzw. Nutzenfunktion formuliert: $u_i(x_i) \geq u_i(x_j), \forall i, j$. Nun wird dieser Ausdruck so abgeändert, dass Passant

[24] Varian (1974) würde von „*stark* gerecht“ sprechen. Für „schwach gerecht“ müsste nur gelten: $\nexists \mathbf{y} : y_i \succ_i x_i, \forall i$ und $x_i \succeq_i x_j, \forall i, j$.

p_i den anderen Passanten p_j nicht um den Sitzplatz beneiden soll, den er erhält, sondern um den Nutzen, den er dadurch erfährt. Dieses Kriterium ist sowohl befürwortend als auch ablehnend diskutierbar. Unter dem Primat eines Interessensausgleichs ist es allerdings begründbar. Durch die oben vorgeschlagene Abänderung des mit der Kardinalisierungsfunktion formulierten Ausdrucks 2.9 wird dieser zu:

$$u_i(x_i) \geq u_j(x_j), \forall i, j \tag{2.10}$$

$$\Leftrightarrow \; u_i(x_i) = u_j(x_j), \forall i, j \; . \tag{2.11}$$

Im Ausdruck 2.10 wurde auf der rechten Seite $u_i(\cdot)$ durch $u_j(\cdot)$ ersetzt, sodass die Nutzen der Passanten p_i und p_j unter der Allokation $\mathbf{x}$ miteinander verglichen werden. Dies ist äquivalent zum Ausdruck 2.11, der auch als eine Forderung nach Wohlfahrts*egalitarismus* bezeichnet werden kann (Thomson, 1983; Ray und Ueda, 1996; Yengin, 2012). Der Unterschied zum Gerechtigkeitskriterium von Varian (1974) wird am folgenden, leicht abgeänderten Beispiel deutlich:

Der Nutzen für Passant p_1 sei durch $u_1(x) = \frac{10}{x+20}$ bestimmt, wobei x hier wieder eine Distanz darstellt, während der von p_2 weiterhin durch $u_2(x) = \frac{10}{x}$ bestimmt sei. Das heißt p_2 kann im Prinzip $20\,m$ weiter gehen als p_1. Die Nutzenfunktionen u_1 und u_2 sind also individualisiert, können aber dennoch eine gesellschaftlich legitimierte, gemeinsame Kardinalisierungsfunktion darstellen. Zum Bsp. kann die Addition von $+20$ im Nenner der Nutzenfunktion von p_1 aufgrund des Alters oder aufgrund bestimmter medizinischer Indikationen legitimiert und gemeinschaftlich akzeptiert sein. Wenn beide Passanten auf eine Sitzgelegenheit in 100 m alloziert werden dann beneidet p_1 p_2 nicht um den Sitzplatz, es gilt: $u_1(x_1) = u_1(x_2)$. Es gilt aber $u_1(x_1) < u_2(x_2)$, d. h. p_1 ist neidisch in Bezug auf den Nutzen von p_2. Wenn allerdings p_1 zur Sitzgelegenheit bei 80 m alloziert wird, dann gilt $u_1(x_1) = u_2(x_2)$ und es besteht ein Interessensausgleich in Bezug auf den Nutzen.

Allerdings würde p_2 unter dieser egalitaristischen Allokation p_1 nun um die Sitzgelegenheit bei 80 m beneiden. Es gilt dann $x_2 \not\succeq_2 x_1$ bzw. $x_2 \prec_2 x_1$. Abgesehen davon, wäre diese Beispielallokation auch nicht Pareto-effizient. Im Allgemeinen gibt es für die Existenz einer egalitaristischen Lösung keine Garantie. Im obigen Beispiel wird dies auch ohne die Pareto-Effizienzforderung klar, wenn die Sitzgelegenheit bei 80 m nicht dort, sondern z. B. bei 70 m steht.

2.6.4 Das Differenzprinzip

Die diskutierten Kriterien liefern keine Garantie für eine zufriedenstellend als gerecht zu bezeichnende Lösung. Allerdings formuliert Rawls (1999) aus einer übergeordneten gesellschaftlichen Perspektive Prinzipien für Gerechtigkeit, aus denen sich ein auf dem ökonomischen Nutzenkonzept basierendes Kriterium für die Problemstellung der vorliegenden Arbeit ableiten lässt. Ausgangspunkt ist dabei die These, dass in einer Gesellschaft Ungleichheit zulässig ist, wenn diese zum Vorteil aller ist (ebd., S. 54). Eine beispielhafte Ausprägung dieser Auffassung *kann* das ökonomische Prinzip der Arbeitsteilung sein. Rawls (ebd.) formuliert den folgenden allgemeinen gesellschaftlichen Grundsatz:

> All social values—liberty and opportunity, income and wealth, and the social bases of self-respect—are to be distributed equally unless an unequal distribution of any, or all, of these values is to everyone's advantage. (ebd.)

Im Umkehrschluss formuliert Rawls (ebd.) Ungerechtigkeit als Ungleichheit, die nicht zum Vorteil aller ist. Im o. g. Zitat ist lediglich implizit ausgedrückt, ob Gleichheit, die nicht zum Vorteil aller ist, durch Ungleichheit zu ersetzen ist, wenn diese zum Vorteil aller wäre. In seiner unten zitierten Spezifizierung des o. g. Grundsatzes nimmt Rawls (ebd., S. 53) diesbezüglich eine Teilung in zwei Prinzipien vor.

> First: each person is to have an equal right to the most extensive scheme of equal basic liberties compatible with a similar scheme of liberties for others.
> Second: social and economic inequalities are to be arranged so that they are both (a) reasonably expected to be to everyone's advantage, and (b) attached to positions and offices open to all. (ebd.)

Das erste Prinzip hat vor dem zweiten Vorrang und betrifft die Grundfreiheiten von Personen, wie das Wahlrecht, das Recht öffentliche Ämter zu bekleiden, Meinungs- und Versammlungsfreiheit, Gewissens- und Gedankenfreiheit, Schutz vor psychischer Unterdrückung und der körperlichen Unversehrtheit, Recht auf Eigentum, Schutz vor willkürlicher Gefangennahme und Enteignung (ebd.). Dies sowie der (b)-Teil des zweiten Prinzips haben eher staatsphilosophische Tragweite und sollen für die Problemstellung der vorliegenden Arbeit nicht weiter erörtert werden.

Von mittelbarer Bedeutung ist allerdings die im (a)-Teil des zweiten Prinzips formulierte Forderung, bestehende Ungleichheit so zu arrangieren, dass sie vernünftig erwartbar zu jedermanns Vorteil gereicht. Hieran knüpft sich direkt die Frage an, wie ein solches Arrangement *i. Allg.* möglich sein soll. Rawls (ebd., S. 72) beantwortet diese Frage wie folgt:

> [I]n a basic structure with n relevant representatives, first maximize the welfare of the worst off representative man; second, for equal welfare of the worst-off representative,

> maximize the welfare of the second worst-off representative man, and so on until the last case which is, for equal welfare of all the preceding n−1 representatives, maximize the welfare of the best-off representative man. (Rawls, 1999, S. 72)

Dadurch wird der (a)-Teil des zweiten Prinzips zur Forderung, dass das angestrebte Arrangement „(a) to the greatest benifit of the least advantaged“ (ebd.) sein soll. Diese Forderung kann hinterfragt werden: Denn eine mögliche Folge davon ist, dass auf Besserstellung bzw. Förderung von Eliten zugunsten von Schlechtergestellten verzichtet wird. Dies kann in mancher Hinsicht fragwürdig erscheinen, z. B. in sportlicher, aber v. a. auch in ökonomischer Hinsicht. Gerade hier könnte ja gerade auch die Förderung besonders leistungsstarker Repräsentanten letztlich zu jedermanns Vorteil gereichen: ein erfolgreicher Unternehmer kann möglicherweise sehr effektiv sozialen Wohlstand schaffen. Auch in grundsätzlicherer, die Menschenrechte betreffender Hinsicht, kann die Maxime hinterfragt werden. Ist es etwa unangebracht Emanzipation in einem Land voranzutreiben, nur weil es in anderen Ländern diesbezüglich noch Defizite gibt? Ist es bspw. nicht legitim in einem Land wie Deutschland, mehr Rechte für Frauen zu fordern, nur weil sie effektiv mehr Rechte genießen als Frauen in (manchen) anderen Ländern? Solche grundsätzlichen Fragen betreffen auch die Verortung der vorliegenden Arbeit: Ist es generationen*gerecht*, wenn die Zielkriterien für die Konstruktion auf spezielle Anforderungen von Senioren ausgerichtet sind? Die vorliegende Arbeit umschifft diese möglicherweise kontroverse Frage dadurch, dass im Rahmen der technischen Konzeption eine *bedarfs*gerechte Lösung angestrebt wird.

Genau hierzu ist ein technisch operationalisierbares Kriterium notwendig. Wenn dieses als gerecht betrachtet werden soll, dann lässt sich aus dem o. g. Prinzip und der darauf folgenden Forderung, wie weiter oben in Aussicht gestellt, ein auf dem Nutzenkalkül basierendes Kriterium formulieren, das lautet: *Maximiere den minimal realisierbaren Nutzen.* Rawls (ebd.) weist darauf hin, dass diese Operationalisierung in der Ökonomik als *Maximin*-Regel bekannt ist. Allerdings zieht es Rawls (ebd.) in Bezug auf die genannte Forderung vor, vom sog. *Differenzprinzip* zu sprechen. Denn die Maximin-Regel stellt in der Ökonomik üblicherweise ein Kriterium für Entscheidungen unter Unsicherheit dar. Das Differenzprinzip ist ein ausdrückliches Kriterium für Gerechtigkeit, das nicht mit einem Kriterium für Entscheidungen unter Unsicherheit verwechselt werden soll (ebd.). Die Anwendung der Maximin-Regel für *Entscheidungen unter Unsicherheit* kann z. B. mit der Annahme einer hohen Risikoaversion begründet sein, was aber für die Überlegungen von Rawls (ebd., S. 73) keine tragende Rolle spielt. In diesem Zusammenhang sei ausdrücklich darauf hingewiesen, dass sich das Maximin-Kriterium in der vorliegenden Arbeit ebenfalls

nicht auf Entscheidungen unter Unsicherheit bezieht.

Im oben auf S. 56 beschriebenen leicht abgeänderten Beispiel folgt aus der Anwendung der Maximin-Regel, dass p_1 zur Sitzgelegenheit bei 10 m und p_2 zur Sitzgelegenheit bei 80 m zu allozieren ist. Der minimal realisierte Nutzen beträgt dann 0,125. Würde p_2 zur Sitzgelegenheit bei 10 m und p_1 zur Sitzgelegenheit bei 80 m alloziert, dann wäre zwar die utilitaristische soziale Wohlfahrt (Summe der jeweils realisierten Nutzenwerte) höher, der minimal realisierte Nutzen betrüge aber nur 0,1. Die Minimax-Regel hat gleichsam eine Schutzwirkung vor extrem unvorteilhaften Allokationen für Einzelne. Rawls (ebd., S. 154) schreibt dazu:

> In this respect the two principles of justice have a definite advantage. Not only do the parties protect their basic rights but they insure themselves against the worst eventualities. (ebd.)

2.6.5 Schlussfolgerungen

Tabelle 2.2 zeigt für das leicht abgeänderte Beispiel von S. 56 alle möglichen Allokationen mit den entsprechenden Ausprägungen der diskutierten Kriterien. Die Anwendung des Maximin-Kriteriums hat die als vorteilhaft erachtete Eigenschaft, dass garantiert kein Individuum einen geringeren Nutzen erfährt, als der, welcher maximiert wird. Das Maximin-Kriterium führt dabei stets zu Pareto-optimalen Lösungen, denn für jede andere Lösung als (eine) der Maximin-Lösung(en) ist mindestens ein Individuum schlechter gestellt als in der Maximin-Lösung.

Postulat 1 *Sei* $\mathbf{x}$ *eine Allokation, wobei das* i*-te Element* $x_i, i \in \{1, \ldots, n\}$*, das repräsentiert, was zu Passant* p_i *alloziert ist. Sei* $u_i(x_i)$ *der Nutzen, den* p_i *bei der Allokation* $\mathbf{x}$ *erfährt. Sei* $\mathbf{X}$ *die Menge der möglichen Allokationen. Dann wähle die Allokation* $\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathbf{X}} \left(\min \left(u_1(x_1), \ldots, u_n(x_n) \right) \right)$.

2.7 Verfahren für Verbundintelligenz

2.7.1 Koordination durch Suche

Das auf Objektverbundebene zugrunde liegende Koordinationsproblem, geeignete Sitzgelegenheiten zu allozieren, erfordert die Auswahl einer Alternative aus mehreren möglichen Allokationen und damit ein Entscheidungsverfahren. Wooldridge (vgl. 2009, S. 65 ff., auch im Weiteren) beschreibt praktisches Schlussfolgern als Verfahren, um Entscheidungen zu treffen. Dabei un-

$p_1 \mapsto$ Sitzgelegenheit bei $x =$	10 m	10 m	80 m	80 m	100 m	100 m
$p_2 \mapsto$ Sitzgelegenheit bei $x =$	80 m	100 m	10 m	100 m	10 m	80 m
$u_1(x) \left(:= \frac{10}{x+20}\right)$	0,3	0,3	0,1	0,1	0,083	0,083
$u_2(x) \left(:= \frac{10}{x}\right)$	0,125	0,1	1	0,1	1	0,125
$\min(u_1, u_2)$	0,125*	0,1	0,1	0,1	0,083	0,083
$u_1 + u_2$	0,425	0,4	1,1*	0,2	1,083	0,208
Pareto-effizient?	Ja	Nein	Ja	Nein	Nein	Nein
Neidfrei?	Nein	Nein	Nein	Nein	Nein	Nein
Egalitaristisch?	Nein	Nein	Nein	Ja	Nein	Nein

Hinweis: u_i gehört zu p_i. *) Maximalwert unter den Allokationsalternativen.

Tabelle 2.2: Mögliche Allokationen für ein Beispiel mit drei Sitzgelegenheiten (je eine bei 10 m, 80 m, 100 m)

terscheidet er Delibaration und Mittel-Zweck-Schlussfolgern. Durch Deliberation sollen die angestrebten Ziele festgelegt werden. Im vorliegenden Fall werden diese durch eine auf Wohlfahrtskriterien basierende Zielfunktion gegeben, bei welcher der minimale Nutzen maximiert werden soll. Durch Mittel-Zweck-Schlussfolgern soll bestimmt werden, *wie* dieses Ziel erreicht werden kann. Diese Wie-Frage stellt sich. In der vorliegenden Arbeit sei Mittel-Zweck-Schlussfolgern als ein Verfahren für die *Suche einer vernünftigen Lösung* aufgefasst.

Mit dieser Auffassung werden zwei wesentliche Kernaspekte deutlich: (1.) Es geht um *Suchen* einer Lösung, d. h. die gewünschte Lösung kann nicht direkt durch eine Berechnungsvorschrift im Sinne einer geschlossenen Formel bestimmt werden. (2.) Es geht um *vernünftige Lösungen*, was hier wie folgt zu verstehen ist: Gemäß einer – in der Ökonomik regelmäßig zugrunde gelegten – Zweckrationalität wird eine Lösung angestrebt, welche ein gegebenes Bewertungskriterium maximiert (bzw. optimiert). Hierbei ist allerdings der Suchaufwand zum Finden der Lösung mit ins Kalkül zu ziehen. Denn es kann als unvernünftig betrachtet werden, wenn Aufwand und Resultat unverhältnismäßig zueinander stehen, bspw. wenn trotz einer erheblichen Erhöhung des Suchaufwands nur eine geringfügig bessere Lösung gefunden wird. Nach dieser Auffassung ist eine vernünftige Lösung also nicht notwendigerweise die Lösung unter allen existierenden Lösungen, für die das Bewertungskriterium maximal ist. Dies ist auch für die Problemstellung der vorliegenden Arbeit der Fall,

wenn für die Bestimmung der optimalen Lösung der Suchaufwand ins schier Unermessliche steigt. Eine Lösung gelte dann als vernünftig, wenn sie mit gegebenem Suchaufwand erwartbar möglichst nahe an die ein Bewertungskriterium maximierende Lösung heranreicht.

Gegeben ein Problem, das in einer Diskurswelt zu lösen ist, besteht die allgemeine Strategie in der vorliegenden Auffassung darin, relevante Alternativen auszuloten. Die Diskurswelt bildet einen scharf umrissenen Realweltausschnitt oder eine gedankliche Welt. Mit Ausloten relevanter Alternativen ist hier gemeint, dass systematisch Repräsentationen möglicher Diskursweltzustände erzeugt und bewertet werden. Dies stellt eine Art und Weise systematischen Nachdenkens dar. Die KI-Forschung befasste sich früh mit der Frage, wie solches systematisches Nachdenken mit Computern nachgebildet werden kann. Dabei begründete sie das hier zugrunde liegende Paradigma, Intelligenz als Fähigkeit systematischen Nachdenkens durch Suchverfahren in einem Zustandsraum[25] nachzubilden und entwickelte dazu entsprechende informationstechnisch operationalisierbare Verfahrensweisen (Russell und Norvig, 2003, S. 59 ff.; Luger, 2009, S. 35 ff.).

Ausgehend von einer Formalisierung für die betreffenden Diskursweltzustände, besteht der allgemeine Ansatz darin, für jeden Zustand eine Menge logischer Folgezustände zu erzeugen. Hieraus ergeben sich Baumstrukturen, wie beispielhaft in Abbildung 2.6 dargestellt (auf die im nächsten und übernächsten Absatz noch näher eingegangen wird). Der Baumstruktur-basierte Ansatz ist dabei von grundlegender Bedeutung für die KI und wird – auch gegenwärtig – für zahlreiche KI-typische Probleme verwendet (Agostinelli et al., 2019; Silver et al., 2016; Helmert et al., 2014; Browne et al., 2012; Laborie, 2003; Haralick und Shapiro, 1979, 1980). Die herausragende Bedeutung der baumartigen Strukturierung für das Lösen von Problemen geht sicherlich auch auf die natürliche Nähe zur Art und Weise zurück, wie Menschen das Lösen von Problemen mit systematischem Nachdenken angehen. Auch elementare logische Probleme werden regelmäßig mithilfe einer Baumstruktur gelöst (Luger, 2009, S. 107 ff.), was die Gültigkeit dieses Ansatzes zur Nachbildung systematischen, streng logischen Nachdenkens unterstreicht.

Die zentralen Komponenten des soeben vorgestellten Paradigmas sind ein (1) Modell der Diskurswelt für die formale Repräsentation von Diskursweltzuständen, eine (2) Expansionslogik für die Erzeugung logischer Folgezustände sowie eine (3) Bewertungsfunktion für die Auswahl einer Lösung. Im Folgenden soll das Paradigma mithilfe der Abbildung 2.6 noch etwas näher erläutert wer-

[25] Der Lösungsraum ist nur eine Teilmenge des Zustandsraum, weil nicht alle durchsuchten Zustände Lösungen sind.

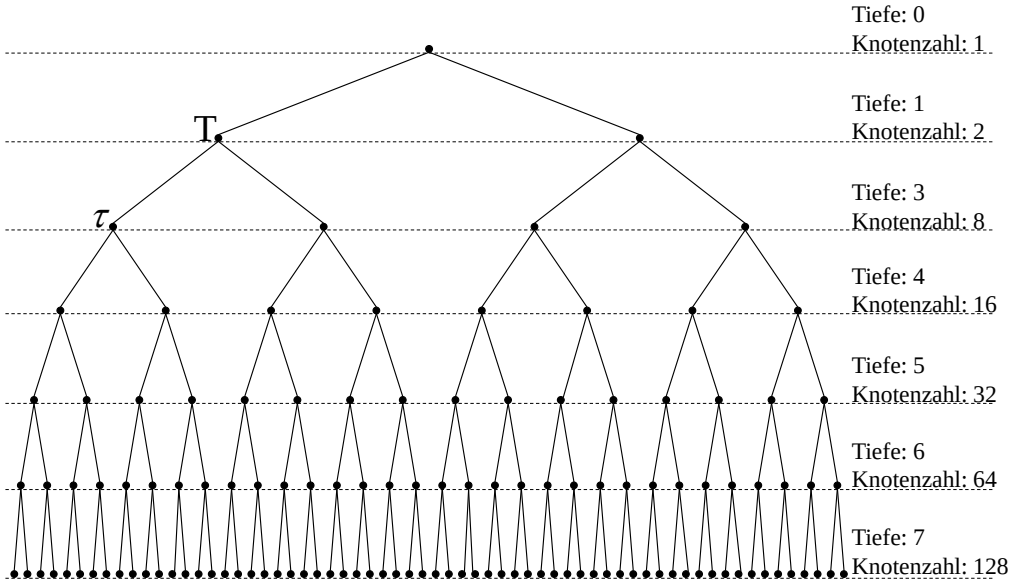


Abbildung 2.6: Struktur für Problemlösen als Suche im Zustandsraum

den: Die Kreise stellen Knoten dar, die jeweils einen Diskursweltzustand repräsentieren. Die Verbindungen zwischen Knoten verknüpfen logische Vorgänger- und Nachfolgerknoten miteinander. Der Nachfolgerknoten hängt dabei unter dem Vorgängenknoten; τ ist bspw. ein Nachfolgerknoten von T bzw. T ist ein Vorgängerknoten von τ. Der ganz obere Knoten ohne Vorgänger stellt einen Startpunkt für das nachgebildete systematische Nachdenken dar. Der Baum-Metapher für solche Strukturen entsprechend werden der Startknoten auch als Wurzel und die Knoten ohne Nachfolger als Blätter bezeichnet.

In einer solchen Baumstruktur stellen Knoten i. d. R. prinzipielle Verzweigungspunkte dar, an denen eine Entscheidung zu treffen ist und sich die Diskurswelt dementsprechend unterschiedlich entwickeln kann. Die Nachfolger eines Knotens werden durch eine sog. *Expansionslogik* erstellt. Die Tiefe, auf der ein Knoten liegt, gibt an, über wie viele Expansionen dieser Knoten ausgehend von der Wurzel erreicht wurde. In der Abb. 2.6 liegen alle Blätter auf der Tiefe 7. Im Allgemeinen müssen aber nicht alle Blätter auf der gleichen Tiefe liegen.

In einer Baumstruktur existiert zu jedem Knoten τ genau ein Pfad, d. h. eine Folge von Knoten, über die τ ausgehend von der Wurzel erreicht wird. Vorausgesetzt die Knoten repräsentieren paarweise verschiedene Diskursweltzustän-

de, dann stellt jeder Knoten eine dedizierte Entwicklung der Diskurswelt dar.[26] Wenn ein Blatt die Bedingungen für einen Zielzustand erfüllt, dann ist durch den Pfad ausgehend von der Wurzel gegeben, welche Entscheidungen zu treffen sind, um diesen Zielzustand zu erreichen, d. h. *wie* der Zielzustand hergestellt werden kann. Die Zielzustände können mittels einer *Bewertungsfunktion* anhand bestimmter Kriterien bewertet werden, um eine Auswahl unter mehreren möglichen Zielzuständen zu treffen. Auch wenn alle Zielzustände über die selbe Menge erfüllter Bedingungen als Zielzustand bestimmt sind, unterscheiden sie sich i. d. R. paarweise aufgrund des Pfades, über den sie erreicht wurden. (In der Regel enthalten die Zustandsbeschreibungen mehr Informationen als die Bedingungen, die für einen Zielzustand erfüllt sein müssen.) Somit kann mit der Bewertung eines Zielzustandes auch der Pfad dorthin bewertet werden.

In der Abbildung 2.6 haben alle Knoten, die keine Blätter sind, genau zwei Nachfolger. Damit verdoppelt sich die Zahl der Knoten auf jeder Tiefe, was schnell die intellektuelle Handhabbarkeit übersteigen kann. Mit der maximalen Zahl der Nachfolger eines Knoten als sog. Verzweigungsgrad kann die Zahl der Knoten auf einer Tiefe, d. h. die Zahl zu erzeugender und zu bewertender Pfade, grundsätzlich bis zu der Zahl anwachsen, die sich ergibt, wenn der Verzweigungsgrad mit der Tiefe potenziert wird. Damit liegt in Abhängigkeit der Tiefe eine exponentielle Komplexität vor. Dies kann auch eine hinnehmbare maschinelle Bewältigbarkeit übersteigen. (Im Abschnitt 3.3.2.1 wird das an einem Beispiel für die vorliegende Problemstellung dargelegt werden.) Dadurch kann es zweckmäßig werden, gleichsam eine weitere Ebene der Intelligenz in die Suchverfahren einzuziehen, um sicherzustellen, dass eine im oben beschriebenen Sinne vernünftige Lösung gesucht bzw. gefunden wird.

2.7.2 Die Bedeutung von Heuristiken

Diese geforderte Ebene von Intelligenz kann durch die Implementierung von Heuristiken eingezogen werden. Laut Luger (2009, S. 123) geht der Heuristik-Begriff auf das griechische Wort für „finden“ i. S. v. „entdecken“ zurück. Dabei sind Heuristiken dadurch gekennzeichnet, dass die Lösungen, die sie finden, nicht zwingend exakt der eigentlich gesuchten Lösung entsprechen, d. h. z. B. nur sub-optimal sind oder Näherungswerte darstellen. Heuristiken können aber dennoch dazu beitragen, wie im vorherigen Abschnitt 2.7.1 gefordert, vernünftige Lösungen zu finden, wenn sich die Abweichung in Grenzen hält und deshalb zustande kommt, damit das Suchverfahren hinnehmbar oder überhaupt erst möglich wird.

[26] Für die vorliegende Arbeit kann dies vorausgesetzt werden.

Heuristiken sind v. a. für zwei Problemklassen sogar erforderlich (Luger, 2009, S. 123 f.):[27] zum einen, wenn das zu lösenden Problem nicht vollständig formalisiert ist, wie bspw. bei Bilderkennung oder auch bei medizinischer Diagnostik; zum anderen, wenn die Berechnung einer exakten Lösung zwar möglich, aber nicht hinnehmbar ist, weil dies zu lange dauert oder zu viel Speicherplatz benötigt. Der aufmerksame Leser wird bereits ahnen, dass die vorliegende Problemstellung nicht in die erste der beiden genannten Problemklassen fällt. Ob sie in die zweite Problemklasse fällt, hängt von der genauen Problembeschreibung ab. Denn je nach Formulierung kann auch bei prinzipiell exponentiell vielen Zielzuständen, hinnehmbar eine exakte Lösung gefunden werden, etwa wenn im vorliegenden Bsp. das Ziel darin besteht, die Lösungen zu finden, welche die Zeit minimiert, die Passanten auf der definierten Strecke unterwegs sind. Bei keinen weiteren Nebenbedingungen ist die Lösung unter der Annahme konstanter Gehgeschwindigkeiten dann jene, bei der kein Passant eine Sitzpause auf einer adaptiven Sitzgelegenheit macht, und kann direkt bestimmt werden.

Es sei darauf hingewiesen, dass in anderen, vorrangig sozialwissenschaftlichen Fachdisziplinen, der Heuristikbegriff stark mit Komplexitätsreduktion durch spontane Eingebungen bzw. Intuition verbunden ist, d. h. mit einem „unmittelbare[n], nicht diskursive[n], nicht auf Reflexion beruhende[n] Erkennen, Erfassen eines Sachverhalts oder eines komplizierten Vorgangs“ (Duden online, 2020). Obwohl dies sinnbildlich durchaus naheliegend ist, ist dies hier irreführend. Denn Intuition wird gemeinhin als eine nicht-explizierbare Handlungsstrategie begriffen, die auf spontaner Eingebung ohne Nachdenken beruht. Heuristische Verfahren in der vorliegenden Auffassung können aber auch auf einer Durchsuchung des Lösungsraums beruhen, die insb. explizierbar ist und für welche das Bild einer spontanen Eingebung folglich unpassend wäre. Auch eine heuristische Lösung kann auf der Eruierung sehr vieler möglicher Lösungen beruhen.

Für den Heuristikbegriff kann i. Allg. keine Forderung an eine Treffgenauigkeit in Bezug auf die optimale Lösung gestellt werden (Luger, 2009, S. 124). Obwohl in einigen Fällen die maximale Abweichung einer heuristischen Lösung zur exakten Lösung abgeschätzt werden kann, ist in vielen Fällen weder eine stochastische Treffgenauigkeit noch eine oberer Schranke für die Abweichung vom Optimum gegeben. Die Wirksamkeit einer Heuristik kann dann empirisch eingeschätzt werden. Hierzu eignet sich die Methode der Simulation.

Wenn heuristische Verfahren für viele unterschiedliche Problemklassen ein-

[27] Luger (2009, S. 123 f.) spricht nicht explizit von einem Erfordernis. Dieses leitet sich aus den Eigenschaften der Problemklassen ab.

setzbar sind und jeweils speziell auszuprägen sind, dann werden die Verfahren als Meta-Heuristik bezeichnet. Der Vorteil von Meta-Heuristiken ist deren Generizität. Eine solche Meta-Heuristik ist die *„Bestensuche“* (vgl. auch im Weiteren: Russell und Norvig, 2003, S. 94 ff.; Luger, 2009, S. 133 ff.). Obwohl dessen Grundform lange bekannt und untersucht ist und einige allgemeingültige Aussagen dazu formuliert werden konnten (z. B. Dechter und Pearl, 1985), stellt die Ausprägung der Meta-Heuristik eine eigene Problemstellungen dar, worauf weiter unten noch einmal eingegangen wird. Zuvor soll die Bestensuche als sog. *A-Suche* beschrieben werden. (Definition 2, unten, expliziert die kennzeichnenden Eigenschaften.)

In einem Zustandsgraphen[28] seien $g(\tau)$ die Kosten, die auf dem Weg vom Startzustand zum Zustandsknoten τ entstehen, und es sei $h(\tau)$ eine Schätzfunktion, welche die minimalen Kosten abschätzt, die auf dem Weg vom Knoten τ zum Zielknoten anfallen. Dann ist $e(\tau) = g(\tau) + h(\tau)$ eine heuristische Evaluationsfunktion, welche den Knoten τ bewertet. Es kann hier davon ausgegangen werden, dass g und h stets positiv (oder 0) sind und, dass g entlang eines Pfades vom Start- zum Zielknoten monoton wächst (oder gleich bleibt).

Um einer etwaigen begrifflichen Verwirrung entgegenzutreten, wenn es um den Begriff „Heuristik“ geht: Es wird sich zeigen, dass $h(\tau)$ den Kern für das heuristische Vorgehen darstellt. Aus diesem Grunde kann und soll diese Funktion als *die Heuristik* bezeichnet werden. Diese ist dann nicht mit der heuristischen Evaluationsfunktion $e(\tau)$ zu verwechseln, die als *heuristisch* bezeichnet wird, weil sie die Heuristik $h(\tau)$ verwendet. Die Verfahrensweise, die darauf basierend das heuristische Vorgehen zum Finden eine Lösung bestimmt, wird als Meta-Heuristik bezeichnet. Algorithmus 1 formuliert die hier zugrundeliegende Meta-Heuristik.

Der aufmerksame Leser mag sich nach Studium des Algorithmus 1 die Frage gestellt haben, wie in einer Baumstruktur, wie in Abb. 2.6 dargestellt, überhaupt die Fälle eintreten sollen, dass ein durch Expandieren entdeckter Nachfolgeknoten schon in der Liste noch zu besuchender oder gar in der Liste der bereits besuchten Knoten enthalten ist (also, dass einer der Else-if-Zweige aktiv wird). Schließlich gibt es in Baumstrukturen zu jedem Knoten nur jeweils genau einen Pfad. Die Fälle sind in der soeben beschriebenen Meta-Heuristik enthalten, weil sie sich auf Zustandsgraphen bezieht, wo es auch mehrere Pfade zu einem Zustandsknoten geben kann. Obwohl Bäume spezielle Graphen sind, lässt sich die Suche im Zustandsraum auch immer als Baum strukturieren, indem einzelne identische Zustände ggf. mehrfach als Knoten repräsentiert werden, je nachdem aus welchen Vorgängerknoten sie expandiert und besucht

[28] Bäume sind spezielle Graphen.

Algorithmus 1 Bestensuche als Meta-Heuristik (vgl. Luger, 2009, S. 134)

input: *Startzustand* (Wurzelknoten)
returns Heuristisch bestimmter Pfad zu einem definierten Zielzustand in einem Zustandsgraphen.

```
open : (leere) Liste für noch zu besuchende Knoten.
closed : (leere) Liste für bereits besuchte Knoten.
Füge den Startzustand zu open hinzu.
while open ist nicht leer do
    T ← Erstes Element aus open. Entferne T aus open.
    if T erfüllt die Bedingungen für einen Zielzustand then
        return Pfad vom Startzustand zu T.  ▷ Erfolgreiche Terminierung.
    end if
    Expandiere T, d. h. generiere logische Nachfolgerzustände.
    for all Nachfolgerzustände τ von T do
        if τ weder in open noch in closed then      ▷ (Erster Besuch von τ)
            e(τ) ← Heuristische Bewertung für τ.
            Füge τ zu open hinzu.
        else if τ in open, aber nun mit geringeren Kosten erreicht then
            g(τ) ← Die (geringeren) Kosten, mit denen τ nun erreicht wurde.
        else if τ in closed, aber nun mit geringeren Kosten erreicht then
            Entferne τ aus closed.
            g(τ) ← Die (geringeren) Kosten, mit denen τ nun erreicht wurde.
            Füge τ zu open hinzu.
        end if
    end for
    Füge T zu closed hinzu.
    Sortiere open gemäß heuristischer Bewertung.
end while
return Kein Zielzustand erreicht.                   ▷ Erfolglose Terminierung.
```

werden.

Die beschriebene Meta-Heuristik „Bestensuche“ sorgt mit der anhand der heuristischen Bewertungsfunktion $e(\tau)$ sortierten Liste dafür, dass die Suche möglichst schnell in eine vielversprechende Richtung gelenkt wird. Dafür ist entscheidend, dass die heuristische Bewertungsfunktion eine möglichst genaue Abschätzung der Kosten vom Start- zum Zielzustand liefert. Wenn in jedem Zustand die Kosten vom Start- zum Zielknoten bekannt wären, dann könnte der Pfad zum Zielknoten direkt gefunden werden. Die Bestimmung von $e(\tau) = g(\tau) + h(\tau)$, wie oben definiert, lässt hier eine wichtige allgemeingültige, streng-deduktive Aussage zu (Luger, 2009, S. 145 f.; Russell und Norvig, 2003, S. 97): Wenn $h(\tau)$ die tatsächlichen minimalen Kosten zur Erreichung des Zielzustandes aus dem Zustand τ niemals überschätzt, so findet die Bestensuche die optimale Lösung, wenn diese existiert, und wird dann auch als *A*-Suche* bezeichnet.[29]

Definition 2 (A-Suche, A*-Suche, zulässige Suche) *Nach Luger (2009, S. 146): Eine A-Suche implementiert die Bestensuche mit $e(\tau) = g(\tau) + h(\tau)$ wie oben beschrieben. Eine A*-Suche ist eine A-Suche, bei der $h(\tau)$ die (minimalen) Kosten zur Erreichung des Zielzustand niemals überschätzt. Eine Suche über einen Zustandsraum heißt zulässige Suche, wenn sie den optimalen Pfad zu einem Zielzustand findet, falls er existiert.*

Es gilt: Jede A*-Suche ist eine zulässige Suche (ebd.). Die Konstruktion einer Heuristik für solch eine zulässige Suche ist immer möglich, z. B. mit $h(\tau) = 0, \forall\tau$, wenn die Kosten entlang eines Pfades monoton steigen. Dann findet die Bestensuche zwar garantiert die optimale Lösung, allerdings durchsucht sie dann den kompletten Zustandsraum. (Sie wird zu einer sog. Breitensuche.) Das heißt, die Bestensuche erfüllt noch nicht ohne Weiteres, die oben formulierte Forderung, eine Ebene der Intelligenz einzuziehen, die im Bedarfsfall eine exhaustive Suche effektiv verhindert.

Um die Verhinderung eine exhaustiven Suche zu gewährleisten, könnte die Meta-Heuristik angepasst werden, z. B. durch Löschen der Anweisung in Algorithmus 1, T zu *closed* hinzuzufügen, und Einfügen einer neuen letzten Anweisung in die While-Schleife: „Behalte nur das erste Element in *open*, lösche alle anderen.“ Dies führt dazu, dass die Suche für jede Verzweigung stets nur die dort gemäß $e(\tau)$ am besten bewertete Alternative weiterverfolgt, ohne jemals eine andere auszuloten.

[29] Agostinelli et al. (2019) verwenden eine Variante der A*-Suche zur Lösung des Rubik-Würfels.

Solch eine Suchstrategie kann als eine Art *Hill-Climbing* betrachtet werden (Luger, 2009, S. 127 ff.; Russell und Norvig, 2003, S. 111 ff.).[30] Dadurch wird zwar die Komplexität des Verfahrens extrem reduziert, weil sie dann linear statt exponentiell ist. Auf der anderen Seite ist ein Zurückspringen in einen vormals entdeckten Zustand (*Backtracking*) nicht möglich, sodass begonnene Pfade, die anfangs vielversprechend erschienen, nicht mehr verlassen werden können, wenn sie sich später als unvorteilhaft herausstellen. Wenn die Heuristik $h(\tau)$ die Kosten zum Zielzustand aber genau genug abschätzt, dann führt die vorgeschlagene Strategie gleichsam auf direktem Wege zur optimalen Lösung. Dies zeigt die Bedeutung der Heuristik. Durch sorgfältige Konstruktion, kann sie ohne exhaustive Auslotung des Zustandsraumes zu zufriedenstellenden Lösungen führen.

Wie bereits oben geschrieben, bildet $h(\tau)$ den Kern des heuristischen Vorgehens. Bei der sog. *gierigen* Bestensuche wird sogar ausschließlich $h(\tau)$ verwendet, d. h. die Suche wird allein anhand der Schätzung der Kosten, um aus dem aktuell untersuchten Zustand τ zum Ziel zu gelangen, gesteuert (Russell und Norvig, 2003, S. 95 ff.). Der Verzicht auf den Einbezug der Kosten $g(\tau)$, die bereits anfielen, um aus dem Startzustand zu τ zu gelangen, kann aber ineffektiv werden, d. h. zu sehr stark von den Zielkriterien abweichenden Lösungen führen, wenn die Heuristik eine unsachgemäße Abschätzung liefert. $g(\tau)$ stellt somit ein gewisses Korrektiv für die Suche anhand tatsächlich angefallener Kosten dar.

Dieses Korrektiv ist aber umso weniger nötig, je sachgerechter die Heuristik ist. Außerdem ist die Suche umso effizienter, je sachgerechter die Heuristik ist, d. h. die Suche muss weniger Alternativen ausloten. Gleichzeitig ist die Suche umso effektiver, je sachgerechter die Heuristik ist, d. h. die Suche führt zu einem gemäß Zielkriterium besseren Ergebnis. Entscheidend ist letztlich wie sachgerecht die Heuristik $h(\tau)$ ist. Diese bildet im Grunde genommen die eigentliche Ebene der Intelligenz für die Lösungssuche. Eine sachgerechte Heuristik bildet implizit spezifisches Wissen über die Problemdomäne ab, für die eine Lösung gesucht wird. Russell und Norvig (ebd., S. 123) sprechen daher statt von heuristischer Suche auch von *informierter* Suche. Denn die Heuristik verarbeitet spezifische Informationen zur Problemlösung. Es gilt: Wenn zwei Heuristiken $h_1(\tau)$ und $h_2(\tau)$ jeweils für kein τ die minimalen Kosten zum Zielzustand überschätzen, dann ist h_2 informierter als h_1, wenn $h_1(\tau) \leq h_2(\tau), \forall\tau$ (Luger, 2009,

[30] Hill-Climbing-Strategien verfolgen eigentlich einen Pfad nur solange, wie sich die Bewertung verbessert. In der hier hypothetisch vorgeschlagenen Anpassung der Bestensuche ist dies nicht so und ein Pfad wird immer bis zum Ziel verfolgt. Die Bezeichnung als Hill Climbing ist hier durch die jeweilige Verfolgung der gemäß $e(\tau)$ am besten bewertete Alternativen begründet.

S. 148).

Wie oben bereits geschrieben, stellt die Konstruktion sachgerechter Heuristiken auch bei Nutzung einer bekannten Meta-Heuristik eine Problemstellung für sich da. Denn um eine effiziente und effektive Problemlösung zu erreichen muss die Heuristik auf die Problemdomäne ausgerichtet werden. Dabei gilt es, möglichst viele Informationen über das konkret zu lösende Problem durch die Heuristik in dessen Lösung einzubringen. Dies erfordert ein tiefergehendes Verständnis der potenziellen Instanzen einer Problemdomäne. Die Aufgabe besteht dann darin, relevante Faktoren mit Prädiktorcharakter zu finden, zu formalisieren und adäquat in der Heuristik abzubilden. Im Prinzip ist eine formale Theorie der Problemdomäne mit hohem Prognosepotential erforderlich. Dabei muss vorzugsweise eine Formulierung gefunden werden, die sich für jede Probleminstanz parametrisieren lässt. Das Spezielle muss allgemein formuliert werden.

Es ist einleuchtend, dass für spezielle Problemdomänen spezifische Heuristiken zu finden sind, die besser geeignet sind, als eine allgemeine Heuristik. Dies gilt sogar innerhalb einer Problemdomäne für unterschiedliche Probleminstanzen. Wenn z. B., um die Einsicht plakativ zu machen, nur für eine Instanz des Problems der Allokation von Sitzgelegenheiten im urbanen Raum eine Lösung zu finden ist, bei der genau bekannt ist wie viele Passanten und wie viele adaptive Sitzgelegenheiten in der Diskurswelt enthalten sind, wie die Sitzgelegenheiten räumlich verteilt sind, wie schnell die Passanten gehen, wann sie Pause brauchen usw., dann könnte mithilfe dieser Informationen prinzipiell eine exakte Lösung vorbestimmt werden, dessen Ergebnis dann für die Heuristik als „Schätzung“ verwendet wird. Auch wenn dies praktisch ggf. nur mit sehr langer Vorlaufzeit möglich wäre, entfällt diese Möglichkeit sogar prinzipiell, wenn die Zahl der zu berücksichtigenden Instanzen unendlich ist.

Die o. g. Einsicht lässt sich aber auch auf die Ebene der Lösungsverfahren übertragen. Das soeben plakativ beschriebene Verfahren, Lösungen vorauszuberechnen und zu hinterlegen, funktioniert nicht für Problemdomänen mit unhandhabbar vielen infrage kommenden Instanzen. Die sog. *No-Free-Lunch*-Theoreme formalisieren und beweisen diese Einsicht für Such- und Optimierungsprobleme (Wolpert und Macready, 1996, 1997). Die wesentliche Aussage ist: Wenn sich zwei Probleme hinsichtlich bestimmter Spezifika unterscheiden, dann kann ein spezielles Lösungsverfahren gefunden werden, welches eines dieser Probleme besser löst als ein allgemeines Lösungsverfahren, welches beide Probleme löst. Das spezielle Lösungsverfahren löst dann aber das andere Problem schlechter als das allgemeine Lösungsverfahren.

2.7.3 Schlussfolgerungen

Es wurde aufgezeigt, dass die Konstruktion adaptiver Sitzgelegenheiten, deren Verfügbarkeit im urbanen Raum koordiniert wird, relevante Probleme für Passanten mit alterstypischen Beeinträchtigungen löst (Kap. 2.4), dass hierzu aber noch keine dedizierte Lösung existiert (Kap. 2.5). Zudem wurde dargelegt, wie ein Zielkriterium für die Allokation gewissen Gerechtigkeitsanforderungen als Wohlfahrtsziel gerecht werden kann (Kap. 2.6). Zum Finden entsprechender Lösungen sind elaborierte Methoden notwendig. Dazu werden hier Methoden der KI angewendet.

Das Lösen des sich als Allokationsproblem ausprägenden Koordinationsproblems wird dabei als eine Suche im Zustandsraum konzipiert. Für diese Suche soll dabei aber nicht Zulässigkeit gemäß Definition 2 gefordert werden, da fraglich ist, ob dies mit hinnehmbarem Suchaufwand überhaupt möglich ist. Gleichwohl wird eine A-Suche gemäß Definition 2 angestrebt, die mit hinnehmbaren Suchaufwand möglichst nahe an eine zulässige Suche heranreicht. Um die Hinnehmbarkeit des Suchaufwandes garantierten zu können, soll das Anpassen der zugrundeliegenden Meta-Heuristik „Bestensuche“ zur Kontrolle des *Backtrackings*, d. h. hinsichtlich der *open*- und *closed*-Listen, „erlaubt“ sein. Die eigentliche Kernaufgabe besteht aber in der Konstruktion der Heuristik $h(\tau)$: Diese bildet den Kern des Verfahrens sowie seiner Wirksamkeit.

Postulat 2 *Es ist eine Variante der A-Suche zu finden, die sicherstellt, dass die Suche in jedem Fall nach hinnehmbaren Aufwand mit einer Lösung terminiert, welche das Zulässigkeitskriterium gem. Def. 2 zwar nicht erreichen muss, aber anstrebt. Hierzu ist insbesondere eine Heuristik zu entwickeln.*

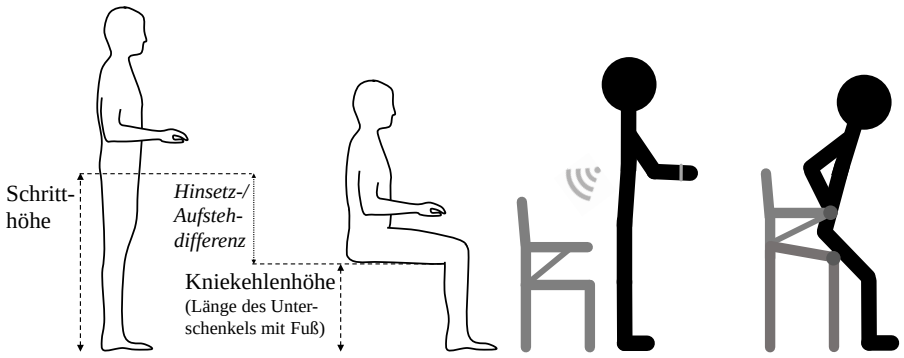
3 Entwicklung eines Safety-orientierten Koordinationsverfahrens

3.1 Präformale Konzeption

Für die übergeordnete Zielstellung, Adaptivität des urbanen Raums mittels smarter städtebaulicher Objekte zu erhöhen, wird ein Lösungsansatz für spezielle Sitzgelegenheiten entwickelt: Je nach individuellen Erfordernissen werden bestimmte, geeignete Sitzgelegenheiten explizit zur Nutzung freigeben bzw. zurückgehalten.

Das Koordinationsproblem besteht in der Allokation von Sitzgelegenheiten. Die Allokation bestimmter, geeigneter Sitzgelegenheiten kann v. a. für Passanten mit alterstypischen Beeinträchtigungen, wie in Kap. 2.4 beschrieben, eine bedeutende Verbesserung der Benutzbarkeit des urbanen Raums darstellen. Passanten mit alterstypischen Beeinträchtigungen haben das Problem, dass sie nur relativ kurze Strecken zu Fuß zurücklegen können und dann ein Erfordernis nach einer Pause im Sitzen haben. Den Passanten sollten deshalb immer dann, wenn sie ein Pausenerfordernis haben, eine Sitzgelegenheit zur Verfügung stehen. Dies konstituiert eine Anforderung nach *(a) erfordernisgenauer Verfügbarkeit* von Sitzgelegenheiten. Die Möglichkeit, dass die Passanten einfach einen Rollator o. Ä. als eine Art mobile Sitzgelegenheit mit sich führen, wird ausgeklammert, weil dies die Benutzbarkeit des urbanen Raums in vielen Fällen reduziert, z. B. bei der Überwindung von Stufen. Die Erfüllung der Anforderung nach erfordernisgenauer Verfügbarkeit von Sitzgelgenheiten kann dazu führen, dass Passanten, die sonst nur mit einem Rollator hinaus in den urbanen Raum gehen, keinen Rollator mehr benötigen, sondern bspw. mit einem Gehstock auskommen. Aber auch, wenn ein Passant nicht auf einen Rollator als Gehhilfe verzichten kann, sind dedizierte Sitzgelegenheiten für Pausen besser geeignet.

Passanten mit alterstypischen Beeinträchtigungen haben außerdem Schwierigkeiten, ihren Körperschwerpunkt kontrolliert vertikal zu verändern (vgl.



(a) Relevante anthropometrische Maße (nach Kothiyal und Tettey, 2001)

(b) Assistenzfunktionalität

Abbildung 3.1: Adaptive Parkbank mit an Körpermaßen ausgerichteter Hinsetz- und Aufstehassistenz

Rinkenauer, 2008, S. 145 ff.). Dadurch haben sie Probleme, beim Hinsetzen und Aufstehen den Höhenunterschied zwischen Schritthöhe (Adler et al., 2010, S. 161) und Höhe der Sitzfläche sicher zu überwinden. Hieraus ergibt sich eine Anforderung nach *(b) Unterstützung beim Hinsetzen und Aufstehen.* Kothiyal und Tettey (2001) empfehlen, Sitzgelegenheiten mit einer erhöhten Sitzfläche auszustatten, um das Hinsetzen und Aufstehen zu erleichtern. Einige spezielle Sitzgelegenheiten sind mit Gasdruckfedern ausgestattet, die das Hinsetzen und Aufstehen mechanisch unterstützen. Sitzgelegenheiten sind also bezüglich der Eignung für Passanten mit alterstypischen Beeinträchtigungen differenziert. Unter Nutzung dynamischer Konstruktionsmittel kann diese Differenzierung bis hin zu Individualisierung gehen.

Abbildung 3.1 skizziert, wie eine Sitzgelegenheit durch Ausrichten der Höhe ihrer Sitzfläche an individuelle anthropometrische Daten ihre Gebrauchstauglichkeit erhöhen kann (vgl.: Adler et al., 2010, S. 161 f.; Kothiyal und Tettey, 2001, S. 20 f.). Unter Ausnutzung dynamischer Gestaltungsmittel mittels Linearaktuatoren, kann solch eine Sitzgelegenheit aktive Assistenz leisten, indem die Sitzfläche (I.) für das Hinsetzen auf Schritthöhe hochfährt und sich leicht nach vorne neigt, (II.) für das ergonomische Sitzen auf Kniekehlenhöhe herunterfährt und sich leicht nach hinten neigt und (III.) für das Aufstehen wieder auf Schritthöhe hochfährt und sich nach vorne neigt.[31]

Im urbanen Raum sind Sitzgelegenheiten i. d. R. für mehr als eine Person ausgelegt, z. B. Parkbänke. Eine mechanische Assistenzfunktionalität sollte de-

[31] In Hubl et al. (2018) sind einige technische Details dazu beschrieben. (Der Beitrag der vorliegenden Arbeit hat einen anderen Fokus, s. u. in diesem Abschnitt)

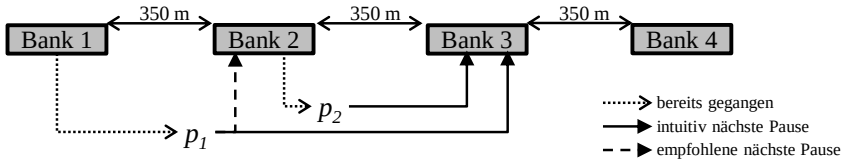


Abbildung 3.2: Anschauungsbeispiel für das Allokationsproblem

aktiviert werden, sobald eine andere als die zu assistierende Person die Sitzgelegenheit benutzt, um Konflikte und Risiken bei der gleichzeitigen Benutzung durch mehr als eine Person zu vermeiden. Die situationsbezogene Nichtbenutzbarkeit der mechanischen Assistenzfunktionalität führt dazu, dass es für Passanten mit alterstypischen Beeinträchtigungen aus *technisch-konzeptionellen* Gründen vorteilhafter ist, wenn beim Hinsetzen und beim Aufstehen jeweils kein anderer Passant dieselbe Sitzgelegenheit benutzt.

Abbildung 3.2 veranschaulicht ein Beispiel für das dahinter liegende Allokationsproblem: Passant p_1 machte eine Erholungspause auf Bank 1 und Passant p_2 machte eine Erholungspause auf Bank 2. p_1 kann 600 m gehen, bis er das Erfordernis nach einer Pause hat. p_1 setzt sich also auf die Bank 3, wenn er sich bei einem Pausenerfordernis stets auf die in Gehrichtung nächste Sitzgelegenheit setzt. p_2 kann nur 300 m gehen, bis er ein Pausenerfordernis hat. Da er folglich bei jeder Sitzgelegenheit eine Pause benötigt, setzt er sich auch auf die Bank 3. Wenn die Bänke nun *individuell* an einzelne Passanten anpassbar sind, dann *kann* es vorteilhaft sein, dass p_1 schon bei Bank 2 eine Pause macht, auch wenn die Sitzplatzkapazität der Bank 3 für zwei Passanten ausreichen würde.

Im Rahmen dieser Arbeit werden Sitzgelegenheiten in sog. *adaptive Sitzgelegenheiten* als SSOs transformiert. Diese können ihre Belegung überwachen und durch aktives Zurückhalten und Freigeben von Sitzplätzen ihre Verfügbarkeit steuern. Dadurch kann ein System konstruiert werden, das Sitzplätze alloziert. Wenn die Zielfunktion für die Allokationen auf die Anforderung nach erfordernisgenauer Verfügbarkeit von Sitzgelegenheiten ausgerichtet wird, dann realisiert der urbane Raum eine *(a) Adaptivität bzgl. Verfügbarkeit von Sitzgelegenheiten*. Hierin geht die Anforderung nach Unterstützung beim Hinsetzen und Aufstehen als konzeptionelle Nebenbedingung für die Zielfunktion als *(b) Adaptivität bzgl. Höhe der Sitzfläche* mit ein.

Abbildung 3.3 stellt den zugrundeliegenden Safety-Zusammenhang als ein Vier-Felder-Portfolio dar. Generell gilt dabei: Wenn der Personenzustand kritisch ist, dann muss ein unkritischer Umweltzustand sichergestellt werden. Wenn der Umweltzustand kritisch ist, dann muss versucht werden, den Per-

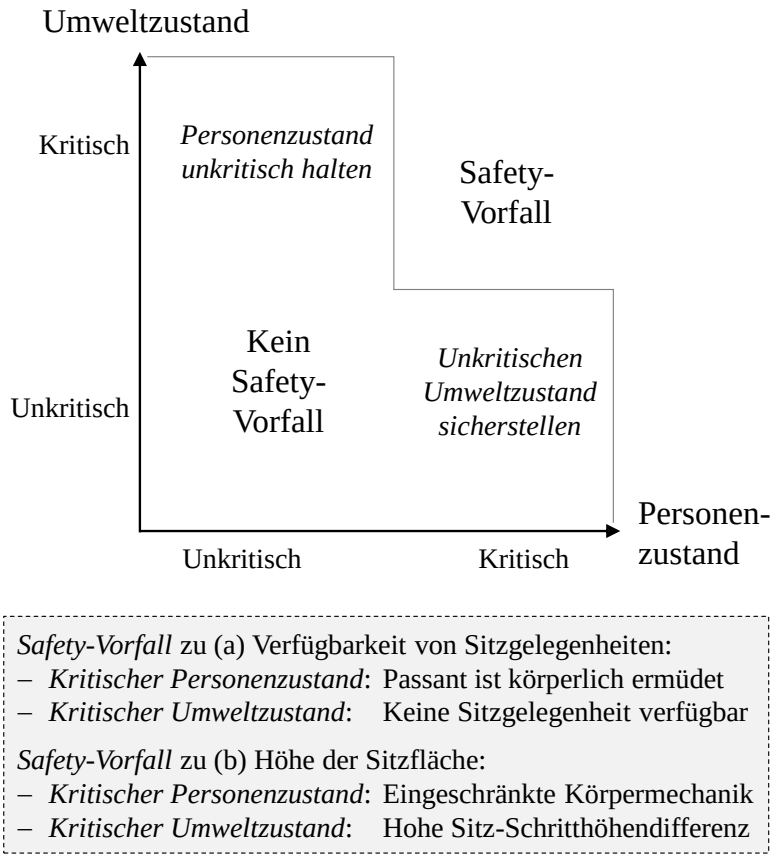


Abbildung 3.3: Safety-Vorfälle als Zusammenkommen eines kritischen Umweltzustands mit einem kritischen Personenzustand

sonenzustand unkritisch zu halten. Insgesamt sind alle Zustandskombinationen unterhalb der dünnen grauen Abtrennungslinie akzeptabel, da diese keine Safety-Vorfälle bilden. Die Ausprägung der Safety-Konzeptualisierung für die vorliegende konkrete Problemstellung, erfolgt durch die Ergänzungen im grauen Kasten unten in der Abbildung. Diese Konzeptualisierung wird im Folgenden der Zielfunktion für das zu entwickelnde Verfahren zur Allokation geeigneter Sitzgelegenheiten zugrunde liegen.

Abbildung 3.4 prägt das Grundmodell für IoT-Objekte (vgl. Abb. 2.1) für die hier als Lösungsansatz konzipierten adaptiven Sitzgelegenheiten aus. Hierzu wird die KI-basierte Informationsverarbeitung entwickelt, welche die Funktionalität des Safety-Controllers darstellt (vgl. Abb. 1.2). Dieser realisiert ein übergreifendes Koordinationsverfahren für das gesamte SSO-System. Das Verfahren wird dabei zentral ausgeführt und das Ergebnis dann an die einzelnen

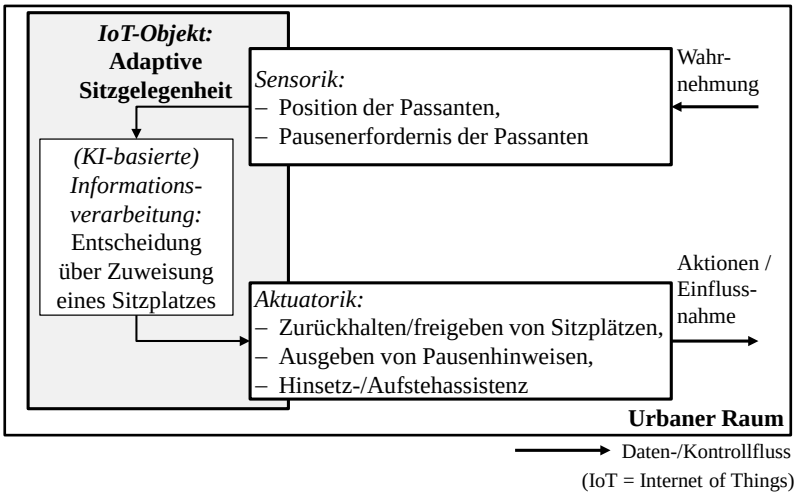


Abbildung 3.4: SSO „Adaptive Sitzgelegenheit“ als Ausprägung des IoT-Grundmodells

SSOs verteilt. Dies lässt sich konzeptionell aber so auffassen, dass das KI-basierte Informationsverarbeitungsverfahren für die Koordination bei jedem SSO lokal ausgeführt wird und jeweils zum selben Ergebnis führt.

3.2 Formalisierung der Problemstellung

3.2.1 Modell der Diskurswelt

Es folgt nun die Entwicklung einer Lösung für die Problemstellung. Dazu wird zunächst die Diskurswelt formalisiert und auf dieser Basis das zu lösende Problem definiert. Die (Art und Weise der) Lösung wird dann im Kap. 3.3 als parametrisierbares und automatisiert durch IT ausführbares Verfahren entwickelt.

Sei $Passanten := \{p_1, p_2, \ldots, p_i, \ldots, p_n\}$ eine Menge von n Passanten. Die Passanten werden im Folgenden auch einfach mit $i \in \{1, \ldots, n\}$ referenziert. (Passant i ist der i-te Passant, also p_i.)

Sei $SSOs$ eine Menge von k adaptiven Sitzgelegenheiten. Wenn im Folgenden von SSOs gesprochen wird, dann sind stets adaptive Sitzgelegenheiten dieser Menge gemeint. Auf die SSOs wird mit $j \in \{1, \ldots, k\}$ referenziert.

Sei $B \in \mathbb{B}^{n \times k}$ mit $\mathbb{B} = \{0; 1\}$ eine Belegungsmatrix,

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,j} & \cdots & b_{1,k} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,j} & \cdots & b_{2,k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{i,1} & b_{i,2} & \cdots & b_{i,j} & \cdots & b_{i,k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,j} & \cdots & b_{n,k} \end{pmatrix}, \tag{3.1}$$

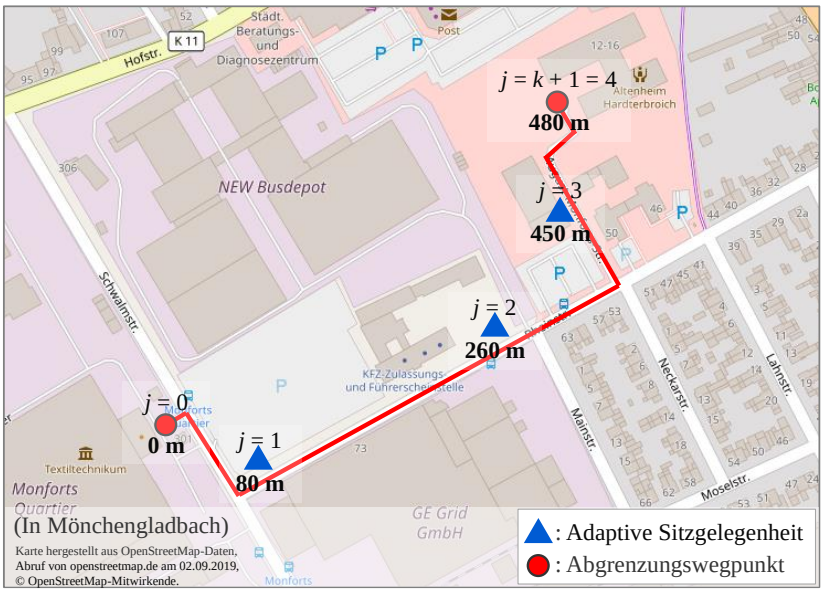
für die gilt:

$$\begin{aligned} b_{i,j} = 1 &\Leftrightarrow \text{Passant } i \text{ belegt Sitzgelegenheit } j \text{ genau einmal.} \\ b_{i,j} = 0 &\Leftrightarrow \text{Passant } i \text{ belegt Sitzgelegenheit } j \text{ keinmal.} \end{aligned} \tag{3.2}$$

In der Diskurswelt kann also jeder Passant jede Sitzgelegenheit maximal einmal belegen. Dies entspricht der Vorstellung, dass die Sitzgelegenheiten, die ein Passant i bereits passierte, für i nicht mehr (erneut) als Pausenmöglichkeit in Betracht kommen, weil die Passanten entlang einer Strecke zu einem Ziel gehen und auf dem Weg nicht umkehren (möchten). Es gelte die Annahme, dass die Passanten die Sitzgelegenheiten genau gemäß B belegen. Die gesetzten Annahmen werden in Kap. 5 noch einmal reflektiert.

Die Sitzgelegenheiten seien Wegpunkte einer definierten Strecke. Diese Strecke wird durch zwei Wegpunkte abgegrenzt, die nicht aus der Menge der $SSOs$ sind. Diese Abgrenzungswegpunkte werden mit $j = 0$ und $j = k + 1$ referenziert. Die Strecke kann einen typischen Weg zwischen zwei markanten Punkten darstellen und eindimensional projiziert werden. Die Abbildung 3.5 veranschaulicht dies an einem Weg zwischen einem Altenheim in Mönchengladbach (rechts) und einer ehemaligen Textilfabrik (links), die als beispielhaftes fußläufig erreichbares Ausflugsziel gilt. (Die adaptiven Sitzgelegenheiten sind hypothetisch eingetragen.)

Sei $pos(j) \in \mathbb{R}$ die Position des Wegpunktes j. Positionen geben die Distanz zu einem definierten Referenzpunkt der Strecke wieder (und sind daher eindimensional). Es gibt $k + 2$ Wegpunkte: Wegpunkt 0 und Wegpunkt k + 1 sind die Abgrenzungswegpunkte. Die Wegpunkte 1 bis k sind die SSOs auf der Strecke.
Es gilt per definitionem $\forall j, j' \in \{0, \ldots, k + 1\} : j < j' \Leftrightarrow pos(j) < pos(j')$. Die Positionen der Wegpunkte steigen also mit ihrer Ordnungszahl j, wobei die Wegpunkte paarweise verschiedene Positionen haben. Dies bedeutet auch,



Projektion der hervorgehobenen Strecke:

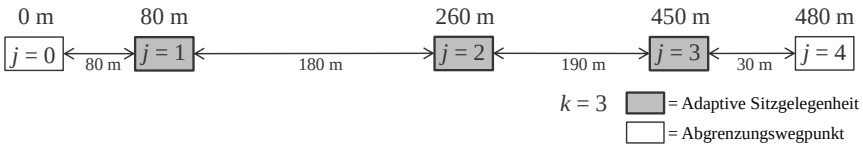


Abbildung 3.5: Beispielstrecke

dass Abgrenzungswegpunkte nicht bei Sitzgelegenheiten positioniert werden. *Hinweis:* Für $1 \leq j \leq k$ kann $pos(j)$ als Position des Mittelpunkts einer Sitzgelegenheit aufgefasst werden; die Ausdehnung der Sitzgelegenheiten entlang der Strecke spielt keine Rolle.

Sei $dir_i \in \{-1; +1\}$ die Gehrichtung von Passant i. Diese sei konstant. Wenn $dir_i = +1$, dann geht Passant i aus Richtung des Abgrenzungswegpunktes $j = 0$ in Richtung des Abgrenzungswegpunktes $j = k + 1$. Wenn $dir_i = -1$, dann geht Passant i aus Richtung des Abgrenzungswegpunktes $j = k + 1$ in Richtung des Abgrenzungswegpunktes $j = 0$.

Sei τ ein Diskursweltzustand. (Im Abschnitt 2.7.2 wurde τ bereits implizit als Variable für Zustandsknoten eingeführt. Im Abschnitt 3.3.1 wird explizit gemacht, dass für das zu entwickelnde Lösungsverfahren *bestimmte* Diskurswelzustände durch Zustandsknoten repräsentiert werden.)

Sei $pos_i(\tau) \in \mathbb{R}$ die Position von Passant i im Diskursweltzustand τ.

Sei $f_i(\tau) \in [0;1]$ der Ermüdungsgrad von Passant i im Diskursweltzustand τ (f für *„fatigue“*, englisch für Ermüdung).

Sei $q_i(\tau) \in \{$*gehend, sitzend, Strecke bewältigt*$\}$ der Bewegungszustand von Passant i im Diskursweltzustand τ.
Abbildung 3.6 stellt die möglichen Übergänge zwischen den Bewegungszuständen dar: Die als Pfeile dargestellten Zustandsübergänge entsprechen einer Änderung des Diskursweltzustandes in einem gedachten Zeitschritt. Jeder Passant startet im Zustand (a), dargestellt durch den auf (a) gerichteten Pfeil ohne Vorgängerzustand. Für die Zustandsübergänge (1) – (5) gelten die folgenden Regeln:

(1) Der Zustandsübergang von *gehend* zu *gehend*, erfolgt dann immer, wenn sich der entsprechende Passant nicht an einem Wegpunkt befindet. Der Zustandsübergang erfolgt auch, wenn sich der Passant an einem Wegpunkt befindet, aber weder die Bedingung für den Übergang (2) noch für (5) erfüllt ist.

(2) Der Zustandsübergang von *gehend* zu *sitzend* erfolgt, wenn der Passant an einer Sitzgelegenheit ankommt, die er gemäß B benutzen soll.

(3) Der Zustandsübergang von *sitzend* zu *sitzend* erfolgt, wenn die Verweildauer des Passanten auf der benutzten Sitzgelegenheiten noch nicht vorüber ist.

(4) Der Zustandsübergang von *sitzend* zu *gehend* erfolgt, wenn die Verweildauer des Passanten auf der benutzten Sitzgelegenheit gerade vorübergeht.

(5) Der Zustandsübergang von *gehend* zu *Strecke bewältigt* erfolgt, wenn der Passant an dem Abgrenzungswegpunkt am Ende seiner Strecke ankommt. Der dann resultierende Zustand (c) ist der finale Bewegungszustand, dargestellt durch die doppelte Umrandung des Zustandes.

Sei $\dot{f}_i \in [0;1]$ die Ermüdungsrate von Passant i. Die Ermüdungsrate stellt einen gleichmäßigen Ermüdungszuwachs pro Zeiteinheit dar. (Der Punkt über dem f-Symbol signalisiert, dass es sich um eine Änderungsrate des Ermüdungsgrades nach der Zeit handelt.)
Wenn die Ermüdungsrate pro Streckeneinheit vorliegt, lässt sich diese aufgrund der oben explizit gemachten Annahme, dass Passanten nicht stehen, eineindeutig in eine Ermüdungsrate pro Zeiteinheit umrechnen; siehe nachfolgenden Ausdruck 3.3.

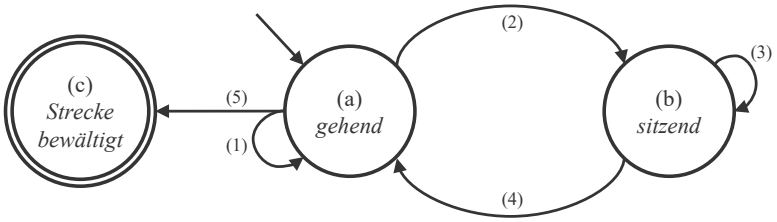


Abbildung 3.6: Zustandsübergangsdiagramm für die Bewegungszustände der Passanten

Sei $v_i \in \mathbb{R}^{>0}$ die normale Gehgeschwindigkeit von Passant i. Diese sei konstant (und echt positiv; eine normale Gehgeschwindigkeit von 0 ist nicht sinnvoll; die Gehrichtung wird nicht durch ein „eigenes“ Vorzeichen der Gehgeschwindigkeit ausgedrückt, sondern mit dir_i).
Wenn die maximale Distanz $d_i^{max} \in \mathbb{R}^{>0}$ bzw. die maximale Zeit $t_i^{max} \in \mathbb{R}^{>0}$, nach der Passant i ein Pausenerfordernis hat, bekannt ist, dann gelte:

$$\dot{f}_i = \frac{v_i}{d_i^{max}} = \frac{1}{t_i^{max}} \ . \tag{3.3}$$

Es existieren medizinisch-validierte Leistungstestverfahren, die direkt oder in Abwandlungen zur Bestimmung der Ermüdungsrate angewendet werden könnten, z. B. der 6-Minuten-Gehtest (vgl. Harada et al., 1999).

Sei $r_i \in \mathbb{R}^{>0}$ die Dauer, die Passant i auf einer Sitzgelegenheit verweilt, wenn er diese benutzt. Diese ist per Annahme so lange, dass der Passant unmittelbar nach einer Pause auf einer Sitzgelegenheit stets wieder einen Ermüdungsgrad von 0 hat.

Je nach Belegung B ergeben sich für die Passanten unterschiedliche Zeitpunkte, zu denen sie Sitzgelegenheiten passieren sowie unterschiedliche Ermüdungsgrade, mit denen sie bei Sitzgelegenheiten ankommen. Wenn bspw. bei $k = 3$ Sitzgelegenheiten für einen Passanten $b_{i,1} = 0, b_{i,2} = 1, b_{i,3} = 1$ und $dir_i = +1$ gilt, dann ist der Ankunftszeitpunkt von i bei Sitzgelegenheit $j = 3$:

$$\frac{|pos(3) - pos(0)|}{v_i} + 1 \cdot r_i \ . \tag{3.4}$$

Der Zähler im Bruch ist die Distanz vom Startpunkt von i bis zur Sitzgelegenheit $j = 3$, die dividiert durch die Gehgeschwindigkeit von i die Gehdauer

vom Startpunkt zur Sitzgelegenheit $j = 3$ ergibt. Der rechte Summand, $1 \cdot r_i$, addiert die Dauer der Pause, die i gemäß B mit $b_{i,1} = 0$ und $b_{i,2} = 1$ auf der Sitzgelegenheit $j = 2$ macht. Um den Zeitpunkt zu bestimmen, zu dem Passant i die Sitzgelegenheit $j = 3$ wieder verlässt, ist auf den Ausdruck 3.4 lediglich ein weiteres Mal r_i für die Dauer der Pause, die i auf der Sitzgelegenheit $j = 3$ macht, zu addieren.

Der aufmerksame Leser wird festgestellt haben, dass die Pausendauer als konstant je Passant modelliert ist, was bedeutet, dass auch mehrere von einem Passanten unmittelbar nacheinander eingelegte Pausen stets als von gleicher Dauer angenommen werden. Diese Dauer r_i stellt die minimale Dauer für eine vollständige Regeneration von i dar. Die Annahme der konstanten Pausendauern je Passant stellt eine modellistische Vereinfachung der Diskurswelt dar. Das Verfahren zum Finden einer Belegung wird aber implizit darauf ausgerichtet sein, dass Pausen dann (und nur dann) vorgesehen werden, wenn der betreffende Passant ein Pausenerfordernis hat, d. h. einer vollständigen Regeneration bedarf. Vor diesem Hintergrund sind durch die Annahme keine gravierenden Verzerrungen zu erwarten. Im Kap. 5 wird noch einmal darauf eingegangen, inwiefern die Annahme für die Grundsätzlichkeit des Verfahrens eine Rolle spielt.

In ähnlicher Weise wie die Zeitpunkte, lässt sich der Ermüdungsgrad bei Ankunft an einer Sitzgelegenheit berechnen. Für das o. g. Beispiel ist der Ermüdungsgrad bei Ankunft an der Sitzgelegenheit $j = 3$:

$$\min\left(\frac{|pos(3) - pos(2)|}{v_i} \cdot \dot{f}_i \;;\; 1\right) . \tag{3.5}$$

Da der Ermüdungsgrad per definitionem nicht größer als 1 wird, wird das Minimum aus dem mittels der Ermüdungsrate und der Gehdauer berechneten Wert und 1 gewählt. Im Bsp. machte i eine Pause auf der Sitzgelegenheit $j = 2$. Nach der Pause hat er per Annahme wieder einen Ermüdungsgrad von 0. Deshalb muss hier nur die Gehdauer von der Sitzgelegenheiten $j = 2$ bis zur Sitzgelegenheit $j = 3$ mit der Ermüdungsrate von i multipliziert werden.

Sei $occ_j(\tau) \subseteq \mathit{Passanten}$ die (Teil-) Menge der Passanten, welche im Diskursweltzustand τ gerade die Sitzgelegenheit j benutzen (*occ* für *„occupancy“*, englisch für Belegung/Auslastung). Passant i kann im Diskursweltzustand τ die Hinsetz- oder Aufstehassistenz der Sitzgelegenheit j nutzen, gdw. $occ_j(\tau) \cup \{p_i\} = \{p_i\}$.

Sei κ die Sitzplatzkapazität je Sitzgelegenheit. Es wird für alle Sitzgelegenheiten die gleiche Kapazität angenommen, z. B. zwei Sitzplätze für Parkbänke.

Sei $\xi_\kappa(B) \in \mathbb{B}$ eine Prüffunktion für die Machbarkeit einer Belegung. Sie gibt 1 zurück, wenn zu jedem Diskurswelt-Zeitpunkt keine Sitzgelegenheit überbelegt ist:

$$\xi_\kappa(B) = 1 \Leftrightarrow \forall \tau, j \in \{1, \ldots, k\} : \text{card}\left(occ_j(\tau)\right) \leq \kappa \ . \tag{3.6}$$

(card steht für die Kardinalität der in Klammer dahinter angegebenen Menge, d. h. hier für die Anzahl der Passanten, die im Diskursweltzustand τ die Sitzgelegenheit j benutzen.)

Sei $s_{(a)}(j, i, B) \in [0; 1]$ eine Safety-Funktion für (a) Adaptivität bzgl. Verfügbarkeit. Sie gibt eine Safety-Bewertung dafür zurück, wie erforderlich eine Pause auf der Sitzgelegenheit j für Passant i ist, wenn i zuvor gemäß B Pausen machte.

Sei $s_{(b)}(j, i, B) \in [0; 1]$ eine Safety-Funktion für (b) Adaptivität bzgl. Höhe der Sitzfläche. Sie gibt eine Safety-Bewertung in Abhängigkeit davon zurück, ob Passant i gemäß der durch B bestimmten Belegungen der anderen Passanten die Hinsetz- oder Aufstehassistenz der Sitzgelegenheit j nutzen kann.

Sei $s(j, i, B)$ eine Safety-Funktion, die aus den beiden Safety-Funktionen $s_{(a)}$ und $s_{(b)}$ als gewichtete Summe mit $\lambda_{(a)}, \lambda_{(b)} \in [0; 1]$ und $\lambda_{(a)} + \lambda_{(b)} = 1$ zusammengesetzt ist:

$$s(j, i, B) = \lambda_{(a)} \cdot s_{(a)}(j, i, B) + \lambda_{(b)} \cdot s_{(b)}(j, i, B) \ , \tag{3.7}$$

Sei $\hat{s}(i, B)$ die Safety-Gesamtbewertung der Strecke für Passant i bei gegebener Belegung B, die als arithmetisches Mittel der Werte der zusammengesetzten Safety-Funktion über alle von i benutzten Sitzgelegenheiten j bestimmt wird:

$$\hat{s}(i, B) = \frac{\sum\limits_{\{j \,|\, b_{i,j}=1\}} s(j, i, B) + s_{(a)}(j_i^{end}, i, B)}{\sum\limits_{j=1}^{k} b_{i,j} + 1} \ , \tag{3.8}$$

$$\text{mit } j_i^{end} = k{+}1, \text{ wenn } dir_i = +1, \text{ und } j_i^{end} = 0, \text{ wenn } dir_i = -1 \ .$$

Die iterative Summe im Zähler des Ausdrucks iteriert über alle Sitzgelegenheiten j, die Passant i gemäß B belegt. (Die in die Iteration eingeschlossenen Terme enden beim „+“-Zeichen.) Die iterative Summe im Nenner zählt die Anzahl der Pausen von i gemäß der Belegungsmatrix B. Die zusätzlichen einzelnen Summanden außerhalb der iterativen Summen in Nenner und Zähler stellen jeweils das Ereignis dar, dass der Passant i die Strecke bewältigt hat. Der zusätzliche Summand im Zähler impliziert, dass es vorteilhaft ist, wenn der Passant i direkt eine Pause benötigt, wenn er am Ende der Strecke ankommt. Diese Formulierung ist damit begründet, dass unnötige Pausen auf der Strecke vermieden werden sollen. Ohne diese zusätzlichen Summanden würde der Ausdruck außerdem implizieren, dass die Safety-Gesamtbewertung für einen Passanten i nicht bestimmbar ist, wenn i keine Pause macht, weil dann der Nenner des Bruchs 0 wäre. Der Fall, dass i zu keiner Sitzgelegenheit alloziert wird, kann aber sinnvoll sein, wenn i die gesamte Distanz bewältigen kann, ohne zu ermüden. In diesem Fall sollte weder der Zähler 0 sein, damit nicht zwangsläufig eine Safety-Gesamtbewertung von 0 resultiert, noch darf Nenner 0 sein.

An dieser Stelle sei noch einmal explizit darauf hingewiesen, dass ein Pausenerfordernis eines Passanten durch einen Ermüdungsgrad von 1 modelliert ist. Ein Ermüdugsgrad von 1 ist so zu interpretieren, dass der Passant ein Pausenerfordernis hat. Der Passant kann aber trotzdem noch weitergehen. Wenn ein Passant einen Ermüdungsgrad von 1 hat, dann werden keine Dringlichkeitsgrade für eine Pause unterschieden. Die relevante Information ist, welche Distanz ein Passant in ermüdetem Zustand, d. h. mit Ermüdungsgrad 1 zurücklegen muss. Ein Ermüdungsgrad *kleiner als* 1 kann als Maß dafür interpretiert werden, wie lange es noch dauert, bis ein Pausenerfordernis auftreten wird.

Sei $agg(\hat{\mathbf{s}}) \in \mathbb{R}$ mit $\hat{\mathbf{s}} \in [0;1]^n$ eine Aggregierungsfunktion für die Safety-Gesamtbewertungen der einzelnen Passanten. Die n Komponenten des Vektors $\hat{\mathbf{s}}$ sind $\hat{s}_i = \hat{s}(i, B)$. Die Aggregierungsfunktion wird in Abschnitt 3.2.2.2 durch den Ausdruck 3.30 spezifiziert.

Es sei nun eine Belegung zu bestimmen, für welche die aggregierte Safety-Gesamtbewertung möglichst hoch ist und keine Sitzgelegenheit gemäß der Ka-

pazität überbelegt wird:

$$\textit{Bestimme ein } B \in \mathbb{B}^{n\times k} \ , \quad \textit{sodass} \tag{3.9}$$

$$agg(\hat{\mathbf{s}}) \textit{ möglichst hoch ist und} \tag{3.10}$$

$$\xi_\kappa(B) = 1 \ . \tag{3.11}$$

Wenn eine KI für das System der adaptiven Sitzgelegenheiten ein solches B findet, dann können die SSOs, z. B. durch visuelle Anzeigen oder per Sprachausgabe, den Passanten mitteilen, welche Sitzgelegenheiten sie auf ihrem Weg benutzen sollen. Es wird angenommen, dass die Passanten den Pausenempfehlungen des Systems folgen. In Kap. 4 wird der potentielle Wert bzw. Nutzen des Systems für Passanten analysiert. Dieser potentielle Wert liefert dann eine rationale Begründung für die Befolgung der Systemempfehlungen. Die Begründung gilt dann streng genommen freilich nur unter den Annahmen, also z. B. auch nur, wenn die Gehgeschwindigkeiten und Verweildauern zutreffend sind und wenn die Menge der betrachteten Passanten vollständig ist. Wenn aber auch unter den, mitunter engen, Annahme keine Lösung gefunden werden kann, dann ist sie auch im Allgemeinen nicht möglich (denn unter den allgemeineren Fällen befinden sich die Fälle mit den engen Annahmen als Spezialfälle). Die Lösung lässt sich aber auch als eine normative Lösung auffassen, die gleichsam vorgibt, wie sich Passanten verhalten sollten, damit ein gemäß der zugrunde gelegten Kriterien vorteilhaftes Resultat eintreten kann. Die Vorgaben beziehen sich dabei primär auf die Sitzpausen. Das System dient in diesem Sinne als aktiver Safety-Controller für das Passantenverhalten.

3.2.2 Modellierung der Safety-Zusammenhänge

3.2.2.1 Formalisierung der Safety-Konzeption

Das Ziel der nun entwickelten Safety-Modellierung ist eine quantitativ interpretierbare Modellierung des in Abb. 3.3 dargestellten zugrundeliegenden Safety-Zusammenhangs, nach welchem ein Safety-Vorfall aus dem Zusammentreffen eines kritischen Umweltzustandes mit einem kritischen Passantenzustand resultiert. Die hier folgende Modellierung stellt den Zusammenhang dar, auf den das System der adaptiven Parkbänke als Safety-Controller ausgerichtet ist, um ein möglichst hohes Safety-Niveau zu erzielen.

Sei $d_i^{went}(\tau) \in \mathbb{R}^{\geq 0}$ die Distanz, die Passant i in einem bestimmte Diskursweltzustand τ seit seiner letzten Pause zurückgelegt hat. Sei $d_i^{togo}(\tau) \in \mathbb{R}^{\geq 0}$ die Distanz, die Passant i in einem bestimmten Diskursweltzustand τ bis zur nächsten zu benutzenden Sitzgelegenheit noch gehen muss. Es werden dabei

implizit die im vorherigen Abschnitt 3.2.1 beschriebenen Verhaltensannahmen herangezogen, dass die Passanten genau die Sitzgelegenheiten benutzen, zu denen sie durch B alloziert sind. Nun sei Folgendes definiert:

$$x_i(\tau) := \frac{d_i^{went}(\tau)}{d_i^{max}} \tag{3.12}$$

$$y_i(\tau) := \frac{d_i^{togo}(\tau)}{d_i^{max}} \tag{3.13}$$

$$z_i(\tau) := 1 - \Big(x_i(\tau) + y_i(\tau)\Big) \tag{3.14}$$

$x_i(\tau)$ und $y_i(\tau)$ normalisieren die seit der letzten Pause bereits gegangene bzw. noch zu gehende Distanz mit der maximalen komfortablen Gehdistanz d_i^{max} von Passant i. So werden die Distanzen d_i^{went} und d_i^{togo} für Passanten mit unterschiedlichen maximalen komfortablen Gehdistanzen vergleichbar gemacht. $z_i(\tau)$ stellt dann ein normalisiertes Maß für die erfordernisgenaue Sitzplatzallokation dar, was durch die äquivalente Formulierung mit Ausdruck 3.15 deutlich wird:

$$z_i(\tau) = \frac{\overbrace{\overbrace{d_i^{max} - d_i^{went}}^{\text{Distanz bis Pausenerfordernis}} - d_i^{togo}}^{\text{Differenz zu Pausenerfordernisposition}}}{\underbrace{d_i^{max}}_{\text{Normalisierungsfaktor}}} \tag{3.15}$$

Der Zähler in Ausdruck 3.15 ist die vorzeichenbehaftete Distanz zwischen der Position der allozierten Sitzgelegenheit und der Position, an der ein Pausenerfodernis entsteht. (Die Rolle des Vorzeichens wird im Anschluss an diesen Absatz expliziert.) Wenn sich i an der nächsten zu benutzenden Sitzgelegenheit j befindet, dann ist $d_i^{togo} = 0$ und der Zähler entspricht genau dem vorzeichenbehafteten Abstand zwischen diesem Wegpunkt j und dem Ort, wo i ein Pausenerfordernis hat.

Da d_i^{togo} stets um genau den Betrag abnimmt, um den d_i^{went} zunimmt und v. v., macht es keinen Unterschied für den Wert von $z_i(\tau)$, ob $z_i(\tau)$ in einem Diskursweltzustand τ ausgewertet wird, in welchem $d_i^{togo} = 0$ gilt oder $d_i^{togo} > 0$. Tatsächlich sollen später für die Verwendung von $z_i(\tau)$ nur solche Diskurswelt-zustände betrachtet werden, in denen $d_i^{togo} = 0$, d. h., in denen sich Passant i direkt bei einer Sitzgelegenheit befindet, die er benutzen soll. Die Formulierung mit den Ausdrücken 3.12 – 3.14 hat konzeptionelle Gründe, die sogleich erläutert werden. Zunächst sei die Rolle des Vorzeichens für $z_i(\tau)$ explizit gemacht:

- Wenn $z_i(\tau) = 0$, dann gibt es keine Abweichung zwischen der Position des nächsten Pausenerfordernis und der Position der nächsten allozierten Sitzgelegenheit.
- Wenn $z_i(\tau) < 0$, dann wird die allozierte Sitzgelegenheit erst erreicht, nachdem der Passant ein Pausenerfordernis entwickelte.
- Wenn $z_i(\tau) > 0$, dann wird die allozierte Sitzgelegenheit schon erreicht, bevor der Passant ein Pausenerfordernis hat.

Des Weiteren gelten folgende Interpretationen:

- $x_i(\tau)$ sei ein Passantenzustand.
  - Alle $x_i(\tau) \geq 1$ seien als kritisch bezeichnet.
  - Ein Passantenzustand gelte als umso kritischer, je höher $x_i(\tau)$ ist.
- $y_i(\tau)$ sei ein Umweltzustand.
  - Alle $y_i(\tau) > 0$ seien als kritisch bezeichnet.
  - Ein Umweltzustand gelte als umso kritischer, je höher $y_i(\tau)$ ist.
- Das erstrebenswerte Safety-Ziel sei $z_i(\tau) = 0$.
  - Wenn $z_i(\tau) < 0$, dann beschreibe es einen Safety-Vorfall.
    $z_i(\tau)$ sei dann umso gefährlicher, je kleiner es ist.
  - Wenn $z_i(\tau) > 0$, dann beschreibe es einen unerwünschten Zustand (aber keinen Safety-Vorfall).
    $z_i(\tau)$ sei dann umso unerwünschter, je größer es ist.

Durch diese Interpretation der Ausdrücke 3.12 – 3.14 ist der in Abb. 3.3 als Portfolio dargestellte zugrundeliegende Safety-Zusammenhang in eine quantitative Interpretierbarkeit überführt. $z_i(\tau)$ ist *metrisch* interpretierbar als normalisierte Abweichung von der Position, an der eine Sitzgelegenheit verfügbar ist, zu der Position, an der eine Sitzgelegenheit erforderlich ist.

Der aufmerksame Leser wird bereits festgestellt haben, dass $x_i(\tau)$ mit fortschreitender zurückgelegter Distanz seit der letzten Pause von 0 aus anwächst und 1 beträgt, wenn die zurückgelegte Distanz gleich der Distanz ist, nach welcher der Passant ein Pausenerfordernis hat. Demnach kann $x_i(\tau)$ als ein Pausenerfordernisindikator interpretiert werden, für den die Aussage gilt: „Je höher $x_i(\tau)$ bzw. die Ermüdung, desto kritischer der Personenzustand“. $x_i(\tau)$ hängt über den Ausdruck 3.3 mit der zuvor definierten Ermüdungsrate $\dot{f}_i$ zusammen und stellt demnach das gleiche dar, wie der Ermüdungsgrad $f_i(\tau)$. Die Einführung der Variable $x_i(\tau)$ für diesen Sachverhalt erfolgt aus dem konzeptionellen Grunde, dass $x_i(\tau)$ als Personenzustand definiert wird, welcher zusammen mit dem als Umweltzustand definiertem $y_i(\tau)$ den Safety-Dualismus

für $z_i(\tau)$ bildet. Dass alle $x_i(\tau) \geq 1$ als kritisch zu bezeichnen sind, ist damit begründet, dass $x_i(\tau)$ nur dann größer als 1 wird, wenn eine Distanz überschritten wird, nach welcher eigentlich eine Pause für den Passanten i erforderlich gewesen wäre. Der gemäß vorliegender Modellierung Safety-relevante Personenzustand von i im Diskursweltzustand τ ist also durch $x_i(\tau)$ als Indikator für den Ermüdungsgrad gegeben.

Für die Interpretation von $y_i(\tau)$ ist zunächst einmal klar zu machen, dass $y_i(\tau)$ den Zustand der Umwelt von i modelliert. Obwohl $y_i(\tau)$ einen Dualismus mit $x_i(\tau)$ bildet, ist $y_i(\tau)$ in der konzeptionellen Betrachtung von $x_i(\tau)$ zu lösen: $x_i(\tau)$ beschreibt den Zustand eines Passanten; $y_i(\tau)$ beschreibt den Zustand seiner Umwelt, also nicht den Passanten selbst, obwohl sich dieser in der Umwelt befindet. Konzeptionell betrachtet ist der Safety-relevante Zustand der Umwelt von i die Entfernung zur nächsten von ihm zu benutzenden Sitzgelegenheit. Zum Verständnis dieser Konzeption des Umweltzustandes und insb., warum $y_i(\tau) > 0$ als kritisch gelten soll, soll folgende Vorstellung als Hilfestellung dienen:

Der Passant i ist in der Diskurswelt situiert. Nun stellt sich i im Diskursweltzustand τ die Frage, ob er sich in diesem Zeitpunkt bzw. Diskursweltzustand für eine Pause hinsetzen kann. Diese Frage ist dabei *unabhängig vom Ermüdungsgrad* des Passanten i gestellt. Zwar ist die Frage im Besonderen dann relevant, wenn der Ermüdungsgradindikator $x_i(\tau) \geq 1$ ist. Dennoch ist der Umweltzustand losgelöst davon zu betrachten. Der Ermüdungsgrad ist quasi eine offengelassene Größe und i stellt sich die hypothetische Frage: „Wenn ich jetzt ermüde, kann ich mich dann für eine Pause hinsetzen?“. Vor diesem Hintergrund erscheint es möglicherweise passender, wenn diese hypothetische Frage mit „Nein“ beantwortet werden muss bzw. wenn $y_i^{(\tau)} > 0$, von einem *potentiell* kritischen Umweltzustand zu sprechen. Allerdings ist gemäß der Safety-Konzeption nicht der Umweltzustand *potentiell* kritisch ist, sondern ein Safety-Vorfall *tritt* dann *potentiell ein*: Wenn sich i im Diskursweltzustand τ nicht hinsetzen kann, weil die normalisierte Entfernung zur nächsten zu benutzenden Sitzgelegenheit nicht 0 beträgt, dann befindet sich i in einem kritischen Umweltzustand, der per definitionem notwendige Bedingung für einen Safety-Vorfall ist. Es ist gerade das Konstituierende des Safety-Dualismus, dass ein Safety-Vorfall nur dann auftritt, wenn ein kritischer Personenzustand und ein kritischer Umweltzustand zusammenkommen, wobei die Definition, ob ein Personenzustand oder ein Umweltzustand kritisch ist, nur im Lichte des betrachteten potentiellen Safety-Vorfalls erfolgen kann.

Vor dem Hintergrund der hypothetischen Frage, die sich Passant i im Diskursweltzustand τ stellt, ist begründbar, dass ein höheres $y_i(\tau)$ kritischer ist als ein kleineres, da es die normalisierte Distanz bis zur nächsten zu benut-

zenden Sitzgelegenheit darstellt. Daraus folgt aber auch, dass jede Umwelt als kritisch gilt, bei der nicht sichergestellt ist, dass i stets eine Sitzgelegenheit dort zur Verfügung hat, wo er sich gerade befindet. (Wie oben geschrieben, kann dies, so es dem Leser gedanklich hilft, auch als „*potentiell* kritische Umwelt“ verstanden werden.) Eine Umwelt, bei der i stets eine Sitzgelegenheit dort zur Verfügung hat, wo er sich gerade befindet, wäre hier tatsächlich die bestmögliche aller Welten. Solch eine wäre sogar denkbar, etwa durch eine lückenlose Aneinanderreihung von Sitzgelegenheiten entlang des Weges von i (oder einer Sitzgelegenheit, die mit i mitwandert – was aber im vorherigen Abschnitt 3.2.1 formulierten mathematische Modell ausgeschlossen ist, weil die Sitzgelegenheiten fixe, vom Diskursweltzustand unabhängige Positionen haben.) In der Regel werden aber die allermeisten Umweltzustände, in denen sich i befindet kritisch sein, d. h. $y_i(\tau) > 0$. Safety-Vorfälle müssen also i. d. R. dadurch verhindert werden, dass der Personenzustand unkritisch, d. h. $x_i(\tau) < 1$, gehalten wird. Denn es kommt nur zu einem Safety-Vorfall, wenn $y_i(\tau) > 0$ *und* (gleichzeitig) $x_i(\tau) \geq 1$.

Ein Safety-Vorfall liegt genau dann vor, wenn $z_i(\tau) < 0$ ist. Aufgrund des Ausdrucks 3.14 gilt hierfür

$$z_i(\tau) < 0 \Leftrightarrow x_i(\tau) + y_i(\tau) > 1 \ . \tag{3.16}$$

Hierfür sind wiederum die drei durch die Ausdrücke 3.17 – 3.19 formulierten drei Fälle zu unterscheiden. Zum einen gilt Ausdruck 3.16, wenn $x_i(\tau)$ und $y_i(\tau)$ beide einen als kritisch bezeichneten Wert haben. Dies stellt dann einen sog. *unmittelbaren Safety-Vorfall* dar:

$$x_i(\tau) \geq 1 \wedge y_i(\tau) > 0 \Rightarrow z_i(\tau) < 0 \ . \tag{3.17}$$

Der Ausdruck 3.16 kann aber auch für ein unkritisches $x_i(\tau) < 1$ erfüllt sein, wenn dann $y_i(\tau)$ entsprechend hoch ist. Zum Bsp. sei $d_i^{max} = 500\,\mathrm{m}$, die bereits von i gegangene Distanz seit seiner letzten Pause $d_i^{went} = 450\,\mathrm{m}$ und die Distanz bis zur nächsten zu benutzenden Sitzgelegenheit $d_i^{togo} = 100\,\mathrm{m}$. Dann ist $x_i(\tau) = \frac{450\,\mathrm{m}}{500\,\mathrm{m}} = 0,9 < 1$, aber $z_i(\tau) = 1 - (\frac{450\,\mathrm{m}+100\,\mathrm{m}}{500\,\mathrm{m}}) = -0,1 < 0$. Dieser Fall antizipiert gleichsam einen Safety-Vorfall. Denn wenn der diesen Fall verallgemeinernde Ausdruck 3.18 (s. u.) gilt, dann wird unweigerlich zu einem späteren Diskursweltzustand der Ausdruck 3.17 gelten, z. B., wenn später $d_i^{went} = 530\,\mathrm{m}$ und dann $d_i^{togo} = 20\,\mathrm{m}$, sodass $x_i(\tau) = 1,06 > 1$ und $y_i(\tau) = 0,04 > 0$. Daher stelle die durch den folgenden Ausdruck 3.18 formulierte Implikation einen sog.

antizipierten Safety-Vorfall dar:

$$x_i(\tau) < 1 \wedge y_i(\tau) > 1 - x_i(\tau) \Rightarrow z_i(\tau) < 0 \ . \tag{3.18}$$

Der Ausdruck 3.16 kann genauso auch für ein unkritisches $y_i(\tau) = 0$ erfüllt sein, wenn dann $x_i(\tau)$ entsprechend hoch ist. Zum Bsp. sei $d_i^{max} = 500\,\text{m}$, die bereits von i gegangene Distanz seit seiner letzten Pause $d_i^{went} = 550\,\text{m}$ und die Distanz bis zur nächsten zu benutzenden Sitzgelegenheit $d_i^{togo} = 0\,\text{m}$. Dann ist $z_i(\tau) = 1-(\frac{550\,\text{m}+0\,\text{m}}{500\,\text{m}}) = -0,1 < 0$. In diesem Fall musste unweigerlich zu einem früheren Zeitpunkt in der Diskurswelt mindestens einmal ein unmittelbarer Safety-Vorfall aufgetreten sein. Daher stelle die durch den folgenden Ausdruck formulierte Implikation einen sog. *retrospektiven Safety-Vorfall* dar:

$$y_i(\tau) = 0 \wedge x_i(\tau) > 1) \Rightarrow z_i(\tau) < 0 \ . \tag{3.19}$$

Wenn $z_i(\tau) > 0$, dann ist notwendigerweise $x_i(\tau) + y_i(\tau) < 1$, was nicht möglich ist, wenn ein (unmittelbarer) Safety-Vorfall vorliegt. Wenn $z_i(\tau) > 0$ dann liegt auch weder ein retrospektiver noch ein antizipierter Safety-Vorfall vor. Denn $z_i(\tau) > 0$ bedeutet, dass Passant i ein Sitzgelegenheit benutzt, bevor er ermüdet. Dieser Zustand gilt zwar als unerwünscht, stellt aber keinen Safety-Vorfall dar. Der Zustand gilt aus dem Grunde als unerwünscht, weil dann Passanten zu früh bzw. unnötige Sitzpausen machen, was Sitzkapazitäten „ohne Not verschwendet“. Die Vermeidung unerwünschter Zustände ist in der Zielvorgabe enthalten, weil $z_i(\tau) = 0$ erreicht werden soll. Dies impliziert dann, dass Passant i genau dann eine Sitzpause machen soll, wenn er diese benötigt und zielt somit auf eine i. e. S. *erfordernisgenaue Verfügbarkeit* ab, denn es gilt:

$$z_i(\tau) := 1 - x_i(\tau) + y_i(\tau) \stackrel{!}{=} 0 \tag{3.20}$$

$$\Leftrightarrow \quad x_i(\tau) + y_i(\tau) = 1 \tag{3.21}$$

$$\Leftrightarrow \quad \frac{d_i^{went}}{d_i^{max}} + \frac{d_i^{togo}}{d_i^{max}} = 1 \tag{3.22}$$

$$\Leftrightarrow \quad \underbrace{d_i^{went} + d_i^{togo}}_{\text{Distanz zwischen Sitzpausen}} = \underbrace{d_i^{max}}_{\text{Distanz bis Pausenerfordernis}} \tag{3.23}$$

$z_i(\tau)$ stellt gemäß der entwickelten Safety-Konzipierung das Safety-Maß dar. Es ist insbesondere *quantitativ* und dabei *metrisch* interpretierbar, denn es repräsentiert ein normalisiertes Maß für die Distanz um die eine Sitzgelegenheit für i zu spät ($z_i(\tau) < 1$) oder zu früh ($z_i(\tau) > 1$) verfügbar ist. Nach Denormalisierung durch Multiplikation mit d_i^{max} ist diese Distanz dann in der gewählten Streckeneinheit gegeben. Auf die Darstellung und quantitative Interpretierbar-

keit des Safety-Maßes wird im Abschnitt 5.2 noch einmal eingegangen.

3.2.2.2 Safety-Bewertungsfunktionen

Das Safety-Maß $z_i(\tau)$ für die erfordernisgenaue Verfügbarkeit soll nun so abgebildet werden, dass es zwischen 0 und 1 liegt und umso höher ist, je erstrebenswerter der dahinterliegende Zustand ist. Dazu wird im Folgenden die Safety-Bewertungsfunktion $s_{(a)}(\cdot) \in [0;1]$ konstruiert. Das „$\cdot$“-Symbol steht hier zunächst als ein Platzhalter für Parameter der Funktion. Denn für die folgende Entwicklung einer Safety-Bewertungsfunktion wird $z_i(\tau)$ als Parameter verwendet, obwohl die Funktion $s_{(a)}$ mit den Parametern (j, i, B) eingeführt wurde. Der Zusammenhang zwischen $z_i(\tau)$ und (j, i, B) als Parameter für $s_{(a)}$ klang oben bereits an und besteht darin, für $z_i(\tau)$ ausschließlich solche Diskursweltzustände τ zu betrachten, in welchen Passant i gerade eine Sitzgelegenheit j passiert, die er gemäß B benutzen soll. Die Überführung lässt sich wie folgt beschreiben:

- Rufe $s_{(a)}(j, i, B)$ nur für Sitzgelegenheiten j auf, die Passant i benutzen soll.
- Gehe davon aus, dass sich i direkt bei j befindet.
- Setze folglich $d_i^{togo} = 0$.
- Schlage in B die Sitzgelegenheit j' der vorherigen Pause von Passant i nach. Wenn i vor j noch keine Pause machte, dann setze j' auf den Startwegpunkt von i.
- Dann ist $d_i^{went} = |pos(j) - pos(j')|$.

Da d_i^{max} bekannt ist, lässt sich der Wert von $z_i(\tau)$ für die spezifizierten Diskursweltzustände aus den spezifizischen Parametern (i, j, B) direkt berechnen, wobei τ dann, wie beschrieben, ein Diskursweltzustand ist, in welchem sich Passant i gerade bei der Sitzgelegenheit j befindet. Um in den nächsten Abschnitten notationsbedingte Unübersichtlichkeit zu vermeiden, wird $z_i(\tau)$ einfach als z notiert. Es ist für die Konstruktion der Safety-Bewertungsfunktion unwesentlich, dass sich das Safety-Maß auf einen bestimmten Passanten i und einen bestimmten Diskursweltzustand τ bezieht.

Es gelte $s_{(a)}(z) = 1$ für $z = 0$. Links und rechts von $z = 0$ sollen die Werte $s_{(a)}(z)$ mit der Entfernung von der Stelle $z = 0$ abnehmen, d. h. $s_{(a)}(z)$ soll für $z < 0$ streng monoton steigend und für $z > 0$ streng monoton fallend sein.[32] Dabei muss $z = 0$ nicht unbedingt eine Spiegelachse darstellen. Es ist

[32] Die Safety-Funktion $s_{(a)}$ repräsentiert somit Präferenzen mit Scheitelpunkt (*single-peaked preferences*, vgl. Thomson, 2011, S. 476 ff.).

sogar ausdrücklich beabsichtigt, dass die Funktion $s_{(a)}(z)$ auf der linken Seite anders verlaufen kann als auf der rechten Seite. Denn die linke Seite stellt mit $z < 0$ Safety-Vorfälle dar, während die rechte Seite mit $z > 0$, unerwünschte Zustände darstellt, die aber keine Safety-Vorfälle sind.

Es gelte für den Verlauf von $s_{(a)}(z)$ links und rechts von $z = 0$ jedoch gleichermaßen das Folgende: (I) Es gibt keine Nullstelle. Zusammen mit der Forderung, dass $s_{(a)}$ *streng* monoton abnehmend in Richtung $\pm\infty$ sein soll (d. h. für $z < 0$ streng monoton steigend und für $z > 0$ streng monoton fallend) bedeutet dies schon rein mathematisch, dass sich die Funktion links und rechts asymptotisch dem Wert 0 nähert, diesen aber niemals annimmt. In der Sache ist dies dadurch begründet, dass z als normalisiertes Distanzmaß betragsmäßig unendlich groß werden kann. Die asymptotische Annäherung an den Wert 0 geht mit der beabsichtigten Eigenschaft einher, dass die Grenzrate der Bewertung, d. h. das Differential von $s_{(a)}$ nach z, für z gegen $\pm\infty$ betragsmäßig abnimmt. Dies bedeutet, je größer z betragsmäßig ist, d. h. je größer die Abweichung zwischen dem Ort der Verfügbarkeit einer Sitzgelegenheit und dem Ort der Erfordernis einer Sitzgelegenheit schon ist, desto weniger gravierend wird eine noch höhere Abweichung bewertet. Dies erscheint sinnvoll, da es für einen Passanten so gut wie unerheblich sein dürfte, ob eine Abweichung 1000 m oder 1010 m beträgt, während er den Unterschied zwischen einer Abweichung von 10 m und 20 m als gravierend empfinden dürfte.

(II) Genauso sollen geringe Abweichungen von $z = 0$ unterproportional „bestraft“ werden, sodass die Grenzrate von $s_{(a)}(z)$ auch für z gegen 0 betragsmäßig abnimmt. Bei $z = 0$ soll die Grenzrate sogar 0 sein, d. h. infinitesimal kleine Abweichungen vom erstrebenswerten Zustand so bewertet werden, wie der erstrebenswerte Zustand $z = 0$ selbst. Auch nicht infinitesimal kleine, aber kleine Abweichungen dürften Passanten als unerheblich empfinden; ob eine Sitzpause z. B. 1 m früher oder später möglich ist, dürfte kaum eine Rolle spielen.

Für die soeben geforderten Eigenschaften kommen sog. Sigmoid-Funktionen infrage, d. h. Funktionen mit S-förmigem Verlauf, mit denen dann $s_{(a)}(z)$ abschnittsweise konstruiert wird. Es sollen zwei Sigmoid-Funktionen gebildet werden, eine für den linken Definitionsbereich mit $z < 0$ und eine für den rechten Definitionsbereich mit $z > 0$ (wo die Funktion dann einen umgekehrt S-förmigen Verlauf hat). An der Nahtstelle, d. h. am rechten Definitionsrand des linken Abschnitts und am linken Definitionsrand des rechten Abschnitts, sollen die Teilfunktionen jeweils die Grenzrate 0 und den Wert 1 haben. Es gibt mehrere Funktionsvorschriften für Sigmoid-Funktionen – z. B. die logistische Funktion – allerdings nehmen diese i. d. R. nirgends eine Grenzrate von 0 an. Für den rechten und linken Definitionsbereich wird daher jeweils eine halbe Gaußsche Glockenfunktion verwendet. Denn diese erfüllt alle o. g. Forderun-

gen. Der Basisausdruck lautet:

$$\underbrace{\left(\sqrt{2\pi w^2}\cdot\exp\left(\frac{z^2-0}{2w^2}\right)\right)^{-1}}_{\text{Gaußsche Glockenfunktion (ohne Abszissenverschiebung)}}\cdot\underbrace{w\sqrt{2\pi}}_{\text{Streckungsfaktor}} \tag{3.24}$$

$$=\exp\left(-\frac{z^2}{2w^2}\right) \tag{3.25}$$

Dabei ist π die Kreiszahl, w ein Parameter für die Wendestellen, d. h. für die Stellen, an denen die Grenzrate von $s_{(a)}(z)$ aufhört zu steigen und beginnt zu sinken bzw. v. v. und exp die Exponentialfunktion zur Basis der Eulerschen Zahl. Der Streckungsfaktor $w\sqrt{2\pi}$ sorgt dafür, dass die Funktion ihren Hochpunkt stets bei $(0\,|\,1)$ hat. Die Wendestellen liegen bei $\pm w$.

Dieser Ausdruck soll so parametrisiert werden, dass die dadurch formulierte Funktion durch bestimmte Punkte $(z_{li}\,|\,s_{li})$ und $(z_{re}\,|\,s_{re})$ verläuft, wobei $z_{li}<0$ und $z_{re}>0$. Außerdem muss $0<s_{li}<1$ und $0<s_{re}<1$ gelten. Für gegebenes z_{li} und s_{li} bzw. z_{re} und s_{re} ergeben sich dann die linke Wendestelle w_{li} bzw. die rechte Wendestelle w_{re} durch

$$w_{li}=\frac{|z_{li}|}{\sqrt{-\ln(s_{li}^2)}}\quad\text{bzw.}\quad w_{re}=\frac{|z_{re}|}{\sqrt{-\ln(s_{re}^2)}}\;. \tag{3.26}$$

Die Wendestellen dienen als Parameter für die nun wie folgt abschnittsweise definierte Safety-Funktion bzgl. erfordernisgenauer Verfügbarkeit:

$$s_{(a)}(z)=\begin{cases}\exp\left(-z^2\cdot\frac{1}{2}(w_{li})^{-2}\right) & ,\text{für } z<0\\ 1 & ,\text{für } z=0\\ \exp\left(-z^2\cdot\frac{1}{2}(w_{re})^{-2}\right) & ,\text{für } z>0\end{cases} \tag{3.27}$$

(Potenzschreibweise für Brüche aus Gründen der Lesbarkeit.)

Abbildung 3.7 zeigt eine beispielhafte so entstehende Safety-Funktion, die durch die Punkte $(-1\,|\,0,2)$ und $(1\,|\,0,75)$ parametrisiert wurde.

Nachdem das Safety-Maß für die bedarfsgenaue Verfügbarkeit metrisch und darauf basierend die zugehörende Safety-Bewertungsfunktion bestimmt ist, fehlt noch eine Bestimmung der Safety-Funktion $s_{(b)}(\cdot)\in[0;1]$ für (b) Adaptivität bzgl. Höhe der Sitzfläche. Diese Funktion wird ebenfalls den zugrundeliegenden Safety-Zusammenhang quantifizeren, dass ein Safety-Vorfall genau dann auftritt, wenn ein kritischer Personenzustand und ein kritischer Umwelt-

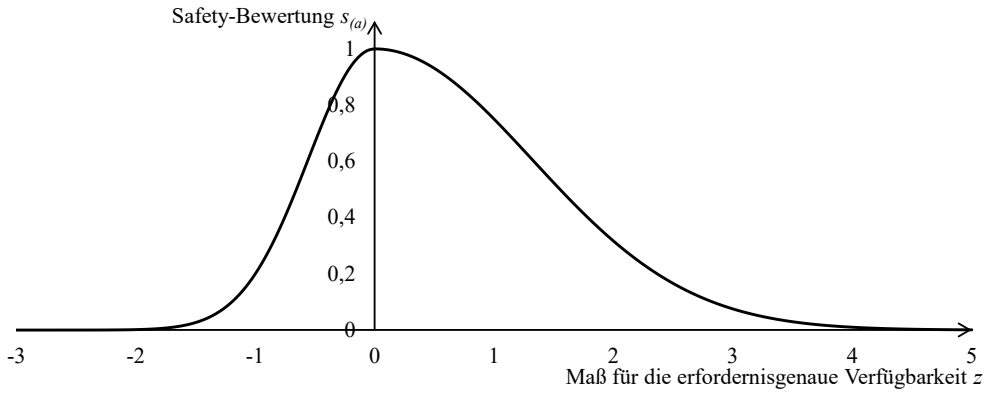


Abbildung 3.7: Safety-Funktion als abschnittsweise auf der Gaußschen Glockenfunktion basierend definierte Sigmoid-Funktionen

zustand zusammenkommen. Während bei (a) Adaptivität bzgl. Verfügbarkeit der Umweltzustand so gut wie immer kritisch ist, wird nun angenommen, dass der Personenzustand aufgrund motorischer Beeinträchtigungen immer kritisch ist. Ein Safety-Vorfall wird hier verhindert, indem der kritische Umweltzustand durch aktive Hinsetz- und Aufstehassistenz vermieden wird.

Der Bestimmung der Safety-Funktion für Adaptivität bzgl. Höhe der Sitzfläche wird allerdings kein metrisches Maß zugrunde liegen, sondern eine begründete Festlegung numerischer Bewertungen für die vier möglichen Fälle, dass ein Passant i bei Benutzung einer Sitzgelegenheit j (I) weder Hinsetz- noch Aufstehassistenz, (II) nur die Hinsetzassistenz, (III) nur die Aufstehassistenz und (IV) sowohl Hinsetz- als auch Aufstehassistenz nutzen kann. Die Randfälle (I) und (IV) werden auf die Extremwerte 0 und 1 abgebildet. Für die Fälle (II) und (III) sind dann Zwischenwerte $\gamma^{Hin}, \gamma^{Auf} \in [0;1]$ festzulegen, wobei γ^{Hin} die Bewertung darstellt, wenn nur Hinsetzassistenz, γ^{Auf} die Bewertung, wenn nur Aufstehassistenz nutzbar ist, und es gelte:

$$\gamma^{Hin} + \gamma^{Auf} = 1 \ . \tag{3.28}$$

Hinsetzassistenz ist nutzbar, wenn bei Ankunft von Passant i bei Sitzgelegenheit j kein anderer Passant auf der Sitzgelegenheit sitzt. Aufstehassistenz ist nutzbar, wenn bei Abgang von Passant i von Sitzgelegenheit j kein anderer Passant auf der Sitzgelegenheit sitzt. Oben wurde das „$\cdot$“-Symbol als Platzhalter für die Parameter der Funktion $s_{(b)}$ verwendet, obwohl diese Funktion – so wie $s_{(a)}$ auch – mit den Parametern (j, i, B) eingeführt wurde. Die Verwendung eines gleichen Parametertripels für $s_{(a)}$ und $s_{(b)}$ hat einen beabsichtigten

Sinn, der weiter unten klar gemacht wird. Zur Spezifizierung der Funktion sollen aber andere Parameter verwendet werden, die aus dem Tripel (j, i, B) wie folgt überführt werden können:

- Sei τ ein Diskursweltzustand, in welchem Passant i sich gemäß der Belegung B gerade auf die Sitzgelegenheit j hinsetzt.
- Sei τ' der Diskursweltzustand, in welchem Passant i wieder von der Sitzgelegenheit j aufsteht.
- Sei $\underline{occ} = \text{card}\left(occ_j(\tau) \setminus \{p_i\}\right)$, d. h. die Anzahl der Passanten ohne Passant i, die j gerade gemäß B benutzen, wenn sich i auf j hinsetzt.
- Sei $\overline{occ} = \text{card}\left(occ_j(\tau') \setminus \{p_i\}\right)$, d. h. die Anzahl der Passanten ohne Passant i, die j gerade gemäß B benutzen, wenn i wieder von j aufsteht.

Sei die Safety-Funktion für die Nuzbarkeit der (b) Adaptivität bzgl. Höhe der Sitzfläche nun wie folgt spezifiziert:

$$s_{(b)}(\underline{occ}, \overline{occ}) = \begin{cases} 0 & \text{, für } \underline{occ} > 0 \wedge \overline{occ} > 0 \\ \gamma^{Hin} & \text{, für } \underline{occ} = 0 \wedge \overline{occ} > 0 \\ \gamma^{Auf} & \text{, für } \underline{occ} > 0 \wedge \overline{occ} = 0 \\ 1 & \text{, für } \underline{occ} = 0 \wedge \overline{occ} = 0 \end{cases} \tag{3.29}$$

In Ausdruck 3.7 wurde die zusammengesetzte Safety-Funktion definiert, welche sowohl (a) Adaptivität bzgl. Verfügbarkeit mit $s_{(a)}$ als auch (b) Adaptivität bzgl. Höhe der Sitzfläche mit $s_{(b)}$ berücksichtigt. Sinnvollerweise müssen sich dann $s_{(a)}$ und $s_{(b)}$ auf die gleiche Sitzgelegenheit beziehen. Aus diesem Grunde ist die zusammengesetzte Safety-Funktion für die Eingabe (j, i, B) definiert, welche auch die Argumente für $s_{(a)}$ und $s_{(b)}$ darstellt, aus denen dann wie oben beschrieben $z_i(\tau)$ sowie $\underline{occ}$ und $\overline{occ}$ bestimmt werden. Die zusammengesetzte Safety-Bewertungsfunktion $s(j, i, B)$ gewichtet dann die beiden (Teil-) Safety-Funktionen $s_{(a)}$ und $s_{(b)}$ gegeneinander, die jeweils für dieselbe Sitzgelegenheit j, denselben Passanten i und dieselbe Belegung B ausgewertet werden.

Prinzipiell könnten die Gewichte $\lambda_{(a)}$ und $\lambda_{(b)}$ der zusammengesetzten Safety-Funktion für jeden Passanten individuell bestimmt werden. Genauso könnten prinzipiell auch γ^{Hin} und γ^{Auf} für jeden Passanten individuell bestimmt werden. Hier wird aber davon ausgegangen, dass diese Parameter über alle Individuen hinweg gleich sind. Dies ist vergleichbar mit der Argumentation von Piacquadio (2017), dass eine Quantifizierung des Nutzens durch eine akzeptierte gemeinsame Kardinalisierungsfunktion – hier durch die Safety-Funktion – erfolgen kann, die eine Art Kompromiss zwischen den Individuen darstellt. Eine

andere Argumentation ist, davon auszugehen, dass diese Parameter wie anthropologische Konstanten behandelt werden, welche sich zwischen Individuen nicht wesentlich unterscheiden. (Im Übrigen könnte auch die Sitzplatzkapazität κ für jede Sitzgelegenheit j unterschiedlich festgelegt werden, worauf aber unter der Annahme gleichartiger SSOs verzichtet wird.)

Mit Ausdruck 3.8 wurde bereits formuliert, wie die Safety-Gesamtbewertung $\hat{s}(i, B)$ für einen Passanten bezogen auf die Diskurswelt, d. h. bezogen auf die Strecke mit den k Sitzgelegenheiten, bestimmt wird. Offen ist noch die Bestimmung der aggregierten Safety-Gesamtbewertung $agg(\hat{\mathbf{s}})$ $\big($wobei der Vektor $\hat{\mathbf{s}}$ die n Komponenten $\hat{s}_i = \hat{s}(i, B)$ hat$\big)$. Für die Aggregation kommen verschiedene „herkömmliche“ Möglichkeiten in Betracht, etwa Mittelwertsbildung oder Aufsummierung. Die Bildung und Maximierung des arithmetischen Mittelwertes oder der Summe würde z. B. bei zwei Passanten eine Allokation, bei der $\hat{s}_1 = 0,1$ und $\hat{s}_2 = 0,9$ ist, einer Allokation vorziehen, bei der $\hat{s}_1 = 0,4$ und $\hat{s}_2 = 0,5$. Dies erscheint aber ungeeignet, weil bei der dann bevorzugten Allokation der sehr hohe für den 2. Passanten realisierte Safety-Wert auf Kosten des sehr niedrigen realisierten Safety-Wertes für den 1. Passanten zu gehen scheint. Um dies zu vermeiden, lautet die aggregierte Safety-Gesamtbewertung, welche gemäß Ausdruck 3.10 möglichst hoch werden soll, mit Verweis auf das Postulat 1 und seine Begründung:

$$agg(\hat{\mathbf{s}}) = \min_i \hat{s}_i \ . \tag{3.30}$$

3.3 Lösungsverfahren

3.3.1 Zustandsraumexploration

Bis hierhin wurde das Problem zwar formal beschrieben, aber es ist noch offen, *wie* es *gelöst* werden soll. In diesem Abschnitt soll nun dargelegt werden, wie eine (im Abschnitt 2.7.1 beschriebene) Baumstruktur erzeugt und genutzt werden kann, um ein $B \in \mathbb{B}^{n \times k}$ gemäß den Ausdrücken 3.9 – 3.11 zu finden. Die Erzeugung der Baumstruktur erfolgt dabei schrittweises durch Expandieren einzelner Diskursweltzustände. Die durch Expandieren erzeugten Diskursweltzustände bilden dann einen Zustandsbaum, die den Zustandsraum aufspannen. Mit Zustandsraumexploration ist das Traversieren dieses Zustandsbaumes gemeint. Die Exploration besteht i. d. R. aus sehr vielen Expansionen. Es sei begrifflich definiert:

Definition 3 (Vollständiger Pfad und Zielknoten) *Ein vollständiger Pfad ist ein Pfad im Zustandsbaum, der genau $n \cdot k + 2$ Knoten hat. Ein vollständiger*

Pfad endet in einem Zielknoten. Ein Zielknoten repräsentiert einen Diskursweltzustand, in welchem alle Passanten die Strecke bewältigt haben.

Alle Zielknoten sind Blätter und alle Blätter sind auf der gleichen Tiefe. In einem vollständigem Pfad passiert jeder der n Passanten jede der k Sitzgelegenheiten, wobei jedes dieser *Passierereignisse* einen Knoten definiert (und sinnvollerweise $n, k > 0$ ist). Zu diesen $n \cdot k$ Knoten kommen der Start- und die Zielknoten hinzu. Der Startknoten repräsentiert einen initialen Diskursweltzustand und hat keinen Vorgänger. Auf dem Startknoten wird erstmalig die Expansionslogik angewendet, die zum ersten Passierereignis führt. Der Startknoten selbst repräsentiert kein Passierereignis und hat stets genau das erste Passiereignis als Nachfolger. Neben dem Start- und den Zielknoten werden lediglich solche Diskursweltzustände als Knoten repräsentiert, in denen ein Passant eine Sitzgelegenheit passiert, weil sich nur dann der Fortlauf der Diskurswelt verzweigen kann.

Definition 4 (Verzweigungsknoten und -konvention) *Ein Verzweigungsknoten repräsentiert ein Passiereignis, d. h. das Ereignis, dass (irgend-) ein Passant i (irgend-) eine Sitzgelegenheit j passiert.*
Wenn sich der Passant i auf die Sitzgelegenheit j hinsetzt, wird der Pfad im linken Nachfolger des Verzweigungsknoten fortgesetzt.
Wenn sich der Passant i nicht auf die Sitzgelegenheit j hinsetzt, sondern weitergeht, wird der Pfad im rechten Nachfolgerknoten fortgesetzt.

Der linke Nachfolgerknoten eines Verzweigungsknoten kann *null*, d. h. nicht vorhanden sein, sodass der linke Pfad dort abbricht. Dies ist der Fall, wenn die Sitzgelegenheit, welche der Passant passiert, bereits vollbesetzt ist. Dieses Abbrechen stellt dann *per Konstruktion* sicher, dass, bezogen auf einen vollständigen Pfad, $\xi_\kappa(B) = 1$ gilt. In einem vollständigen Pfad repräsentieren die Verzweigungen an den Verzweigungsknoten eine Belegung $B \in \mathbb{B}^{n \times k}$: Wenn ein Verzweigungsknoten nach links verzweigt, dann ist $b_{i,j} = 1$, wobei i der Passant ist und j die Sitzgelegenheit, welche das Ereignis für den Verzweigungsknoten konstituieren. Für rechte Verzweigungen gilt dann entsprechend $b_{i,j} = 0$. Da für Verzweigungsknoten immer mindestens ein rechter Nachfolgerknoten exisitiert, sind alle Blätter Zielknoten (und repräsentieren nicht Passierereignisse).

In der Expansionslogik für die Verzweigungen, können auch die Informationen für die Safety-Funktionen bestimmt werden. Die Expansionslogik stellt den Kern zur Exploration der Diskurswelt unter Berücksichtigung der Nebenbedingungen dar. Ausgangspunkt der Exploration ist der Startknoten, welcher den initialen Diskursweltzustand repräsentiert. Dieser Startknoten wird zunächst geklont, der Klon unter den originalen Startknoten gehängt und dann

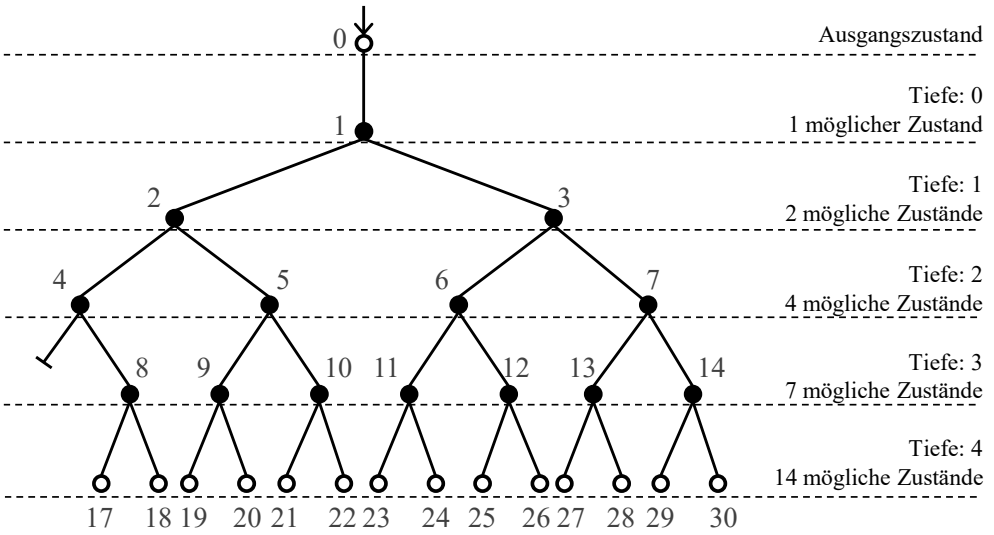


Abbildung 3.8: Beispiel-Zustandsraumstruktur für das zu lösende Problem

auf dem Klon erstmalig die Expansionslogik angewendet. Dies wird deshalb so gemacht, weil die Anwendung der Expansionslogik auf einen Knoten diesen Knoten so manipuliert, dass er danach entweder ein Verzweigungsknoten oder der Zielknoten ist. Der Startknoten soll aber nicht so manipuliert werden (und stellt insbesondere kein Passiereignis dar). Bevor die Expansionslogik genauer expliziert wird, soll ein Blick auf die Abbildung 3.8 dem Grundverständnis der Baumstruktur für die vorliegende Arbeit dienen:[33]

Der obere nicht ausgefüllte Kreis (Knoten mit der Nr. 0) stellt den Startknoten mit dem initialen Zustand dar. Ausgefüllte Knoten stellen die Verzweigungsknoten dar. Da der Startknoten niemals einen Verzweigungsknoten (oder einen Zielknoten) repräsentieren soll, wird der Knoten Nr. 0 dupliziert und als Knoten Nr. 1 darunter gehängt. Knoten Nr. 1 repräsentiert also zunächst noch exakt den gleichen Zustand, wie Knoten Nr. 0, der kein Passiereignis darstellt. Durch die erstmalig auf Knoten Nr. 1 angewendete Expansionslogik wird das erste Passierereignis ermittelt (das existiert, wenn $n, k > 0$). Die Zustandsinformationen von Knoten Nr. 1, wie die Positionen der Passanten usw., werden dabei bis zu diesem Passierereignis fortgeschrieben, was die Zustandsinformationen überschreibt, die von Knoten Nr. 0 übernommen wurden. Die nicht ausgefüllten Knoten ganz unten (Nr. 17 bis 30) sind Zielknoten. Diese liegen gemäß der Definition 3 alle auf der Tiefe $n \cdot k$. (Die Wurzel hat die Tiefe 0 und der Ausgangszustand wird nicht zur Tiefe mitgezählt.) Während der Expan-

[33] Vergleiche, auch im Weiteren, Hubl (2019).

sion von der Wurzel zu den Zielknoten werden die Safety-Funktionen ausgewertet, sodass in den Zielknoten jeweils die Safety-Gesamtbewertung ermittelt werden kann. Damit kann dann unter den durch die Zielknoten repräsentierten möglichen Belegungen – im Beispiel: 14 Stück – eine Belegung gemäß der Safety-Zielfunktion bestimmt werden.

Es sei explizit darauf hingewiesen, dass in der Abbildung 3.8 nicht alle Informationen visualisiert sind, die in der Baumstruktur enthalten sind. Die zentrale Information etwa, welcher Passant welche Sitzgelegenheit in einem Verzweigungsknoten passiert, ist nicht ablesbar. Dies ist für die grundsätzliche Erläuterung, wie die Baumstruktur für die vorliegende Problemstellung zu interpretieren ist, aber auch nicht unbedingt erforderlich. Die Abbildung 3.8 stellt z. B. den (bereits vollständig erzeugten) Baum für eine Diskurswelt dar, in der $n = 2$ Passanten eine Strecke zurücklegen, auf der sich $k = 2$ Sitzgelegenheiten befinden. Der Knoten Nr. 4 repräsentiert dann bspw. das Ereignis, dass ein Passant an einer Sitzgelegenheit vorbeikommt, an der sich zuvor bereits zwei andere Passanten hinsetzten, die Sitzgelegenheit zwei Sitzplätze hat und die zwei Passanten, die sich zuvor darauf setzten, immer noch auf dieser Sitzgelegenheit sitzen. Der Passant, der im Zustand Nr. 4 an dieser Sitzgelegenheit vorbeikommt, kann sich dann also nicht dort hinsetzen. Die linke Verzweigungsmöglichkeit, die gemäß der in Definition 4 festgelegten Konvention die Alternative repräsentiert, dass sich der passierende Passant auf die passierte Sitzgelegenheit hinsetzt, wird deshalb durch die Expansionslogik abgebrochen.

Die Grundidee der Expansionslogik kann informal wie folgt in vier Schritten beschrieben werden:

1. Ausgehend vom gegebenen Diskursweltzustand, bestimme die Dauer bis irgendein Passant als nächstes eine Sitzgelegenheit passiert und memoriere den passierenden Passanten sowie die passierte Sitzgelegenheit. Die bestimmte Dauer wird im Folgenden auch als der Zeithorizont für die Expansion bezeichnet.
2. Für jeden Passanten, der die Strecke noch nicht bewältigt hat: Aktualisiere seine Position und seinen Ermüdungsgrad für den Zeithorizont. Beachte dabei, ob der Passant über die gesamte Dauer des Zeithorizontes sitzt, die ganze Zeit geht oder im Zeithorizont erst sitzt und dann geht. (Im letzten Fall ist der Bewegungszustand des Passanten von *sitzend* zu *gehend* zu ändern.)
3. Wenn der 1. Schritt ergibt, dass kein Passant mehr eine Sitzgelegenheit passiert, dann führe den 2. Schritt statt mit dem Zeithorizont mit der Dauer aus, die noch nötig ist, bis alle Passanten die Strecke bewältigt

haben. (Danach sind alle Passanten im Bewegungszustand *Strecke bewältigt*).

4. Wenn noch nicht alle Passanten die Strecke bewältigt haben, dann erstelle eine linke und einen rechte Verzweigung. Wenn die passierte Sitzgelegenheit keine freie Sitzplatzkapazität hat, dann setze den linken Verzweigungsknoten auf *null*. Wenn die passierte Sitzgelegenheit noch freie Kapazität hat, dann platziere den passierenden Passanten auf ihr und setze den Bewegungszustand des Passanten auf *sitzend*. Für den rechten Verzweigungsknoten ist nichts zu ändern, da hier der Passant einfach weitergeht.

Formal ist die Expansionslogik durch die Algorithmen 2 – 5 beschrieben. Diese vier Algorithmen bilden im Grunde eine einzige Prozedur, die hier in vier Sinnabschnitte unterteilt ist. Die Prozedur beginnt im Algorithmus 2 mit der Eingabe (*input*) und endet im Algorithmus 5 mit der Ausgabe (*return*). Die Algorithmen 2 – 5 sind also im Zusammenhang zu lesen. Zum Bsp. werden Werte von Variablen, die in Algorithmus 2 bestimmt werden, in Algorithmus 5 verwendet (etwa bei i^* und j^*). Die Algorithmen 2 – 5 beschränken sich auf das Wesentliche zur Beschreibung der Expansionslogik. Für alle in dieser Arbeit beschriebenen Algorithmen (1 – 7) findet sich kommentierter Quellcode für eine ausführbare Java-Implementierug im Anhang (S. 147 ff.).[34]

Algorithmus 2 bestimmt die als Zeithorizont bezeichnete Dauer ab einem Diskursweltzustand τ, bis ein Passant eine Sitzgelegenheit passiert. Die Eingabe τ ist dabei ein Zustandknoten, der bis zum nächsten Passierereignis fortgeschrieben werden soll. Wenn ab τ kein Passant mehr eine Sitzgelegenheit passiert, dann bleibt der Zeithorizont, wie zu Beginn des Algorithmus festgelegt, „unendlich“. In Zeile 9 wird der nächste Wegpunkt j in Gehrichtung von Passant i ausgewählt: $\{j \mid dir_i \cdot pos(j) > dir_i \cdot pos_i(\tau)\}$ ist genau die Menge der Wegpunkte, die i noch nicht passierte. Hieraus ist das j zu wählen, welches den geringsten Abstand zur aktuellen Position von i hat. Da durch die Bedingung in Zeile 8 sichergestellt ist, dass der Passant i die Strecke noch nicht bewältigt hat, existiert stets mindestens ein Wegpunkt in Gehrichtung von i, nämlich der Abgrenzungswegpunkt. Die Bedingung in Zeile 10 schließt die Abgrenzungswegpunkte aus der Ermittlung des Zeithorizontes aus. Der *Zeithorizont* ist die kürzeste der Dauern bis eine Passant eine Sitzgelegenheit passiert. Der zur Berechnung der *verbleibenden Verweildauer* in Zeile 4 benötigte *Hinsetzzeitpunkt von i*, wenn Passant i in τ noch sitzt, wird in Algorithmus 5 erfasst (der Passant *muss* sich bei einer vorherigen Diskursweltverzweigung

[34]Inklusive der in Kap. 4 beschriebene Simulationslogik für die Validierung.

hingesetzt haben). Der *Zeitpunkt*(τ) ist der aktuelle Diskurswelt-Zeitpunkt des übergebenen Diskursweltzustandes τ.

Algorithmus 3 beinhaltet die eigentliche Fortschreibung der Diskurswelt für die allermeisten Fälle, dass es noch mindestens ein Passiereignis gibt und der *Zeithorizont* folglich nicht „unendlich“ ist. Die Sortierung der Passanten in Zeile 2 erfolgt, damit die Passanten in Zeile 7 in der richtigen Reihenfolge ggf. von der Sitzgelegenheit aufstehen, weil dies in der folgenden Zeile 8 zur Auswertung der Safety-Funktion für die Nutzbarkeit der Aufstehassistenz maßgebend ist. Mit Auswertung der Safety-Funktion für die Nutzbarkeit der Aufstehassistenz ist gemeint, dass ermittelt wird, ob andere Passanten auf der Sitzgelegenheit j sitzen, wenn i von dort aufsteht. Zusammen mit der Auswertung für die Nutzbarkeit der Hinsetzassistenz, die in Algorithmus 5 ermittelt wird, kann der Wert für $s_{(b)}(j, i, B_\tau)$ vollständig bestimmt werden. B_τ gibt dabei die bis zum Diskurswelt τ durch die Explorationsschritte bestimmte Belegung an. In Zeile 18 wird der Abgrenzungswegpunkt j_i^{end} in Abhängigkeit der Gehrichtung von i bestimmt und ist entsprechend 0 oder $k+1$. Die Zeile 19 prüft dann unter Berücksichtigung der Gehrichtung $dir_i \in \{-1; +1\}$, ob Passant i den Abgrenzungswegpunkt erreicht hat. Die Auswertung von $s_{(a)}(j_i^{end}, i, B_\tau)$ in Zeile 21 wird für den zusätzlichen Summanden im Zähler von Ausdruck 3.8 benötigt. In der Zeile 23 wird die Distanz kumuliert, die Passant i im Diskursweltzustand τ mit einem Pausenerfodernis zurücklegen musste. Diese Information wird später für das heuristische Lösungsverfahren verwendet (siehe Abschnitt 3.3.2). Die Berechnung der im Zeithorizont mit Pausenerfordernis zurückgelegten Distanz mit $\frac{\max\left(f_i(\tau)-1\ ;\ 0\right)}{\dot{f}_i} \cdot v_i$ macht sich zunutze, dass der Ermüdungsgrad, wenn er durch die Berechnung in Zeile 13 oder Zeile 16 größer als 1 wurde, erst in der Zeile 24 auf 1 herabgesetzt wird. $\frac{f_i(\tau)-1}{\dot{f}_i}$ entspricht bei der Berechnung also genau der Dauer, die Passant i im Zeithorizont mit einem Pausenerfordernis zurücklegte. Es sei darauf hingewiesen, dass die Expansionslogik den in Algorithmus 2 übergebenen Diskursweltzustand τ manipuliert und Funktionen mit τ als Argument so zu verstehen sind, dass sie ein Attribut von τ zurückgeben. Die Zuweisung *Zeitpunkt*(τ) := *Zeitpunkt*(τ) + *Zeithorizont* bedeutet daher z. B., dass zum Zeitpunkt, mit dem der Diskursweltzustand τ zur Fortschreibung übergeben wurde, die Dauer des Zeithorizontes addiert wird und dies dann der Zeitpunkt des fortgeschriebenen Diskursweltzustandes τ wird.

Algorithmus 4 beinhaltet die Fortschreibung der Diskurswelt für den Fall, dass der *Zeithorizont* „unendlich“ ist, weil es kein Passiereignis mehr gibt. In diesem Fall wird τ zu einem Zielknoten expandiert. Der wesentliche Unterschied zur Fortschreibung mit Algorithmus 3 liegt darin, dass Algorithmus 4

nicht den *Zeithorizont* zur Fortschreibung zugrundelegt – bzw. nicht zugrundelegen kann –, sondern die *Gehdauer*, bis die Passanten ihren Abgrenzungswegpunkt schließlich erreichen. Da außerdem der durch diesen Algorithmus fortgeschriebene Diskursweltzustand nicht mehr weiter expandiert wird, müssen im Vergleich zu Algorithmus 3 weniger Informationen fortgeschrieben werden. Es ist z. B. auch nicht notwendig, den in Zeile 12 bestimmten Ermüdungsgrad wieder auf 1 herabzusetzen, falls er größer geworden ist.

Algorithmus 5 erstellt die Nachfolgerknoten für den soeben fortgeschriebenen Zustandsknoten τ und gibt sie in Zeile 17 als Ergebnis der Expansion zurück. Dabei bildet τ_{links} den linken und τ_{rechts} den rechten Nachfolgerknoten. Wenn der soeben fortgeschriebene Zustandsknoten τ einen Zielknoten bildet, dann werden in den Zeilen 14 u. 15 τ_{links} und τ_{rechts} jeweils auf *null* gesetzt, sodass τ dann effektiv keine Nachfolger hat. Dazu wird mit der Bedingung in Zeile 1 geprüft, ob τ ein Zielknoten (geworden) ist. Da der *Zeithorizont* die Dauer bis zu einem Passiereignis ist und ein Passierereignis einen Verzweigungsknoten bildet, gilt hier – nach der erfolgten Fortschreibung des Diskursweltzustandes τ:

$$\textit{Zeithorizont} \neq \infty \Leftrightarrow \exists i : q_i(\tau) \neq \textit{Strecke bewältigt} \; . \tag{3.31}$$

Ein Zielknoten liegt andernfalls vor. Wenn ein passierender Passant i^* existiert, dann wird in Zeile 2 geprüft, ob die passierte Sitzgelegenheit j^* schon voll besetzt ist. (Hierbei reicht die Prüfung auf Gleichheit mit κ aus, da per Konstruktion niemals $\text{card}\left(occ_j(\tau)\right) > \kappa$ werden kann.) Wenn j^* voll besetzt ist, dann wird τ_{links} auf *null* gesetzt, da der linke Nachfolgerknoten per Konvention die Diskursweltverzweigung repräsentieren soll, dass sich i^* auf j^* hinsetzt, sich i^* aber nicht auf j^* hinsetzen kann. (Hierdurch wird $\xi_\kappa(B) = 1$ sichergestellt.) Wenn die Sitzgelegenheit j^* noch Kapazität hat, dann wird in Zeile 4 ff. der linke Nachfolgerknoten als Klon des zuvor fortgeschriebenen Diskursweltzustandes τ erstellt. τ_{links} ist also zunächst ein Duplikat von τ, welches in einem späteren Explorationsschritt durch die Anwendung der Expansion fortgeschrieben wird. Gemäß der Verzweigungskonvention wird in τ_{links} noch für die etwaige[35] nächste Exploration der Passant i^* auf die Sitzgelegenheit j^* gesetzt. Der rechte Nachfolgerknoten τ_{rechts} wird in Zeile 12 ebenfalls von τ geklont, bleibt aber unverändert. Im linken Nachfolgerknoten von τ hat sich

[35] Unter Verwendung des im nachfolgenden Abschnitt beschriebenen heuristischen Verfahrens, wird i. d. R. nicht der gesamte Zustandsraum untersucht. Es werden dann nicht alle Verzweigungsknoten expandiert. *Hinweis:* Verzweigungsknoten, die aufgrund unvollständigen Baumaufbaus keine Nachfolger haben, gelten hier nicht als Blätter.

i^* also gerade auf j^* gesetzt und im rechten nicht. Diese Nachfolger τ_{links} und τ_{rechts} werden in einem etwaigen nächsten Expansionsschritt bis zum nächsten Passierereignis fortgeschrieben. (*Hinweis*: Die Verknüpfung der Knoten zu einer Baumstruktur als Datenstruktur ist hier nicht explizit dargestellt.)

Algorithmus 2 Kalkulation des Zeithorizontes für die Expansion

input: τ

1: $Zeithorizont := \infty$
2: **for all** $i \in Passanten$ **do**
3: **if** $q_i(\tau) = sitzend$ **then**
4: $verbleibende\ Verweildauer\ von\ i :=$
$r_i - \Big(Zeitpunkt(\tau) - Hinsetzzeitpunkt\ von\ i\Big)$
5: **else**
6: $verbleibende\ Verweildauer\ von\ i := 0$
7: **end if**
8: **if** $q_i(\tau) \neq Strecke\ bewältigt$ **then**
9: $j := \underset{\{j \mid dir_i \cdot pos(j) > dir_i \cdot pos_i(\tau)\}}{\arg\min} |pos(j) - pos_i(\tau)|$
10: **if** $0 < j < k + 1$ **then**
11: $Gehdauer := \frac{|pos_i(\tau) - pos(j)|}{v_i}$
12: $Gesamtdauer := Gehdauer + verbleibende\ Verweildauer\ von\ i$
13: **if** $Gesamtdauer < Zeithorizont$ **then**
14: $Zeithorizont := Gesamtdauer$
15: Nächster passierender Passant $i^* := i$
16: Nächste passierte Sitzgelegenheit $j^* := j$
17: **end if**
18: **end if**
19: **end if**
20: **end for**

Die durch die Algorithmen 2 – 5 beschriebene Expansionslogik muss noch in eine Explorationslogik eingebettet werden. Die durch Algorithmus 1 beschriebenen Bestensuche stellt eine solche Explorationslogik dar. Die Expansion wird direkt vor der *For-all-Nachfolgerzustände*-Schleife aufgerufen. Bei der Bestensuche handelt es sich, wie in Abschnitt 2.7.2 beschrieben, um eine Meta-Heuristik. Für eine exhaustive, d. h. den Zustandsraum erschöpfende und somit die Belegung B mit *maximal* möglicher aggregierter Safety-Bewertung $agg(\hat{\mathbf{s}})$ findende, Exploration kann die Expansion in eine einfachere Breitensuchenlogik eingebettet werden, siehe Algorithmus 6.

Algorithmus 3 Fortschreibung der Diskurswelt (normale Expansion)

if $Zeithorizont \neq \infty$ **then**
  Sortiere $Passanten$ nach deren *verbleibenden Verweildauern.*
  for all $i \in Passanten$ (in zuvor sortierter Reihenfolge) **do**
    if $q_i(\tau) = sitzend \wedge$
      verbleibende Verweildauer von $i < Zeithorizont$ **then**
      $Aufstehzeitpunkt\ von\ i :=$
        $Zeitpunkt(\tau) + verbleibende\ Verweildauer\ von\ i$
      $j := j$ für das gilt $p_i \in occ_j(\tau)$
      $occ_j(\tau) := occ_j(\tau) \setminus \{p_i\}$
      Werte $s_{(b)}(j, i, B_\tau)$ für Aufstehassistenz aus.
      $q_i(\tau) := gehend$
      $f_i(\tau) := 0$
      $verbleibende\ Gehdauer :=$
        $Zeithorizont - verbleibende\ Verweildauer\ von\ i$
      $pos_i(\tau) := pos_i(\tau) + verbleibende\ Gehdauer \cdot dir_i \cdot v_i$
      $f_i(\tau) := \dot{f}_i \cdot verbleibende\ Gehdauer$
    else if $q_i(\tau) = gehend$ **then**
      $pos_i(\tau) := pos_i(\tau) + Zeithorizont \cdot dir_i \cdot v_i$
      $f_i(\tau) := f_i(\tau) + \dot{f}_i \cdot Zeithorizont$
    end if
    $j_i^{end} = (dir_i + 1) \cdot \frac{1}{2} \cdot (k + 1)$
    if $dir_i \cdot pos_i(\tau) \geq dir_i \cdot pos(j_i^{end})$ **then**
      $q_i(\tau) := Strecke\ bewältigt$
      Werte $s_{(a)}(j_i^{end}, i, B_\tau)$ aus.
    end if
    $d_i^{fatigued}(\tau) := d_i^{fatigued}(\tau) + \frac{\max\left(f_i(\tau) - 1\ ;\ 0\right)}{\dot{f}_i} \cdot v_i$
    $f_i(\tau) := \min\left(f_i(\tau)\ ;\ 1\right)$
  end for
  $Zeitpunkt(\tau) := Zeitpunkt(\tau) + Zeithorizont$
end if

Algorithmus 4 Fortschreibung der Diskurswelt (Abschlussexpansion)

if *Zeithorizont* $= \infty$ **then**
  Sortiere *Passanten* nach deren *verbleibenden Verweildauern.*
  for all $i \in$ *Passanten* (in zuvor sortierter Reihenfolge) **do**
    if $q_i(\tau) =$ *sitzend* **then**
      $j := j$ für das gilt $p_i \in occ_j(\tau)$
      $occ_j(\tau) := occ_j(\tau) \setminus \{p_i\}$
      Werte $s_{(b)}(j, i, B_\tau)$ für Aufstehassistenz aus.
      $f_i(\tau) := 0$
    end if
    $j_i^{end} = (dir_i + 1) \cdot \frac{1}{2} \cdot (k + 1)$
    $Gehdauer := \frac{|pos_i(\tau) - pos(j_i^{end})|}{v_i}$
    $f_i(\tau) := f_i(\tau) + \dot{f}_i \cdot Gehdauer$
    $q_i(\tau) :=$ *Strecke bewältigt*
    Werte $s_{(a)}(j_i^{end}, i, B_\tau)$ aus.
    $d_i^{fatigued}(\tau) := d_i^{fatigued}(\tau) + \frac{\max\left(f_i(\tau) - 1\ ;\ 0\right)}{\dot{f}_i} \cdot v_i$
  end for
end if

Algorithmus 5 Erstellung der Diskursweltverzweigungen

if *Zeithorizont* $\neq \infty$ **then**
  if card $\left(occ_{j^*}(\tau)\right) = \kappa$ **then**
    $\tau_{links} :=$ **null**
  else
    $\tau_{links} :=$ Klon(τ)
    $occ_{j^*}(\tau_{links}) := occ_{j^*}(\tau_{links}) \cup \{p_i\}$
    $q_{i^*}(\tau_{links}) :=$ *sitzend*
    Hinsetzzeitpunkt von $i^* :=$ *Zeitpunkt*(τ_{links})
    Werte $s_{(a)}(j^*, i^*, B_\tau)$ aus.
    Werte $s_{(b)}(j^*, i^*, B_\tau)$ für Hinsetzassistenz aus.
  end if
  $\tau_{rechts} :=$ Klon(τ)
else
  $\tau_{links} :=$ **null**
  $\tau_{rechts} :=$ **null**
end if
return $(\tau_{links}\ ,\ \tau_{rechts})$

Algorithmus 6 Breitensuchenlogik für exhaustive Exploration

input: *Startzustand*
  Reihe den *Startzustand* in eine (leere) *Warteschlange* (FIFO-Liste) ein.
  while *Warteschlange* ist nicht leer **do**
    T ← Erstes Element in *Warteschlange*. Entferne T aus *Warteschlange*.
    Expandiere T. ▷ Aufruf von Algorithmus 2 ff. → (τ_{links} , τ_{rechts})
    for all Nachfolgerzustände τ von T **do**
      if $\tau \neq$ **null then**
        Reihe τ in die *Warteschlange* ein.
      end if
    end for
  end while
  return Zielzustand mit maximaler Bewertung.
    ▷ Das Optimum wird gefunden.

3.3.2 Heuristisches Lösungsverfahren

3.3.2.1 Meta-Heuristik und Zustandsbewertung

Bei der exhaustiven Lösungsbestimmung muss für jeden Passanten und für jede Sitzgelegenheit die Möglichkeiten überprüft werden, dass der Passant die Sitzgelegenheit benutzt. Da sich der Zustandsbaum dann je nach zu überprüfender Alternative in eine der beiden Möglichkeiten verzweigt, ergibt sich, dass insgesamt *bis zu* $2^{n \cdot k}$ Möglichkeiten zu überprüfen sind. Diese Problemkomplexität wird auch an der Belegungsmatrix (siehe Ausdruck 3.1) deutlich, die n Zeilen und k Spalten mit binären Wertebereich für jedes Element hat: Schreibt man die Zeilen dieser Matrix hintereinander, dann ergeben sich $n \cdot k$ binäre Stellen, d. h. ein Zahlenraum von 0 bis $2^{n \cdot k} - 1$.

Dadurch führt die exhaustive Lösungsbestimmung schon für relativ kleine Probleminstanzen zu inakzeptabler Berechnungskomplexität. Zum Bsp. gibt es schon für nur 9 Passanten und 5 Sitzgelegenheiten bis zu $2^{9 \cdot 5} = 2^{45} > 3,5 \cdot 10^{13}$ (35 Billionen) Möglichkeiten für Belegungen. Selbst, wenn jede mögliche Belegung in nur einer Nanosekunden (10^{-9} s) erzeugt und bewertet werden könnte – was sehr optimistisch ist[36] –, bräuchte die Auslotung von $2^{9 \cdot 5}$ Belegungsmöglichkeiten dann etwa 9 Stunden. Es handelt sich dabei zwar um einen *worst Case*, da, wie oben beschrieben, wegen begrenzter Sitzplatzkapazitäten nicht immer alle Pfade voll expandiert werden. Allerdings wäre es angesichts der schieren Unzahl prinzipiell möglicher Belegungen hier nicht vernünftig zu

[36] Zur Einordnung: Licht kommt in einer Nanosekunde 30 cm weit.

erwarten, dass die Dauer für exhaustive Berechnungen in einem einstelligen Minutenbereich liegt.

Für hinnehmbare Berechnungskomplexität erfolgt die Bestimmung einer Lösung heuristisch. Hierzu wird die bereits in Kapitel 2.7.2 beschriebene *Bestensuche* als Meta-Heuristik verwendet. Der diese Meta-Heuristik realisierende Algorithmus 1 wird dabei wie folgt geringfügig modizifiert, um auszuschließen, dass eine bestimmte Rechenkomplexität C, gemessen an der Zahl expandierter Zustandsknoten, überschritten wird:[37]

- Füge einen Zähler c ein, welcher die Zahl der Expansionen zählt. (Inkrementierte c direkt vor oder nach der Zeile „Expandierte T...“.)
- Füge ein: Wenn $c \geq C - n \cdot k$, dann
  - unterbinde das Hinzufügen expandierter Zustände in die *closed*-Liste (d. h. füge T nicht mehr der *closed*-Liste hinzu),
  - behalte nur den heuristisch am besten bewerteten Zustandsknoten in der *open*-Liste (d. h. lösche am Ende der While-Schleife alle Elemente, außer dem ersten in der *open*-Liste).

Mit dieser Modifikation kann für jedes $C \geq n \cdot k$ garantiert werden, dass nicht mehr als C Expansionen durchgeführt werden. Es ist zu beachten, dass die Zahl der Expansionen in jedem Fall mindestens $n \cdot k$ beträgt, da jeder vollständige Pfad über $n \cdot k$ Verzweigungsknoten geht, auch wenn C kleiner als $n \cdot k$ festgelegt ist. Die Modifikation wird deshalb eingebaut, weil die Lösungsbestimmung auch unter Nutzung einer Heuristik im ungünstigsten Fall exhaustiv werden kann. Wenn aber bei der modifizierten Meta-Heuristik die Zahl der bereits durchgeführten Expansionen die Schwelle von $C - n \cdot k$ überschreitet, dann wird das Verfahren einen Zielknoten nur noch über einen Pfad, gleichsam direkt, suchen. Da Pfadabbrüche nur in der linken Verzweigung auftreten können, existiert von jedem Knoten aus ein Pfad zu einem Zielknoten und da ein vollständiger Pfad genau $n \cdot k$ Verzweigungsknoten hat, wird $C \geq n \cdot k$ dabei garantiert nicht überschritten. Wenn es zu einem Überschreiten der Schwelle kommt, dann kann das Verfahren zwar noch bei jeder Verzweigung eine Bewertung vornehmen, aber einen eingeschlagenen Pfad nicht mehr revidieren. Wenn das Verfahren aber schon frühzeitig solche Alternativen ausgeschlossen hat, die letztlich sehr gering bewertet würden, so resultiert trotzdem eine akzeptable Lösung.

Es wird nun eine A-Suche gemäß Def. 2 (für eine, wie beschrieben, leicht modifizierte Bestensuche) entwickelt, d. h. es ist insbesondere eine heuristische

[37] C kann dann in Abhängigkeit der Rechenleistung und akzeptabler Berechnungsdauer bestimmt werden.

Funktion $e(\tau)$ zu entwickeln, die sich aus den tatsächlichen Kosten $g(\tau)$ vom Startknoten zum Zustandsknoten τ plus einer Heuristik $h(\tau)$ zusammensetzt, welche die geringsten Kosten vom Zustandsknoten τ bis zum Zielknoten abschätzt. Hierzu sei zunächst dargelegt, wie die Kosten überhaupt bestimmt werden. Denn das Kostenmaß spielt eine Rolle für das gesamte heuristische Verfahren. Die Berücksichtigung der Kosten vom Startzustand bis zum Zustand τ soll dafür sorgen, dass die Verfolgung eines Pfades abgebrochen wird, wenn sich die tatsächlichen Kosten auf diesem Pfad als zu hoch herausstellen.

Die aggregierte Safety-Gesamtbewertung $\min_i \hat{s}(i, B)$ – siehe Ausdrücke 3.30 und 3.8 – ist dabei als Kostenmaß nicht geeignet, da die Safety-Gesamtbewertung eine Mittelwertsbildung beinhaltet und implizit unterstellt, dass die Passanten die komplette Strecke bewältigt haben. Die implizite Unterstellung kommt durch die zusätzlichen Summanden in Zähler und Nenner zum Ausdruck, wobei insbesondere der zusätzliche Summand im Nenner auch aus rein mathematischen Gründen nicht einfach weggelassen werden kann, weil er verhindert, dass der Nenner 0 werden kann. Auch wenn eine Weise gefunden wird, mit dieser impliziten Unterstellung umzugehen, so führt die Mittelwertsbildung dazu, dass die Safety-Gesamtbewertung nicht monoton ist. Die Mittelwertsbildung erfolgt zur Normalisierung, damit die Gesamtbewertung für einen Passanten nicht nur dadurch höher wird, je mehr Sitzgelegenheiten dieser belegt. Ein Mittelwert kann aber durch Hinzukommen eines weiteren Wertes steigen oder sinken, je nachdem, ob der hinzukommende Wert größer oder kleiner als der bisherige Mittelwert ist. Durch diese Nichtmonotonie würde $g(\tau)$ allerdings seine steuernde Wirkung verfehlen.

Aus diesen Gründen wird für die Kosten, um vom Startzustand zum Zustand τ zu gelangen, die maximale Distanz $d_i^{fatigued}(\tau)$ verwendet, die ein Passant i in ermüdetem Zustand bis zum Zustand τ zurücklegen musste:

$$g(\tau) = \max_i d_i^{fatigued}(\tau) \ . \tag{3.32}$$

Da sich die mit kritischem Ermüdungsgrad insgesamt zurückgelegte Distanz pro Passant nur erhöhen, sich aber nicht wieder verringern kann, ist auch die unter den Passanten maximale Distanz mit kritischem Ermüdungsgrad monoton wachsend. Wenn dann in einem Pfad $g(\tau)$ gem. Ausdruck 3.32 verhältnismäßig hoch wird, dann wird die Meta-Heuristik die Expansion eines anderen Zustandes vorziehen.

Es erscheint sachdienlich die Bestimmung von $g(\tau)$ noch etwas weiter zu erläutern. Die Funktion $g(\tau)$ soll zwar die tatsächlichen Kosten vom Startzustand zum Zustand τ abbilden, ist hier aber bereits *durch spezifische Problemei-*

genschaften heuristisch informiert. Als Bestandteil der gesamten heuristischen Bewertung $e(\tau)$ ist diese heuristische Informiertheit für $g(\tau)$ begründbar. Es werden zwei problemspezifische Informationen ausgenutzt:

(1.) Die linke Seite von $s_{(a)}$ ist viel entscheidender als die rechte Seite und

(2.) $s_{(b)}$ ist im Verhältnis unwesentlich.

(Ad 1.) Die linke Seite von $s_{(a)}$ bewertet die Distanz, die ein Passant bis zur nächsten allozierten Sitzgelegenheiten zurücklegen muss, nachdem dieser den kritischen Ermüdungsgrad von $f_i(\tau) = 1$ erreicht und ein dringendes Pausenerfordernis hat. Die rechte Seite bewertet hingegen die Distanz um die eine allozierte Sitzgelegenheit für einen Passanten gleichsam verfrüht kommt. Es erscheint fachlich einleuchtend, dass die Distanz, die ein Passant mit Pausenerfordernis zurücklegen muss, entscheidender ist. Mathematisch ist dies in der Safety-Funktion durch die Parameter $(z_{li} \mid s_{li})$ und $(z_{re} \mid s_{re})$ ausgedrückt, wenn diese so gewählt werden, dass die linke Seite von $s_{(a)}$ bedeutend steiler wird, d. h. v. a. im Bereich um $z = 0$ betragsmäßig die Steigung auf der linken Seite erheblich größer ist als auf der rechten Seite, siehe Ausdruck 3.27 und vgl. Abb. 3.7.

(Ad 2.) $s_{(b)}$ bewertet die Nutzbarkeit der Hinsetz- und Aufstehassistenz bei Benutzung einer adaptiven Sitzgelegenheit. Diese stellt zwar eine Besonderheit für die vorliegende Problemstellung dar, wird aber durch $g(\tau)$ nicht berücksichtigt. Wenn adaptive Sitzgelegenheiten eine knappe Ressource sind, dann ist ohnehin davon auszugehen, dass die Assistenzfunktionalität nur sehr selten genutzt werden kann, da es dann nur sehr selten vorkommen dürfte, dass eine Sitzgelegenheit von keinem anderen Passanten als dem zu unterstützenden belegt ist. Für den Fall, dass nicht die Sitzgelegenheiten die knappen Ressourcen darstellen, sondern Sitzgelegenheiten mit effektiv nutzbarer Assistenzfunktionalität, wird das heuristische Verfahren also wenig beitragen können. Wenn aber bekannt ist, dass die Sitzgelegenheiten selber nicht knapp sind, dann kann die Sitzplatzkapazität κ auf 1 gesetzt werden, um sicherzustellen, dass jeder Passant auch die Assistenzfunktionalität nutzen kann. Die Nichtberücksichtigung von $s_{(b)}$ im heuristischen Verfahren kann aber dann als angemessen angesehen werden, wenn $\lambda_{(a)} \gg \lambda_{(b)}$. Die Festsetzung der Parameterwerte erfolgt im Kap. 4 (wo auch analyisiert wird, inwieweit das heuristische Verfahren wirksam für die Problemstellung der vorliegenden Arbeit ist).

Schließlich bleibt noch zu klären, warum Ausdruck 3.32 Maximierung beinhaltet (und nicht etwa Minimierung, wie in Ausdruck 3.30): Bei Betrachtung der linken Seite von $s_{(a)}$ wird deutlich, dass der Safety-Wert umso höher ist, je niedriger der Erfordernisgenauigkeitsindikator z betragsmäßig ist; d. h. je höher

die zurückgelegte Distanz mit kritischen Ermüdungsgrad ist, desto geringer ist die Safety-Bewertung diesbezüglich. Wenn nun unter den Distanzen, welche die Passanten mit kritischem Ermüdungsgrad zurücklegen müssen, die maximale Distanz betrachtet wird, dann wird implizit die minimale Safety-Bewertung diesbezüglich berücksichtigt. Es wird zwar bspw. nicht berücksichtigt, dass eine kumulierte Distanz von 100 m mit kritischen Ermüdungsgrad und einer Pause erheblich niedriger bewertet sein kann, als die gleiche kumulierte Distanz von 100 m und vier Pausen, da mehrere relativ kurze Distanzen mit kritischem Ermüdungsgrad weitaus weniger gravierend sind als eine relativ lange Distanz mit kritischem Ermüdungsgrad. Dennoch wird die Safety-Gesamtbewertung für einen Passanten i stets umso geringer, je höher $d_i^{fatigued}(\tau)$ wird. Im Sinne der Aggregierungslogik sind die schlechtesten Ausprägungen zu betrachten, um genau diese zu optimieren. Da – im Vorgriff auf die Entwicklung der eigentlichen Heuristik $h(\tau)$ im nachfolgenden Abschnitt 3.3.2.2 – die maximale Distanz mit kritischem Ermüdungsgrad unter den Passanten auch das Maß für die gesamte heuristische Funktion $e(\tau)$ darstellt, muss in der Meta-Heuristik die Sortierung der heuristisch bewerteten expandierten Zustandsknoten in *aufsteigender* Reihenfolge erfolgen; d. h. der Zustandsknoten mit der heuristisch bestimmten kleinsten maximalen (Gesamt-) Distanz mit kritischem Ermüdungsgrad muss stets als erstes Element in der *open*-Liste stehen.

3.3.2.2 Die Heuristik

Unabhängig von dem zu $g(\tau)$ und $h(\tau)$ im vorherigen Abschnitt 3.3.2.1 Geschriebenem gilt für die zu entwickelnde Heuristik *idealerweise*:

$$h(\tau) = \left(\min_i \hat{s}(i, B) \right) - g(\tau) \ , \tag{3.33}$$

$$\Leftrightarrow g(\tau) + h(\tau) = \left(\min_i \hat{s}(i, B) \right) \ . \tag{3.34}$$

Denn dann könnte die Meta-Heuristik unter Nutzung der *absteigend* sortierten heuristischen Funktionswerte $e(\tau) = g(\tau) + h(\tau)$ auf direktem Wege, d. h. mit nur genau $n \cdot k$ expandierten Verzweigungsknoten, ein B^* finden, für das $\xi_\kappa(B) = 1$ (nicht nur für ein möglichst hohes, sondern) für das *höchstmögliche* $agg(\hat{\mathbf{s}})$ gilt (vgl. Ausdrücke 3.9 – 3.11 u. 3.30). Wenn es allerdings möglich ist, dieses Ideal zu erfüllen, dann handelt es sich nicht um ein wirkliches Suchproblem, weil es dann schließlich möglich wäre, das Problem „direkt“, ohne Auslotung verschiedener Alternativen, exakt zu lösen. Die heuristische Lösung wird daher von diesem Ideal abweichen (müssen).

Es soll gezeigt werden, dass eine Heuristik konstruiert werden kann, um

das Problem so zu lösen, dass zum einen der Berechnungsaufwand hinnehmbar und zum anderen das Ergebnis zufriedenstellend ist. Mit Verweis auf die wesentliche Aussage der am Ende von Kap. 2.7.2 angesprochenen No-Free-Lunch-Theoreme, kann die Heuristik gleichsam als Implementierung der Ausnutzung spezieller Problemeigenschaften aufgefasst werden. An die Berechnung der Heuristik $h(\tau)$ wird dabei die Anforderung gestellt, dass die Zahl der Rechenschritte in Abhängigkeit der Eingabelänge (Anzahl der Passanten und Sitzgelegenheiten) durch ein Polynom beschränkt ist, d. h. dass die Heuristik bezogen auf n und k in $\mathcal{P}$ liegt. Denn dann gilt sie im komplexitätstheoretischen Sinne als effizient.

Das heuristische Verfahren wird durch vier Maßnahmen heuristisch verkürzt (s. u.). Mit heuristischer Verkürzung ist gemeint, dass die Maßnahmen die Komplexität der Suchlogik reduzieren sollen. Es ist dabei allerdings von Bedeutung, dass jede dieser vier heuristischen Maßnahmen in dem Sinne mechanistisch begründet ist, dass es in Zusammenhang der Logik für das Problemlösungsverfahren gestellt wird. Denn das Lösungsverfahren wird durch die heuristischen Verkürzungen auch potentiell vierfach unterminiert. Mit potentieller Unterminierung ist hier gemeint, dass die Problemlösung auch in eine völlig falsche Richtung gehen könnte. Dies zeigt, dass heuristische Lösungen im vorliegenden Sinne, wie in Abschnitt 2.7.2 aufgezeigt, aus (vernunftgeleitet) explizierbaren Verfahren resultieren und nicht auf spontanen Eingebungen beruhen (sollten). Die unten durch Algorithmus 7 beschriebene Heuristik selber zeigt, dass es sich hierbei auch nicht um eine Ein-Schritt-Lösung handelt, sondern vielschrittige Berechnungen beinhaltet. Die vier Verkürzungsmaßnahmen für das vorliegende heuristische Verfahren sind: (1.) *Restriktion*, (2.) *Substitution*, (3.) *Approximation* und (4.) *Relaxation*.

Ad 1.) *Restriktion*: Die Maßnahme der Restriktion wurde bereits in der Meta-Heuristik ergriffen, indem die Bedingung $c \geq C - n \cdot k$ eingefügt wurde, siehe Beginn des Abschnitts 3.3.2.1. Dadurch wird der *Suchraum effektiv beschränkt*. Wie bereits geschrieben, soll dies sicherstellen, dass eine bestimmte Höchstzahl von Expansionen nicht überschritten wird. Das Verfahren kommt trotz dieser Beschränkung aber zu einem zufriedenstellenden Ergebnis, wenn die Heuristik so effektiv ist, dass ungünstige Verzweigungen schon früh quasi aussortiert werden.

Ad 2.) *Substitution*: Die Maßnahme der Substitution *ersetzt die Zielgröße* durch eine andere. Diese wurde, wie die Maßnahme der Restriktion, ebenfalls bereits im vorherigen Abschnitt 3.3.2.1 zur Zustandsbewertung $g(\tau)$ ergriffen und es wurde bereits vorweggenommen, dass dies auch für $h(\tau)$ erfolgt. Anstelle der eigentlichen Zielgröße, die auf der rechten Seite des Ausdrucks 3.34 steht, wird die (ganz) rechte Seite des folgenden Ausdrucks 3.35 verwendet, wobei

τ_{ziel} der Zielzustand ist, bis zu dem $h(\tau)$ die Kosten (ab τ) schätzt:

$$e(\tau) = g(\tau) + h(\tau) = \max_i d_i^{fatigued}(\tau_{ziel}) \ . \tag{3.35}$$

Die Adäquatheit dieser Maßnahme wurde bereits vorher begründet (siehe S. 106). Nach der Substitution der Zielgröße gilt für die Heuristik $h(\tau)$ dann nach Umstellen des Ausdrucks 3.35 und nach Einsetzen des Ausdruck 3.32 für $g(\tau)$ analog zu Ausdruck 3.33 *idealerweise*:

$$h(\tau) = \left(\max_i d_i^{fatigued}(\tau_{ziel}) \right) - \left(\max_i d_i^{fatigued}(\tau) \right) \ . \tag{3.36}$$

Ad 3.) *Approximation*: Die Maßnahme der Approximation bezieht sich ausschließlich auf die Heuristik und kommt daher hier erstmalig zur Sprache. Mit der Maßnahme der Approximation wird implizit davon ausgegangen, dass für die Zielgröße *näherungsweise* gilt:

$$\max_i d_i^{fatigued}(\tau_{ziel}) \approx \left(\max_i d_i^{fatigued}(\tau) \right) + \left(\max_i d_i^{fatigued}[\tau_{ziel} - \tau] \right) \ , \tag{3.37}$$

wobei $d_i^{fatigued}[\tau_{ziel} - \tau]$ als Kurzschreibweise für $d_i^{fatigued}(\tau_{ziel}) - d_i^{fatigued}(\tau)$ verwendet wird, d. h. die kumulierte Distanz angibt, die Passant i zwischen dem Diskursweltzustand τ und dem entsprechenden Zielzustand mit Pausenerfordernis gehen muss.

Im Allgemeinen kann im Ausdruck 3.37 die „≈“-Beziehung nicht durch Gleichheit ersetzt werden. Denn inhaltlich steht hinter dem Ausdruck 3.37 die Annahme, dass der Passant i, der vom Startzustand bis zum Zustand τ die größte Distanz mit kritischem Ermüdungsgrad zurücklegen musste, auch der Passant sein wird, der zwischen dem Zustand τ bis zum Zielzustand die größte Distanz mit kritischem Ermüdungsgrad zurücklegen wird. Dies ist aber nicht zwingend der Fall. Im Grunde ist die dahinter liegende Annahme die Folgende: Sei i der Passant, der zwischen dem Startzustand und τ kumuliert die größte Distanz mit Pausenerfordernis zurücklegen muss. Sei i' der Passant, der zwischen τ und dem Zielzustand τ_{ziel} kumuliert die größte Distanz mit Pausenerfordernis zurücklegen muss. Sei $i \neq i'$. Es wird angenommen, dass die kumulierte Distanz, die i zwischen τ und dem Zielzustand τ_{ziel} mit Pausenerfordernis zurücklegen muss, ungefähr der Distanz entspricht, die i' zwischen τ und dem Zielzustand τ_{ziel} kumuliert mit Pausenerfordernis zurücklegen muss.

Die Begründung für die Approximation ist, dass die Heuristik $h(\tau)$ „nur“ als eine Art vernünftige Richtschnur dienen muss, vergleichbar mit der Luftliniendistanz zwischen zwei Orten, um die kürzeste Distanz zwischen diesen Orten

in einem Wegenetz abzuschätzen. Der Vergleich mit der Luftliniendistanz soll deutlich machen, worin das Ziel der Heuristik liegt: Sie muss keine exakte Bestimmung darstellen, sondern soll eine Orientierung zur effektiven Steuerung der Suche liefern. Unter Beachtung der notwendigen bzw. charakterisierenden Eigenschaften von A-Suchen gemäß Definition 2 stellt $\max_i d_i^{fatigued}[\tau_{ziel} - \tau]$ eine vernünftige Orientierung zur Abschätzung der Restkosten vom Zustand τ bis zum Zielzustand dar.

Die durch den Ausdruck 3.37 zugrundegelegte Orientierung ist v. a. vor dem Hintergrund der prinzipiellen Funktionsweise von A-Suchen eine vernünftige Abschätzung der Gesamtkosten. Denn $h(\tau)$ soll eine Abschätzung liefern, die ausschließlich die Kosten von τ zum Zielzustand betrifft. Die Funktion $h(\tau)$ soll per Konstruktion unabhängig von $g(\tau)$ sein. Bei von $g(\tau)$ unabhängiger Betrachtung von $h(\tau)$ liegt in $h(\tau)$ die Information nicht vor, für welchen Passanten $g(\tau) = \max_i d_i^{fatigued}(\tau)$ gilt oder wie hoch $g(\tau)$ ist. Aufgrund dieser strikt unabhängigen Betrachtung von $g(\tau)$ und $h(\tau)$ wird τ in $h(\tau)$ implizit als neuer Startzustand festgelegt und für jeden Passanten die zurückgelegte Distanz folglich als 0 zugrunde gelegt. Die strikt unabhängige Betrachtung von $g(\tau)$ und $h(\tau)$ spielt dabei wiederum die begründete Rolle, dass $h(\tau)$ die Suche in eine vielversprechende Richtung steuern soll, während $g(\tau)$ hilft zu vermeiden, dass sich die Suche in vergeblichen Pfaden verliert (vgl. Luger, 2009, S. 140).

Es sei an dieser Stelle angemerkt, dass die rechte Seite des Ausdrucks 3.37 die linke Seite stets überschätzt. Diese Tatsache betrifft aber nicht die Eigenschaft der A*-Suche gem. Def. 2, weil diese von der Größenbeziehung zwischen der Heuristik und den tatsächlichen Kosten von τ zum Zielknoten abhängt. Davon abgesehen, wurde die A*-Eigenschaft im Postulat 2 nicht gefordert, weil es weniger wichtiger ist, dass garantiert die optimale Lösung gefunden wird, als dass eine zufriedenstellende Lösung mit hinnehmbarem Berechnungsaufwand gefunden wird.

Ad 4.) *Relaxation*: Die Maßnahme der Relaxation besteht im *Weglassen von (Neben-) Bedingungen*. Diese Maßnahme bezieht sich direkt auf die Kernlogik zur Berechnung der Heuristik $h(\tau)$. Die Kernlogik ist im Algorithmus 7 formalisiert. Die Grundidee der anzuwendenden Logik ist in Abb. 3.9 veranschaulicht und kann verbal wie folgt beschrieben werden:

- Die Relaxation besteht im Ignorieren der Kapazitätsbedingungen für die Sitzgelegenheiten. Es wird also davon ausgegangen, dass auf jeder Sitzgelegenheit unendlich viele Passanten gleichzeitig sitzen können. Dies wird zu sog. *ideellen* Belegungen führen.
- Jeder Passant i macht Sitzpausen idealerweise im Abstand von d_i^{max}. Die nächste ideale Sitzpause liegt für jeden Passanten i also in Entfernung

d_i^{max} zur Position seiner letzten Sitzpause.

- Da sich in Entfernung d_i^{max} zur Position der letzten Sitzpause von i i. d. R. keine Sitzgelegenheit befindet, wird die Sitzgelegenheit gewählt, bei der die Distanz zur idealen Position für die nächste Sitzpause am geringsten ist. Hierbei wird nicht unterschieden, ob diese Sitzpause dann vor oder hinter der idealen nächsten Pausenposition liegt. Dadurch soll ein kleiner Ausgleich dafür erfolgen, dass die rechte Seite der Safety-Funktion $s_{(a)}(z)$ (siehe Ausdruck 3.27) in der Heuristik keine explizite Berücksichtigung findet und dazu führen, dass nicht immer Sitzgelegenheiten gewählt werden, die vor der Position liegen, wo der Ermüdungsgrad kritisch wird. Denn zu frühes Einlegen von Sitzpausen führt in $s_{(a)}(z)$ zu geringerer Bewertung. Zudem sollen gemäß $\hat{s}(i, B)$ unnötige Sitzpausen vermieden werden (siehe Ausdruck 3.8).
- Da im vorherigen Schritt der Passant i stets zur idealen nächsten Sitzgelegenheit alloziert wurde, ohne zu prüfen, ob diese noch Kapazitäten hat, ist diese Allokation rein ideell. Diese ideelle Allokation wird für jeden Passanten i so lange wiederholt, bis für jeden Passant i sein Ziel-Abgrenzungswegpunkt die kürzeste Entfernung zu seinem idealen nächsten Wegpunkt hat. Dann ist ein sog. ideeller Zielzustand erreicht.
- Für die zuvor erstellte ideelle Belegung wird, bezogen auf den Diskursweltabschnitt von τ bis zum ideellen Zielzustand, die unter den Passanten maximale kumulierte mit Pausenerfordernis zurückgelegte Distanz berechnet und diese als $h(\tau)$ zurückgegeben.
- *Hinweis:* Durch die Expansionslogik wird sichergestellt, dass nur realisierbare Belegungen erzeugt werden. (Die ideellen Belegungen dienen der heuristischen Entscheidung, welche Suchpfade weiterverfolgt werden.)

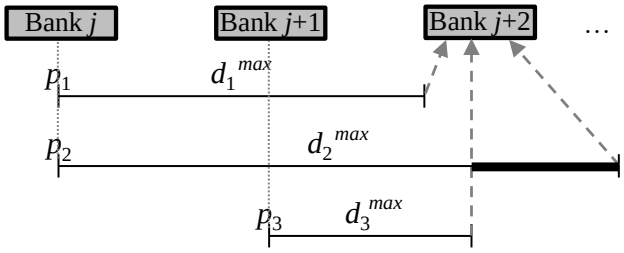
Da oben gefordert wurde, dass die Heuristik in Abhängigkeit von n und k in $\mathcal{P}$ liegen soll, sei die Laufzeit des Algorithmus 7 zur Berechnung von $h(\tau)$ analysiert: Die Schleife in Zeile 1 iteriert über alle n Passanten. Die Schleife in Zeile 4 iteriert im worst Case über alle k Sitzgelegenheiten (wobei noch ein letzter Iterationsschritt zu einem Abgrenzungswegpunkt hinzukommt). Die Schleife in 6 iteriert ebenfalls im worst Case über alle k Sitzgelegenheiten. Da in den Schleifendurchläufen keine weiteren Prozeduren aufgerufen oder aufwändige Rechenfunktionen verwendetet werden, liefert die Zahl der Schleifendurchläufe eine sachgerechte Bestimmung für die Berechnungskomplexität der Heuristik.

Aufgrund der Verschachtelung der Schleifen beläuft sich die Gesamtzahl der Iterationen im worst Case auf $n \cdot k \cdot k$. Damit ist die Laufzeitkomplexität zwar in der Größenordnung kubisch, aber durch ein Polynom beschränkt – genauer: durch ein Polynom dritten Grades – und liegt damit, wie gefordert, in

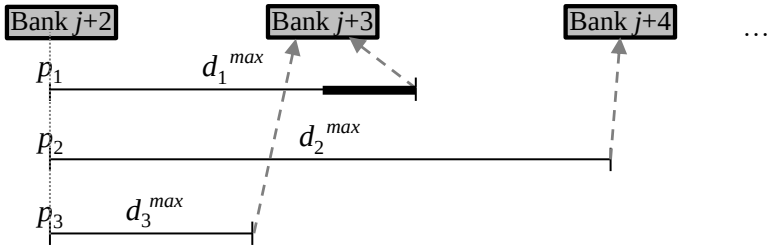
(*In diesem Beispiel* gehen alle Passanten von links nach rechts.)

– Sitzplatzkapazität $\kappa = \infty$.

→ Verzweigungszustand τ und 1. ideelle Allokation:



→ 2. ideelle Allokation:



→ 3. ideelle Allokation …

⋮

– Summiere *pro Passant* die fett hervorgehobenen Teilstrecken und gib davon die maximale Summe zurück.

Legende:

- Position der letzten Sitzpause des entsprechenden Passanten.
- Maximale Distanz, die der entsprechende Passant gehen kann, bis er ein Pausenerfordernis hat. **Fett hervorgehobene Teilstrecke**: Distanz, die der entsprechende Passant mit kritischen Ermüdungsgrad geht.
- Ideelle Allokation für den entsprechenden Passanten: (Adaptive Park-) Bank, die am wenigsten weit weg von der Position liegt, wo der Passant einen kritischen Ermüdungsgrad erreicht (und nicht die zuletzt von dem Passanten benutzte Parkbank ist).

Abbildung 3.9: Veranschaulichung der Grundidee für die Heuristik

Algorithmus 7 Berechnung der Heuristik $h(\tau)$

input: τ

for all $i \in Passanten$ **do**
  Nächste ideale Pausenposition $pos_{nä}^{ideal} :=$
    $letzte\ Pausenposition\ von\ i + (dir_i \cdot d_i^{max})$ ▷ $d_i^{max} = \frac{v_i}{f_i}$
  $\widetilde{d}_i^{fatigued} := 0$ ▷ Heuristisch bestimmtes $d_i^{fatigued}[\tau_{ziel} - \tau]$
  while $pos(0) \leq pos_{nä}^{ideal} \leq pos(k+1)$ **do**
    ▷ Bis ein Abgrenzungswegpunkt erreicht ist.
    $geringste\ Abweichung := \infty$
    for all $j : dir_i \cdot pos(j) > dir_i \cdot letzte\ Pausenposition\ von\ i$ **do**
      ▷ Für alle Wegpunkte, die noch vor i liegen.
      $Abweichung := |pos(j) - pos_{nä}^{ideal}|$
      if $Abweichung < geringste\ Abweichung$ **then**
        $geringste\ Abweichung := Abweichung$
        $j^* := j$
      end if
    end for
    $letzte\ Pausenposition\ von\ i := pos(j^*)$
    $pos_{nä}^{ideal} := pos(j^*) + dir_i \cdot d_i^{max}$
    if $\Big(dir_i \cdot (pos_{nä}^{ideal} - letzte\ Pausenposition\ von\ i)\Big) > 0$ **then**
      ▷ Distanz zur nächsten Pausenposition $> d$.
      $\widetilde{d}_i^{fatigued} := \widetilde{d}_i^{fatigued} + geringste\ Abweichung$
    end if
  end while ▷ Abgrenzungswegpunkt erreicht.
end for
return $h := \max_i \widetilde{d}_i^{fatigued}$

$\mathcal{P}$. Im Verhältnis zu exponentieller Komplexität wächst kubische Komplexität sehr langsam, z. B. ist $45^3 = 91.125 \ll 2^{45} \approx 35$ Billionen. Es soll in diesem Zusammenhang jedoch darauf hingewiesen werden, dass die Laufzeit der gesamten heuristische Lösungssuche nicht alleine durch die Komplexität der Heuristik bestimmt ist, sondern auch durch die Meta-Heuristik. Es kann auch bei polynomieller Komplexität der Heuristik nötig sein, explizit auf die Zahl der durch die Meta-Heuristik ausgeführten Expansionen zu achten. Das nun folgende Kapitel wird zeigen, dass das heuristische Lösungsverfahren trotz der oben beschriebenen eingebrachten heuristischen Verkürzungen äußerst effektiv ist – v. a. auch mit einer *Restriktion* durch die Meta-Heuristik, die durchaus als drastisch bezeichnet werden kann, und der geradezu extremen *Relaxation* in der Heuristik, wo die Sitzplatz-Kapazitätsbedingung komplett ignoriert wird.

4 Validierung der Wirksamkeit

4.1 Simulationsrahmen und -bedingungen

Die Wirksamkeit des entwickelten Lösungsverfahrens für die vorliegende Problemstellung wird anhand einer Simulation überprüft. Die Methode der Simulation zur Validierung eignet sich insbesondere deshalb, weil die *intendierte* Wirksamkeit des Verfahrens prinzipiell validiert werden soll. Die Entwicklung des Verfahrens basiert auf formalen Modellannahmen und die Fragestellung, gegen die hier geprüft werden soll, ist, ob das Verfahren unter diesen Annahmen das leistet, was es verspricht. Wenn das Verfahren unter den gesetzten Annahmen nicht effektiv ist, dann ist die intendierte Wirksamkeit verfehlt. Gerade für die heuristische Lösung ist die Wirksamkeit aufgrund der Konstruktion zunächst einmal eine Hypothese, die sich erst bewähren muss. Denn die Konstruktion des heuristischen Verfahrens ist zwar begründet, jedoch ist damit die Effektivität nicht analytisch bewiesen. Auch wenn für die intendierte Wirksamkeit ein analytischer Beweis vom Grundsatz her die erste Wahl darstellt, kann er nicht erbracht werden, wenn der notwendige mathematisch-analytische Apparat für die Zusammenhänge zwischen Ein- und Ausgaben nicht beherrschbar oder nicht bekannt ist. Dies ist gerade bei KI-basierten – und insb. bei in diesem Sinne heuristischen – Verfahren häufig der Fall. Auch für die vorliegende Arbeit trifft dies zu.

Die Simulation weist als Validierungsmethode eine sehr hohe interne Validität auf. Es kann analytisch ausgeschlossen werden, dass für die erhaltenen Resultate andere als die mathematisch modellierten Einflussparameter eine Rolle spielen. Falls die Simulation die Wirksamkeit jedoch nicht belegen kann, bedeutet dies allerdings umgekehrt auch, dass hierfür keine anderen Störeinflüsse geltend gemacht werden können. In diesem Falle wäre die intendierte Wirksamkeit des Verfahrens i. Allg. widerlegt.

Für die Interpretation der Simulationsergebnisse kommt den zugrunde gelegten Annahmen eine zentrale Bedeutung zu. Im Idealfall bildet die Simulation die Realwelt so nach, dass ihre Ergebnisse 1:1 auf die Realwelt übertragbar sind. Es erweist sich allerdings gleichsam als ein Paradox des Idealen, dass dieser im vorherigen Satz beschriebene Idealfall gerade durch die der mathe-

matischen Modellierung inhärenten Idealisierung nicht eintritt. Denn bei der Modellierung werden die meisten Merkmale der Realwelt verkürzt, um den Fokus auf die für die Fragestellung wesentlichen, idealtypischen Zusammenhänge zu richten. In diesem Sinne ist eine auf mathematischen Modellen basierende Simulation stets idealistisch relativiert. Diese idealistische Relativierung ist allerdings kein Mangel, sondern ermöglicht erst analytische Erkenntnis, da die Zusammenhänge in idealtypische Faktoren zerlegt – d. h. im Sinne des Begriffs: analysiert – sind. Die zugrunde gelegten Annahmen, welche die Interpretation der Simulationsergebnisse idealistisch relativieren, liegen durch das Modell der Diskurswelt (in Kap. 3.2.1) formal explizit vor.

Für die Durchführung der Simulation ist es über die für die allgemeinen Zusammenhänge getroffenen Annahmen hinaus erforderlich, eine Reihe von bislang prinzipiell offen gebliebenen Parametern im Modell mit Werten zu belegen, sodass ausführbare Simulationsfälle entstehen. Die Tabelle 4.1 stellt eine Übersicht der Simulationsparameter inkl. deren Wertefestsetzungen sowie über die betrachteten abhängigen Größen auf. Die Parameter und Größen werden im nachfolgenden Text unter Bezugnahme auf die jeweiligen in der Tabelle angegebenen Nummern erläutert. (*Hinweis*: Die Logik der Parameterfestsetzungen in den Simulationsdurchläufen wird zu Beginn des nachfolgenden Abschnittes dargelegt.)

(Ad 1) Die Lösung wird mit drei *Verfahren* bestimmt: *exhaustiv*, *heuristisch* und *intuitiv*. Die exhaustive Lösungsbestimmung erfolgt mittels Algorithmus 6; von allen vollständigen Pfaden wird derjenige mit der besten Bewertung gewählt. Die heuristische Lösungsbestimmung erfolgt wie in Kap. 3.3.2 beschrieben. Da insb. auch überprüft werden soll, ob das entwickelte Lösungsverfahren im Vergleich zur erwartungsgemäßen Sitzplatzbelegung ohne eine KI-basierte Steuerung effektiv ist, wurde als drittes Verfahren eine intuitive Sitzplatzbelegung implementiert. Hierbei gehen die Passanten so lange, bis ihr Ermüdungsgrad 1 ist, d. h. bis sie ein Pausenerfordernis haben, und setzen sich dann auf die in Gehrichtung nächste Sitzgelegenheit, auf der noch mindestens ein Platz frei ist.

(Ad 2) Die *Anzahl der Passanten* ist eine entscheidende Größe für die Knappheit der Sitzgelegenheiten. Denn die gesamte Problemstellung basiert auf der Voraussetzung, dass geeignete Sitzgelegenheiten im urbanen Raum eine knappe Ressource für Passanten sind. Daher wird die Zahl der Passanten von 1 bis 15 schrittweise erhöht. Zwar gehört zur Knappheitsbedingung für die Sitzgelegenheiten auch, dass mehrere Passanten die selben Sitzgelegenheiten in sich überlappenden Zeitfenstern benutzen möchten, denn es können beliebig viele Passanten in der Diskurswelt vorhanden sein, wenn sich ihre angestrebten Belegungszeiten für die einzelnen Sitzgelegenheiten nicht überschneiden. Aller-

Nr.	Parameter	Wert bzw. Wertebereich
(1)	Lösungsverfahren	exhaustiv ; heuristisch ; intuitiv
(2)	n	$1, 2, \ldots, 15$
(3)	Zu bewältigende Strecke	$48.000\,\mathrm{cm}$
(4)	k (inkl. der Positionen)	3 (siehe Abb. 3.5)
(5)	κ	2
(6)	dir_i	$\sim \mathcal{U}\ \{-1; 1\}$
(7)	v_i	$\sim \mathcal{U}\ \left[60\,\frac{\mathrm{cm}}{\mathrm{s}}; 140\,\frac{\mathrm{cm}}{\mathrm{s}}\right]$
(8)	$\dot{f}_i$	$\sim \mathcal{U}\ \left[\frac{v_i}{25.000\,\mathrm{cm}}; \frac{v_i}{5.000\,\mathrm{cm}}\right]$
(9)	r_i	$\sim \mathcal{U}\ [240\,\mathrm{s}; 720\,\mathrm{s}]$
(10)	$f_i(\tau_{init})$	$0\ , \forall i$
(11)	γ^{Hin} $\left(\gamma^{Auf} = 1 - \gamma^{Hin}\right)$	$0,5$ $(0,5)$
(12)	$\lambda_{(a)}$ $\left(\lambda_{(b)} = 1 - \lambda_{(a)}\right)$	$0,7$ $(0,3)$
(13)	$(z_{li} \mid s_{li})$, $(z_{re} \mid s_{re})$	$(-1 \mid 0,2)$, $(1 \mid 0,75)$
(14)	C	2.500
(15)	Anzahl Simulationsläufe	250
	Abhängige Größen	
(16)	$\max agg(\hat{\mathbf{s}})$	Zielgröße
(17)	$\max_i d_i^{fatigued}(\tau_{ziel})$	Kontrollgröße
(18)	c	Kontrollgröße

Tabelle 4.1: Simulationsparameter und abhängige Größen

dings ergibt sich in der vorliegenden Diskurswelt diese Knappheit schon alleine durch Erhöhen der Anzahl der Passanten, weil Zeitüberschneidungen für angestrebte oder anvisierte Sitzplatzbelegungen dann direkte Folge sein werden (siehe dazu auch die Erläuterung zu 3 & 4). Höhere Anzahl von Passanten bedeutet hier aber nicht nur knappere Sitzgelegenheiten, sondern auch eine (erheblich) höhere Problemlösungskomplexität.

(Ad 3 & 4) Die *Strecke*, sowie die *Positionen – und damit auch die Anzahl – der smarten Sitzgelegenheiten* darauf, ist durch die Beschreibung des Beispielszenarios gegeben, welches in Abb. 3.5 dargestellt ist. Für die vorliegende Problemstellung ist die Knappheit der SSOs, d. h. deren Anzahl (und damit letztlich auch die Distanzen zwischen ihnen) im Verhältnis zur Anzahl der Passanten, relevant. Um Auswirkungen aufgrund dieses Verhältnisses zu untersuchen genügt es, die Anzahl der Passanten zu variieren und die Strecke und die SSOs als konstante Größen zu belassen. Strecke und Anzahl der SSOs bilden dabei aber keinen erkennbar besonderen Ausnahmefall, wie er bspw. zu erwarten wäre, wenn so viele SSOs auf der Strecke vorhanden sind, dass sich jeder Passant bei einem Pausenerfordernis direkt auf eine Sitzgelegenheit setzen kann.

(Ad 5) Die *Sitzplatzkapazität pro Sitzgelegenheit* wird mit $\kappa = 2$ festgesetzt. Dies entspricht nicht nur einer durchaus realistischen, beobachtbaren Kapazitätsbeschränkung, sondern ist auch die kleinste Kapazität, bei der es möglich ist, dass ein sitzender Passant verhindert, dass ein anderer Passant die Hinsetz- oder Aufstehassistenz nutzen kann.

(Ad 6) Die *Gehrichtungen der Passanten* und damit auch ihre Startpositionen werden jeweils per Zufallsziehung bestimmt, wobei die Gehrichtungen -1 und $+1$ gleichverteilt sind.

(Ad 7) Die *Gehgeschwindigkeiten der Passanten* werden ebenfalls jeweils per Zufallsziehung aus einer Gleichverteilung bestimmt. Hierzu könnte zwar im Prinzip auch eine Normalverteilung zugrunde gelegt werden, weil Gehgeschwindigkeiten als eine natürliche Größe tendenziell einer Normalverteilung gehorchen. Hier wurde die Gleichverteilung aus dem Grunde gewählt, dass sie keine Ausprägungen bevorzugt (und dabei einfacher zu handhaben ist). Bei einer Normalverteilung ist der Mittelwert gleichzeitig der Modalwert, d. h. der am häufigsten gezogene Wert. Bei Verwendung einer Gleichverteilung ist dies nicht der Fall und die Streuung ist höher. Dies wird als ein Vorteil erachtet, um zu zeigen, dass die etwaig belegte grundsätzliche Wirksamkeit nicht von einer bestimmten Gehgeschwindigkeit abhängt (da bei Normalverteilungen die erwartete Gehgeschwindwindigkeit, wie bereits geschrieben, auch die am häufigst gezogene Gehgeschwindikeit wäre). Die Intervallgrenzen der Gleichverteilungen wurden mit 60 cm/s und 140 cm/s so gewählt, dass jede Zufallsziehung

auch realistisch erfüllbar bzw. in der Realität anzutreffen ist. Für die prinzipielle Wirksamkeit des Lösungsverfahrens spielt das aber eigentlich gar keine entscheidende Rolle, da alle Passanten aus der gleichen Gleichverteilung ziehen und das Verfahren an keiner Stelle spezifische Informationen über absolute Werte verwendet.

(Ad 8) Die *Ermüdungsraten* der Passanten werden jeweils per Zufallsziehung aus einer Gleichverteilung mit den Intervallgrenzen $v_i \div 25.000\,\text{cm}$ und $v_i \div 5.000\,\text{cm}$ bestimmt, wobei die Divisoren (bzw. Nenner) jeweils die maximale Distanz d_i^{max} angeben, die der jeweilige Passant gehen kann, bevor er ein Pausenerfordernis hat, siehe Ausdruck 3.3. Es wurde keine logische Abhängigkeit in die Zufallsziehung integriert, aus der etwa für einen langsamen Passanten ein kleines d_i^{max} folgt. Auch wenn eine solche Logik sachgerecht sein könnte, würde sie die Gesamtinterpretierbarkeit erschweren. Für die vorliegenden Validierung soll vermieden werden, dass Werte systematisch gemeinsam auftreten, wenn sie nicht durch das Lösungsverfahren systematisch gemeinsam auftreten (d. h. die sog. unabhängigen Variablen sollen in ihrer „Entstehung“ auch unabhängig voneinander sein, sodass ihre Wirkungen auf die Zielgrößen ggf. jeweils unabhängig voneinander analysiert werden können; die Festsetzung der Ermüdungsraten erfolgt nur deshalb über die Gehgeschwindigkeiten, damit die Intervallgrenzen mittels d_i^{max} gesetzt werden können, was leichter interpretierbar ist, als direkt irgendwelche Werte zwischen 0 und 1 festzusetzen – es ergibt sich dadurch a priori keine Korrelation zwischen den Gehgeschwindigkeiten und den Ermüdungsraten). Hinsichtlich der Verwendbarkeit einer Normalverteilung gilt Analoges wie für (7) die Gehgeschwindigkeiten.

(Ad 9) Die *Regenerationsraten* der Passanten werden jeweils per Zufallsziehung aus einer Gleichverteilung mit den Intervallgrenzen 240 s und 720 s bestimmt, welche die Zeit angibt, die ein Passant auf einer Sitzgelegenheit verweilt, wenn er diese benutzt. Diese Verweildauer ist unabhängig vom Ermüdungsgrad zum Zeitpunkt, an dem sich der Passant hinsetzt. Es wird also angenommen, dass ein Passant, wenn er sich einmal hinsetzt, stets eine gewissen Zeit sitzt, auch wenn er schon erholt ist. Diese Annahme dient der Vereinfachung für die Modellimplementierung und ändert nichts an der Grundsätzlichkeit der Problemzusammenhänge. Auch hier wird keine logische Abhängigkeit zu anderen Größen wie die Gehgeschwindikeit oder Ermüdungsrate konstruiert. (Für die Verwendbarkeit einer Normalverteilung gilt Analoges wie für (7) die Gehgeschwindigkeiten.)

(Ad 10) Initiale Ermüdungsgrade: Alle Passanten haben initial einen Ermüdungsgrad von 0. (Tatsächlich könnten Werte für die initialen Ermüdungsgrade auch mit wenig Aufwand anders festgesetzt werden. Es erscheint aber plausibel, dass sich die Passanten vollständig erholt auf den Weg machen – zumal

die Abgrenzungswegpunkte im Szenario Orte zum Verweilen sind.)

(Ad 11) Die *Safety-Parameter für die Nutzbarkeit der Hinsetz- und Aufstehassistenz* wurden auf $\gamma^{Hin} = \gamma^{Auf} = 0,5$ festgesetzt. Ohne weitere Differenzierungen erscheint es sinnvoll davon auszugehen, das Hinsetzassistenz genauso nützlich ist wie Aufstehassistenz.

(Ad 12) Für die *Gewichtungsfaktoren der zusammengesetzten Safety-Funktion* wurde $\lambda_{(a)} = 0,7$ und $\lambda_{(b)} = 0,3$ festgesetzt. Hierin drückt sich die (von der Heuristik implizit verwendete) spezifische Information aus, dass erfordernisgenaue Verfügbarkeit von Sitzgelegenheiten, d. h. Adaptivität bzgl. (a) Verfügbarkeit, wichtiger ist als Hinsetz- und Aufstehassistenz, d. h. Adaptivität bzgl. (b) Höhe der Sitzfläche. Dies ist auch damit begründet, dass Adaptivität bzgl. Höhe der Sitzfläche ohnehin nur selten nutzbar ist, wenn Sitzgelegenheiten knapp sind und daher fast durchgehend von mehreren Passanten gleichzeitig benutzt werden. (Wenn eine andere als der zu assistierende Passant eine Sitzgelegenheit benutzt, dann wird die Adaptivität bzgl. Höhe der Sitzfläche deaktiviert, um Nutzungskonflikte zu verhindern.) Die Parameterwerte für $\lambda_{(a)}$ und $\lambda_{(b)}$ stehen ungefähr im Verhältnis von $1 : 2$ zueinander.

(Ad 13) Die *Safety-Funktion für erfordernisgenaue Verfügbarkeit* wurde mit $(z_{li} \,|\, s_{li}) = (-1 \,|\, 0,2)$ und $(z_{re} \,|\, s_{re}) = (1 \,|\, 0,75)$ so parametrisiert, wie sie in Abb. 3.7 dargestellt ist. Hierin drückt sich die (von der Heuristik implizit verwendete) spezifische Information aus, dass die linke Seite dieser Funktion erheblich steiler ist, als ihre rechte Seite. Die Parameterfestsetzung kann als Teil des Konstuktionsbeitrags für die spezifizierte vorliegende Problemstellung betrachtet werden.

(Ad 14) Die *maximale Anzahl an Expansionen* wird hier mit 2.500 recht deutlich beschränkt. Dies soll aber zeigen, dass die Heuristik auch bei einer solchen Restriktion effektiv ist.

(Ad 15) Die *Zahl der Simulationsläufe* bezieht sich auf die Zahl der gezogenen Zufallsgrößen (6 – 9) je Anzahl an Passanten n. Hierbei werden für jedes der drei Lösungsverfahren im jeweils entsprechendem Simulationslauf dieselben Parameter verwendet, d. h.: Für jeden Passanten, werden die Zufallsgrößen je n und je Simulationslauf gezogen. Die Zahl der Simulationsläufe wurde auf 250 festgesetzt. Es ist zu erwarten, dass sich dadurch hinreichende Tendenzen für die abhängigen Größen abzeichnen.

(Ad 16) Hierbei handelt es sich um *die Zielgröße*. Dabei handelt es sich um das $agg(\hat{\mathbf{s}})$, welches sich aus der Forderung gem. Ausdruck 3.10 ergibt. Die Maximierungsforderung bezieht sich auf die Wahl des Zielzustandes, in welchem die aggregierte Safety-Gesamtbewertung möglichst hoch ist. Die Lösungsverfahren führen diese Wahl für die durch sie aufgedeckten Zielzustände durch. (Es wird sich also nur beim exhaustiven Verfahren garantiert um das globale

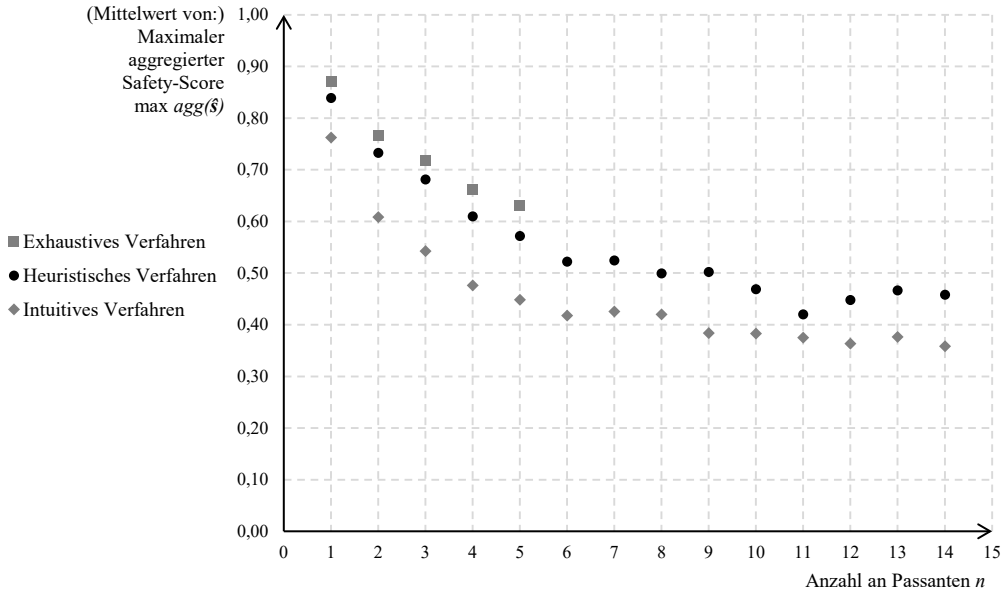


Abbildung 4.1: Ergebnisse für die Safety-Gesamtbewertung (Zielgröße)

Maximum handeln.)

(Ad 17) Die sich ergebende *maximale Distanz, die ein Passant kumuliert in ermüdetem Zustand zurücklegen musste* wird als Kontrollgröße protokolliert. Die Heuristik verwendet diese Größe als Substitut für das Safety-Maß. Die sich ergebende maximale Distanz unter den Passanten sollte möglichst klein sein.

(Ad 18) Die *Zahl der Expansionen* wird als Kontrollgröße protokolliert.

4.2 Simulationsergebnisse

Die Ergebnisse der Simulation sind in der Tabelle 4.2 dargestellt, die durch die Abb. 4.1 als Plot visualisiert werden. Die Tabellen 4.3 und 4.4 stellen zudem die Kontrollgrößen dar. Dargestellt sind jeweils die arithmetischen Mittelwerte aus 250 Simulationsdurchläufen. Die Varianz in den 250 Simulationsläufen ist in allen Fällen so gering, dass davon ausgegangen werden kann, dass die Konfidenzintervalle für die Mittelwerte genügend schmal sind, um die Mittelwerte als eine geeignete Schätzung der theoretischen Erwartungswerte für die Zielgrößen aufzufassen, die bei Verwendung der Lösungsverfahren jeweils resultieren.

Erzieltes $\max agg(\hat{\mathbf{s}})$ (Gerundete Mittelwerte aus 250 Durchläufen; Varianz $\sigma^2 \leq 0,3$ jeweils $\forall n$)			
$\boldsymbol{n}$	**exhaustiv**	**heuristisch**	**intuitiv**
1	0,87	0,84	0,76
2	0,77	0,73	0,61
3	0,72	0,68	0,54
4	0,66	0,61	0,48
5	0,63	0,57	0,45
6	–	0,52	0,42
7	–	0,52	0,43
8	–	0,50	0,42
9	–	0,50	0,38
10	–	0,47	0,38
11	–	0,42	0,38
12	–	0,45	0,36
13	–	0,47	0,38
14	–	0,46	0,36
15	–	0,46	0,35

Tabelle 4.2: Ergebnisse für die Safety-Gesamtbewertung (Zielgröße)

Das exhaustive Verfahren wurde nur für Passantenzahlen $n \leq 5$ ausgeführt.[38] Es ist aber ein klarer Trend zu erkennen, der sich für $n > 5$ extrapolieren lässt. Der Fokus der Betrachtung liegt ohnehin auf dem heuristischen Verfahren im Vergleich zum intuitiven Verfahren, welche beide bis $n = 15$ ausgeführt wurden.

[38] Die Berechnungen in 250 Durchläufe dauerten auf der eingesetzten Hardware – Intel(R) Core(TM) i7-8550U CPU @ 1,8 GHz, 1,99 GHz, 4 Kerne, 8 logische Prozessoren, x64-basierter Prozessor, 24 GB RAM Hauptspeicher – ab $n > 5$ unakzeptabel lange.

	Sich ergebendes $\max_i d_i^{fatigued}(\tau_{ziel})$		
n	**exhaustiv**	**heuristisch**	**intuitiv**
1	12.327 cm	9.611 cm	15.471 cm
2	21.159 cm	16.831 cm	22.469 cm
3	24.032 cm	19.263 cm	24.807 cm
4	28.850 cm	22.691 cm	27.296 cm
5	31.337 cm	24.233 cm	27.909 cm
6	–	25.877 cm	28.430 cm
7	–	26.103 cm	28.082 cm
8	–	27.034 cm	28.507 cm
9	–	27.374 cm	29.177 cm
10	–	29.206 cm	29.444 cm
11	–	29.373 cm	29.538 cm
12	–	28.570 cm	29.479 cm
13	–	27.988 cm	29.356 cm
14	–	28.316 cm	29.696 cm
15	–	27.698 cm	29.492 cm

Tabelle 4.3: Ergebnisse für die mit kritischem Ermüdungsgrad zurückgelegte Distanz (Kontrollgröße)

Benötigte Anzahl der Expansionen: c			
$\boldsymbol{n}$	**exhaustiv**	**heuristisch**	**intuitiv**
1	15	4	4
2	127	12	7
3	1.023	46	10
4	8.191	202	13
5	65.535	494	16
6	–	1.003	19
7	–	1.554	22
8	–	2.079	25
9	–	2.355	28
10	–	2.445	31
11	–	2.490	34
12	–	2.490	37
13	–	2.489	40
14	–	2.489	43
15	–	2.488	46

Tabelle 4.4: Ergebnisse für die Zahl der Expansionen (Kontrollgröße)

4.3 Interpretation der Simulationsergebnisse

Die Ergebnisse belegen die Effektivität des heuristischen Lösungsverfahrens. Der Safety-Wert mit dem heuristischen Verfahren ist durchweg höher als beim intuitiven Verfahren. Dass der Safety-Wert bei steigender Passantenanzahl abnimmt ist erwartungsgemäß, weil dann die Knappheit geeigneter Sitzgelegenheiten deutlicher wird. Dann kommt das Safety-Allokationsproblem überhaupt erst zum Tragen. Allerdings kann auch das exhaustive Verfahren nicht verhindern, dass nicht mehr jeder Passant zu den für ihn geeignetsten Sitzgelegenheiten alloziert werden kann.

Den Ergebnissen ist zu entnehmen, dass die heuristischen Resultate nur geringfügig unter den Resultaten mit dem exhaustiven Verfahren liegen. Dies bedeutet, dass das heuristische Verfahren der optimalen Lösung ziemlich nahekommt. Die heuristischen Werte fallen für größere n (für die keine exhaustive Lösung mehr bestimmt wurde) auch nicht stark ab, sodass davon ausgegangen werden kann, dass die Leistungsfähigkeit des heuristischen Verfahrens auch für größere n Bestand hat.

Anhand der Ergebnisse kann also konstatiert werden: Das Allokationsproblem von Passanten zu geeigneten Sitzgelegenheiten ist ein Problem, zu dessen Lösung KI-basierte Informationsverarbeitung effektiv beiträgt. Denn das intuitive Verfahren liefert erheblich geringere Safety-Werte. Das intuitive Verfahren stellt im Vergleich zum KI-basierten Verfahren ein naives Vorgehen dar.

Das entwickelte Verfahren ist effektiv gegenüber dem Status quo. Für jeden Passanten resultiert bei Einsatz des entwickelten Verfahrens eine höhere erwartete Safety in Bezug auf das Erfordernis nach geeigneten Sitzplätzen für Pausen als ohne Einsatz des Verfahrens. Zwar kann sich in Einzelfällen a posteriori herausstellen, dass Passanten mit dem intuitiven Verfahren einen höheren Safety-Wert erzielt hätten. Bei wiederholter Nutzung des Systems adaptiver Sitzgelegenheiten ist das intuitive Vorgehen dem KI-basierten Verfahren im Erwartungswert aber systematisch, tendenziell deutlich unterlegen.

Es soll noch kurz auf zwei Beobachtungen in den Daten der Kontrollgrößen eingegangen werden. Erstens: Die Zahl der Expansionen beim heuristischen Verfahren konvergiert für höhere n gegen die festgesetzte Expansionsobergrenze von 2.500. Dass die Obergrenze dabei nicht erreicht wird, liegt daran, dass der Schwellwert von $C - n \cdot k$ i. d. R. nicht auf einer wurzelnahen Ebene im Zustandsbaum überschritten wird, sodass der verbleibende Pfad zum Zielknoten weniger als $n \cdot k$ Knoten umfasst. Beim intuitiven Verfahren beträgt die Zahl der Expansionen im Übrigen stets $n \cdot k + 1$, weil in diesem Verfahren ein direkter Pfad genommen wird (und der Expansionszähler auch den Versuch der Expansion des Zielknotens mitzählt).

Zweitens: Die geringste maximale kumulierte Distanz, die ein Passant mit kritischem Ermüdungsgrad zurücklegen muss, ist für das heuristische Verfahren stets geringer als beim exhaustiven Verfahren. Dies liegt daran, dass das heuristische Verfahren ausschließlich diese Größe als substituierte Zielgröße verwendet. Das exhaustive Verfahren bestimmt die optimale Lösung anhand der zusammengesetzten Safety-Funktion, und berücksichtigt somit auch, ob ein Passant eine Pause zu früh macht und ob er Hinsetz- und Aufstehassistenz nutzen kann.

5 Diskussion

5.1 Generalisierbarkeit des Lösungsverfahrens

Die Frage der Generalisierbarkeit des entwickelten Lösungsverfahrens soll in Bezug auf (1) die *Annahmen des mathematischen Modells*, (2) die gesetzten *Simulationsrahmenbedingungen* und im Hinblick auf (3) *andere Anwendungsfälle* diskutiert werden. Ersteres betrifft dabei die Frage, inwiefern die Annahmen, die im mathematischen Modell enthalten sind, Schlüsse für die empirisch erlebbare Realwelt zulassen. Zweiteres betrifft die Frage, ob die erzielten Ergebnisse nur deshalb erzielt wurden, weil die Rahmenbedingungen für die Simulation entsprechend gewählt wurden, aber sonst völlig anders ausfallen würden. Letzteres betrifft die Frage, für welche Klasse von Problemstellungen das Lösungsverfahren angewendet werden kann.

(Ad 1) Es werden die folgenden *Annahmen des mathematischen Modells* reflektiert:

(a) Passanten benutzen Sitzgelegenheiten nur gemäß der Belegungsmatrix B.

(b) Der Ermüdungsgrad der Passanten ist in der gegebenen Exaktheit bestimmt.

(c) Passanten haben eine fixe Verweildauer auf Sitzgelegenheiten.

(d) Passanten haben eine fixe Gehgeschwindigkeit.

(e) Passanten sind unabhängig voneinander unterwegs.

(f) Passanten kehren auf ihrer Strecke nicht um.

(g) Die Menge der Passanten ist geschlossen.

(Zu a) Dieser Annahme liegt wiederum die Annahme zugrunde, dass Menschen ein bestimmtes Verhalten dann ausüben, wenn es für sie von Nutzen ist (Ajzen, 1991). Im Kap. 4 wurde dargelegt, dass die ermittelten Belegungsmatrizen B zu Allokationen von Sitzgelegenheiten führen, die einen höheren aggregierten, durch die Safety-Gesamtbewertung quantifizierten, Erwartungsnutzen stiftet. Dies liefert zumindest ein vernunftgeleitetes Argument, warum Passanten die Sitzgelegenheiten nur gemäß der Belegungsmatrix B benutzen *sollten*.

(Zu b) Es ist hier nicht notwendig, dass der exakte Ermüdungsgrad bestimmt wird, sondern dass ein zeit- oder wegabhängiges Maß existiert, nach dem vernünftigerweise davon ausgegangen werden kann, dass ein Passant ein Pausenerfordernis hat. Die Ermüdungszwischengrade, d. h. die Werte zwischen 0 und 1, sind dafür nicht entscheidend. Sie geben lediglich an, wie lange es noch dauert, bis ein Pausenerfordenis entsteht. Da sich körperliche Ausdauer nicht spontan ändert, kann davon ausgegangen werden, dass die Ermüdungsraten zuverlässig bestimmt werden können. In der Entwicklung des mathematischen Modells (Kap. 3.2.1) wurde bereits auf entsprechende medizinisch-validierte Testverfahren, wie der 6-Minuten-Gehtest (Harada et al., 1999), hingewiesen.

(Zu c) Diese Annahme ist zum einen damit begründet, dass ein Passant, der zu einer Sitzgelegenheit alloziert ist, die er mit einem geringen Ermüdungsgrad erreicht, dort mutmaßlich trotzdem nicht nur sehr kurz verweilen würde. Zum anderen kann diese Annahme analog zur Annahme (a) begründet werden, dass Passanten auch eine Verweildauerempfehlung befolgen, wenn sich dies für sie als nützlich erweist. Wenn also das Verfahren nur unter der Bedingung funktioniert, dass die Passanten sich an empfohlene Verweildauern halten – die lange genug sind, dass sich die Passanten von einer etwaigen Ermüdung erholen können –, dann ist es für die Passanten vernünftig „vorgeschriebene“ Verweildauern zu befolgen. Prinzipiell können anstelle fixer Verweildauern je Passant aber auch variable Verweildauern verwendet werden. Erholungsraten können, analog zu den Ermüdungsraten aus der Annahme (b), ebenso zuverlässig bestimmt werden, indem die Zeit gemessen wird, bis eine Person sich von einem ermüdeten Zustand regeneriert hat.

(Zu d) Hier wird von den normalen Gehgeschwindigkeiten ausgegangen. Als mittlere Gehgeschwindigkeit bezogen auf eine gewissen Strecke kann diese als konstant betrachtet werden.

(Zu e) Diese Annahme trifft in der Realität gerade dann zu, wenn Unterstützung durch andere Personen fehlt. Genau dies stellt auch die fokussierte Zielgruppe in der vorliegenden Problemstellung dar.

(Zu f) Die Passanten gehen zu einem Zielort und möchten keine Zusatzstrecken bewältigen. Da die Zielgruppe in der vorliegenden Problemstellung Passanten mit körperlichen Beeinträchtigungen ihrer Mobilität zu Fuß betrifft, würden Zusatzstrecken eine erhebliche zusätzliche Anstrengung bedeuten.

(Zu g) Diese Annahme impliziert zweierlei: Zum einen wird davon ausgegangen, dass keine anderen Passanten, außer denen in der Menge *Passanten*, die adaptiven Sitzgelegenheiten nutzen (wollen). Zum anderen bedeutet dies, dass auch keine Passanten in die Diskurswelt eintreten können. Ersteres kann gelöst werden, indem alle Passanten, welche die adaptiven Sitzgelgenheiten nutzen wollen, in die *Passanten*-Menge aufgenommen werden. Technisch könnte dies

bspw. in der Realität mit einer Art Anmeldung umgesetzt werden, ggf. auch in Verbindung mit physischen Blockaden, etwa durch absenkbare Poller, um sicherzustellen, dass die Sitzgelegenheiten nur von Passanten benutzt werden, die dazu eine „Berechtigung“ [39] haben. Zweiteres kann in der Realität so gelöst werden, dass das entwickelte Lösungsverfahren zyklisch wiederholt ausgeführt wird. Hierbei müsste dann allerdings beachtet werden, dass sich durch Eintreten neuer Passanten in die Diskurswelt auch die Allokationen für die bereits in der Diskurswelt enthaltenen Passanten ändern können. Ein Ansatz, um das Eintreten von Passanten in die Diskurswelt zu unterschiedlichen Zeitpunkten zu modellieren wäre, nicht nur die Wegpunkte als mögliche Startpositionen zu verwenden.

(Ad 2) Zu den *Simulationsrahmenbedingungen* ist Folgendes vorzubringen: Die entscheidenden Simulationsparameter wurden variiert und dann die Tendenz aus mehreren Simulationsläufen betrachtet (vgl. Kap. 4.1). Die fix gesetzten Größen des Simulationsszenarios waren (i) die zu bewältigende Strecke, (ii) die Anzahl der SSOs inkl. deren Positionen, (iii) die Sitzplatzkapazität für jedes SSO sowie (iv) die initialen Ermüdungsgrade. Die fixen Parameter der Safety-Bewertungsfunktion sind keine Attribute des Szenarios, sondern durch das Problem begründet. Für keine der Parameter (i) – (iv) sind bei einer (sinnvollen) anderen fixen Festsetzung in der Tendenz grundlegend divergierende Ergebnissen zu erwarten. (i) Die zu bewältigende Strecke ist im Verhältnis zu den Ermüdungsraten der Passanten relevant, welche durch Zufallsziehungen variiert wurden. Die Anzahl der SSOs ist im Verhältnis zur Anzahl an Passanten relevant, die als unabhängige Größe sukzessive erhöht wurde. Durch Erhöhung der Passantenzahl im Vergleich zur Zahl der SSOs ändert sich im Wesentlichen nur, dass der beste erzielbare aggregierte Safety-Gesamtwert abnimmt. Es zeigte sich aber auch, dass die aggregierten Safety-Gesamtwerte, die dabei mit der Heuristik erzielt werden, tendenziell nahe beim höchstmöglichen erzielbaren Safety-Gesamtwert und höher als mit dem intuitiven Verfahren liegen. Durch Erhöhen der (iii) Sitzplatzkapazitäten ist zu erwarten, dass der erzielbare höchstmögliche Safety-Gesamtwert zunimmt (bzw. bei Verringern abnimmt), aber nicht, dass sich die Tendenz der erzielten Safety-Gesamtwerte in Abhängigkeit des Verfahrens und der Knappheit der SSOs grundlegend ändert. Andere (iv) initiale Ermüdungsgrade würden lediglich die genauen Zeitpunkte der Pausenerfordernisse verschieben und ist in der Auswirkung auf die zu erzielende Belegung mit anderen Ermüdungsraten oder Gehgeschwindigkei-

[39] Selbstverständlich kann jeder Passant das Recht haben, eine Parkbank zu benutzen. Mit Berechtigung ist hier gemeint, dass ein Passant zur Nutzung des Systems adaptiver Sitzgelegenheiten angemeldet ist (und zur entsprechenden Sitzgelegenheit alloziert wurde).

ten vergleichbar, die variiert wurden. Daher werden die Ergebnisse als über die konkreten Simulationsrahmenbedingungen hinausgehend gültig erachtet.

(Ad 3) Prinzipiell kann das entwickelte Verfahren – nach Anpassungen – auf alle *anderen Anwendungsfälle* übertragen werden, die durch folgende Eigenschaften charakterisiert sind: Es handelt sich um ein Allokationsproblem von Personen auf Ressourcen. Die Personen haben je Ressource genau einmal die Möglichkeit, diese Ressource zu nutzen oder nicht zu nutzen. Die Nutzung der Ressourcen verbraucht Zeit, genauso wie die Anbahnung der Nutzung. Die Personen haben einen bestimmten Bedarf die Ressourcen zu nutzen. Nicht alle Ressourcen stiften den gleich hohen Nutzen für die Personen. Die Ressourcen können grundsätzlich durch mehrere Personen gleichzeitig genutzt werden, wobei es eine obere Grenze für die Anzahl der Simultannutzer gibt (die aber beliebig hoch sein kann). Es macht dabei einen Unterschied, ob die Ressourcen durch eine oder mehrere Personen genutzt werden, wobei die Nutzung durch nur eine Person vorteilhafter ist. Es macht aber keinen Unterschied ob die Ressourcen von nur zwei oder mehr als zwei Personen gleichzeitig genutzt werden.

Im Kapitel 2.5.2 wurde außerdem dargelegt, dass das Problem der Sitzplatzallokation Analogien zu Smart-Parking-Anwendungsfällen aufweist. Zwar lassen sich die dargelegten Smart-Parking-Lösungen nicht direkt, ohne zum Teil erhebliche Modellanpassungen, auf die Problemstellung der vorliegenden Arbeit anwenden. Umgekehrt lässt sich der in dieser Arbeit entwickelte Lösungsansatz aber für Smart-Parking-Probleme anwenden, z. B. indem für alle Fahrzeuge, die einen Parkplatz suchen, die maximalen Distanzen bis zur „Ermüdung“ gleich, aber größer als die maximal mögliche Distanz zu einem Parkplatz, gesetzt werden und die initialen „Ermüdungsgrade“ so festgelegt werden, dass die Fahrzeuge genau dort „ermüden“, wo ihre präferierte Parkposition liegt. Dann sucht das Verfahren eine solche Parkplatzallokation, bei welcher die maximale Distanz zwischen präferierter und allozierter Parkposition minimal ist. Hierzu könnte dann auch eine symmetrische Safety-Funktion verwendet werden, indem $w_{li} = w_{re}$ gesetzt wird. Die Parkdauer wird durch die Verweildauer r_i angegeben. Die „Sitzplatz-“Kapazität κ ist sinnvollerweise für jeden Parkplatz auf 1 zu setzen.

Der in dieser Arbeit entwickelte Lösungsansatz lässt sich aber auch für Probleme im Zusammenhang mit Elektromobilität anwenden. Auf langen Strecken, auf denen Elektrofahrzeuge zwischendurch geladen werden müssen, kann Koordination vorteilhaft bzw. in gewissem Sinne sogar notwendig sein (García-Magariño et al., 2018; Gusrialdi et al., 2017; Coninx et al., 2014): Unkoordiniertes Verhalten kann dazu führen, dass mehrere Fahrzeuge zur gleichen Ladestation fahren und dort wegen Überbelegung warten müssen, weil sie keine

Energiereserven für eine Weiter- oder Rückfahrt mehr haben. Das entwickelte Lösungsverfahren kann entsprechend parametrisiert werden ($v_i \rightarrow$ Fahrgeschwindigkeit, $r_i \rightarrow$ Dauer eines Ladezyklus, $\kappa = 1, \ldots$) und somit auch einen Beitrag für *Smart-Mobility*-Anwendungen im inter-städtischen Bereich liefern.

5.2 Safety-Engineering-Beitrag für Smart City

Zur Diskussion des Safety-Engineering-Beitrages für Smart City sei auf die in Kapitel 2.3 erläuterten zugrunde gelegten Safety-Zusammenhänge rekurriert. Denn die in dieser Arbeit entwickelte Lösung erfolgte mit einem darauf basierenden Safety-Engineering-Ansatz, was v. a. durch das Safety-Maß $z_i(\tau)$ (Ausdruck 3.14) deutlich wird. Dieses Maß könnte zwar auch ohne Bezug zum Safety-Engineering-Ansatz interpretiert werden, z. B. als ein Maß, welches ausdrückt, ob einem Passanten eine Sitzgelegenheit zur richtigen Zeit bzw. am richtigen Ort zur Verfügung steht. Dies wies eine gewisse semantische Nähe zu Anforderungen an Logistik auf (vgl. Pfohl, 2004, S. 12). Demnach könnte die Funktion auch aus logistischer Sicht interpretiert werden. Dass es in diesem Fall diese überdeckenden Interpretationen aus unterschiedlichen Sichtweisen gibt, muss aber nicht verwundern. Schließlich geht es zwar um unterschiedliche Herangehensweisen, aber um den selben Sachverhalt.

Während eine Betrachtung von vorneherein aus logistischer Sicht voraussetzen würde, dass ein logistisches Problem vorliegt, lieferte die Safety-Engineering-Sicht der vorliegenden Arbeit die Methodik, um überhaupt das Safety-Problem zu identifizeren (welches sich im Nachhinein dann durchaus als ein logistisches Problem darstellen ließe). An dieser Stelle sei betont, dass der verwendete Safety-Engineering-Ansatz nach Leveson (2011) als Engineering-Perspektive gerade eine systematische Problemidentifizierung sowie die entsprechende Lösungsentwicklung unterstützt. Gleichwohl soll auch darauf hingewiesen werden, dass von der gesamten von Leveson (ebd.) beschriebenen Engineering-Methodik ein grundlegender Teil verwendet wurde. Dieser Teil wurde dann für die Problemstellung der vorliegenden Arbeit über die Beschreibung von Leveson (ebd.) hinaus heruntergebrochen und durch die Entwicklung des KI-basierten Informationsverarbeitungsverfahrens ausgeprägt.[40]

Als zentraler Dreh- und Angelpunkt des Safety-Engineering-Ansatzes wurde die Konzeption verwendet, dass ein Safety-Vorfall nur dann auftritt, wenn ein kritischer Umweltzustand und ein kritischer Personenzustand zusammenkommen. Dieser konzeptionelle Zusammenhang, der in Abb. 3.3 dargestellt ist,

[40] Das Informationsverarbeitungsverfahren stellt die Kernlogik eines aktiven Controllers im Sinne von Leveson (2011) dar.

soll nun am Beispiel des Safety-Maßes $z_i(\tau)$ in eine Darstellung mit sog. *Iso-Safety-Linien* überführt werden, um die Tragweite der Konzeption deutlich zu machen. Iso-Safety-Linien sind Isoquanten; in der Mikroökonomik kommen solche bspw. als Indifferenzkurven vor. Die Darstellung der grundlegenden Safety-Konzeption mit Iso-Safety-Linien wird durch die Portfoliodarstellung der Abb. 3.3 nahegelegt, da es offenbar verschiedene Kombinationen aus Personen- und Umweltzuständen gibt, die zum gleichen Safety-Niveau führen müssen. Im Ausdruck 3.14, $z_i(\tau) = 1 - \Big(x_i(\tau) + y_i(\tau)\Big)$, ist $x_i(\tau)$ eine numerische Repräsentation für einen Passantenzustand und $y_i(\tau)$ eine numerische Repräsentation für einen Umweltzustand. Die Iso-Safety-Linien ergeben sich dann durch Umstellen des Ausdrucks, wobei das Safety-Maß $z_i(\tau)$ als fixer Parameter behandelt wird:

$$y_i(\tau)\Big(x_i(\tau)\Big) = 1 - x_i(\tau) - z_i(\tau) \ . \tag{5.1}$$

Hieraus ergeben sich die in Abb. 5.1 dargestellten Iso-Safety-Linien. Jede Iso-Safety-Linie repräsentiert dabei ein Safety-Niveau gem. des Safety-Maßes $z_i(\tau)$, z. B. die untere für $z_i(\tau) = 0,5$. Es sei darauf hingewiesen, dass negative $x_i(\tau)$ oder $y_i(\tau)$ nicht vorkommen können, da sie jeweils eine (normalisierte) Distanz ausdrücken. In Abschnitt 3.2.2 wurde definiert, dass das Safety-Maß von $z_i(\tau) = 0$ das angestrebte Maß darstellt und alle $z_i(\tau) < 0$ einen Safety-Vorfall bedeuten. In der Abb. 5.1 repräsentieren also alle Iso-Safety-Linien einen Safety-Vorfall, die über der Iso-Safety-Linie liegen, welche das Safety-Maß $z_i(\tau) = 0$ repräsentiert und daher hier auch als die Nullline bezeichnet werden kann. Der Mehrwert der Darstellung mit Iso-Safety-Linien gegenüber der Portfoliodarstellung liegt in der quantitativen Interpretierbarkeit. $x_i(\tau) = 1$ bedeutet bspw., dass der Passant i im Zustand τ seit seiner letzten Pause genau die Distanz zurück gelegt hat, die er zurücklegen kann, bis er ein Pausenerfodernis hat. $y_i(\tau) = 1$ bedeutet, dass die Distanz, die ein Passant i im Zustand τ bis zur nächsten Pause noch gehen muss, genau der Distanz entspricht, die er gehen kann, bis er ein Pausenerfordenis hat.

Wenn ein Passant i im Zustand τ bspw. bereits drei Viertel der Distanz zurückgelegt hat, die er ohne Pausenerfordernis zurücklegen kann, also wenn $x_i(\tau) = 0,75$, dann wäre es optimal, wenn er eine Sitzgelegenheit in der Distanz zur Verfügung hat, die einem Viertel der Distanz entspricht, die er ohne Pausenerfordernis zurücklegen kann, also wenn $y_i(\tau) = 0,25$. Ist $y_i(\tau)$ größer, dann kommt es zu einem Safety-Vorfall. Ist $y_i(\tau)$ kleiner, dann kommt es zwar zu keinem Safety-Vorfall, jedoch ist der resultierende Zustand trotzdem unerwünscht. Die oben so genannte Nulllinie repräsentiert also das Spektrum der

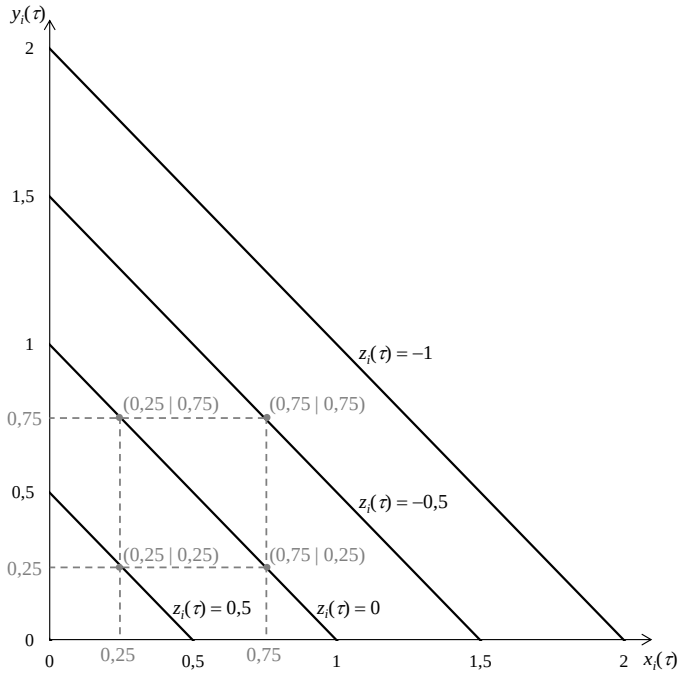


Abbildung 5.1: Iso-Safety-Linien für das Safety-Maß $z_i(\tau)$

Zustandskombinationen, die anzustreben sind. Wenn nur eine Sitzgelegenheit in einer Distanz zur Verfügung steht, die drei Viertel der Distanz entspricht, die i gehen kann, bis er ein Pausenerfordernis hat, dann darf die Distanz, die i bereits zurückgelegt hat nur einem Viertel der Distanz entsprechen, die er gehen kann, bis er ein Pausenerfordernis hat. Ziel dieser Darstellung ist hier nicht die Aufdeckung „verborgener“ Komplexität, sondern wie die Safety-Konzeption grundsätzlich angewendet werden kann, um Problemidentifizierung und dann Lösungsentwicklung zu unterstützen (worin sich i. d. R. ausreichend konstruktionsorientierte Komplexität offenbart).

Die Konzeption des Safety-Maßes kann auch auf andere Anwendungsfälle übertragen werden. Hierzu ist das Safety-Maß so zu spezifizieren, dass es sich aus einer quantitativen Repräsentation des Umweltzustandes und einer quantitativen Repräsentation für die Passantenzustände ergibt. Idealerweise erfolgt die Spezifizierung so – wie im vorliegenden Fall –, dass das Safety-Maß metrisch interpretierbar ist. An die Iso-Safety-Linien kann dann i. Allg. die Fragestellung gerichtet werden, wie sich der Umweltzustand ändern muss, wenn sich der Passantenzustand ändert bzw. auch v. v., um ein bestimmtes Safety-Niveau gem. des gewählten Safety-Maßes zu erhalten. Im vorliegenden

Fall wäre in dieser Hinsicht die ideale Lösung, dass die Sitzgelegenheiten mit dem Passanten quasi mitwandern. Auch wenn diese Ideallösung aus technischer Sicht als zumindest prototypisch durchaus realisierbar betrachtet werden kann, so wurde der Fokus in dieser Arbeit auf einen Lösungsbeitrag durch „reine“ Informationsverarbeitung gelegt. Die Möglichkeit, dass die Passanten eine mobile Sitzgelegenheit einfach mit sich tragen wurde aus diesem Grunde ausgeklammert, weil dadurch die Nutzbarkeit des urbanen Raums an vielen Stellen, z. B. bei Stufen oder für die Nutzung von ÖPNV, mutmaßlich erheblich reduziert wird.

Als ein weiteres Beispiel für die allgemeine Anwendbarkeit der Safety-Konzeption soll die Benutzbarkeit des urbanen Raums für Passanten, die auf einen Rollator angewiesen sind, dienen. Das Angewiesensein dieser Passanten auf einen Rollator ist in Bezug auf Stufen (bzw. Treppen aber auch hohe Bordsteine) oder Bodenrillen (wie z. B. Schienen, Gullis oder taktile Bodenindikatoren) kritisch. Im Gegenzug stellen Stufen und Bodenrillen einen kritischen Umweltzustand für Rollatornutzer dar. Diese Gegenseitigkeit ist ein Ausdruck des Dualismus zwischen Umweltzustand und Passantenzustand. Um zu verhindern, dass der kritische Personenzustand (Angewiesensein auf einen Rollator) und ein kritischer Umweltzustand (Stufen, Bodenrillen) zusammenkommen, gibt es mehrere grundsätzliche Möglichkeiten, wie bspw. die Errichtung von Aufzügen, Rampen oder, für die Bodenrillen, Abdeckungen. Eine andere Möglichkeit könnte auch darin bestehen, die urbane Umgebung so auszustatten, dass die Passanten keinen Rollator mehr benötigen – etwa durch Handläufe zum Festhalten und Stützen oder, im Sinne der Arbeit, durch bedarfsgerechte Sicherstellung der Verfügbarkeit geeigneter Sitzgelegenheiten. Wenn die Passanten keinen Rollator mehr benötigen, dann wird ihr Zustand in Bezug auf Stufen und Bodenrillen *unkritisch*.

Es gibt aber für dieses Beispiel auch einen auf reiner Informationsverarbeitung basierenden Lösungsansatz. Ein spezielles Navigationssystem könnte die Passanten über einen Weg führen, der keine Stufen oder Bodenrillen enthält. Das Navigationssystem würde also versuchen, durch Informationsverarbeitung aktiv zu verhindern, dass die Passanten, die sich aufgrund ihrer Rollatoren in dem definierten kritischen Zustand befinden, nicht auf einen entsprechend definierten kritischen Umweltzustand treffen. Dieses Beispiel soll zeigen, dass der Safety-Engineering-Ansatz die Suche nach Lösungen für generationengerechte, inklusive Smart Cities systematisch unterstützen kann, indem er zunächst einmal den Problemraum öffnet, in welchem dann anschließend Lösungsansätze erkundet werden können. Aus einer Information-Systems-Engineering-Perspektive, welche der vorliegenden Arbeit übergeordnet zugrunde liegt, lautet die Leitfrage hierfür dann: Wie kann Informationsverarbeitung dazu bei-

tragen, dass ein kritischer Personenzustand und ein kritischer Umweltzustand nicht zusammenkommen?

5.3 Einordnung der Künstlichen Intelligenz

Abschließend soll noch die Einordnung der vorliegenden Lösung in das Methodenspektrum der KI reflektiert werden. KI kann unterschiedlich ausgerichtet werden und es kann unterschiedliche Auffassungen davon geben, was KI im Kern ausmacht. Für Russell und Norvig (2003, S. 28) bestehen bspw. richtungsweisende Unterschiede darin, ob sich KI auf Denken oder Verhalten bezieht und ob KI auf Modellierung von Menschen oder das Abbilden und Arbeiten mit Idealen – z. B. „reine“ Rationalität – ausgerichtet ist. Unabhängig von einer Einordnung in eine dieser Richtungen ist KI gegenwärtig sehr stark mit sog. maschinellem Lernen assoziiert und tlw. sogar damit gleichgesetzt. Die vorliegende Arbeit entwickelte aber weder, noch verwendete maschinelle Lernverfahren. Nach Auffassung der vorliegenden Arbeit ist maschinelles Lernen zwar eine bedeutende Methode der KI, aber eine, die neben weiteren steht. Luger (2009, S. 30) schließt bspw. neben Lernen auch Schlussfolgern, Mustererkennung sowie weitere Formen der Inferenz in die Methoden der KI ein, wenn diese durch Computer, also durch IT, automatisiert ausgeführt werden. Im Folgenden sollen *knapp* Ausrichtungen und Auffassungen der KI anhand von vier Dimensionen (I – IV) mit jeweils zwei polar zu verstehenden Ausprägungen (a & b) diskutiert werden. Die Dimensionen sind dabei bewusst exemplarisch für die Argumentation kontrastierend gewählt. Dies wird nach deren Beschreibung dargelegt.

(I) Zielstellung: Die Methoden der KI können auf der einen Seite das allgemeine Ziel haben, (a) menschliche Kognition nachzubilden. Auf der anderen Seite können sie auf (b) das Lösen bestimmter Probleme ausgerichtet sein. Mit (a) der Nachbildung menschlicher Kognition erhofft man sich i. Allg., dass Maschinen Aufgaben übernehmen können, die aufgrund seiner kognitiven Fähigkeit Menschen vorbehalten waren. Dies betrifft viele Anwendungen, die Bild- oder Spracherkennung und situatives Verhalten erfordern. Bei (b) dem Lösen bestimmter Probleme werden dahingegen spezifische Vorteile von IT gegenüber menschlicher Kognition gezielt ausgenutzt, um bestimmte Probleme „intelligenter“ zu lösen als Menschen es vermögen. Dies betrifft i. d. R. höhere Effizienz durch erheblich schnellere Berechnungen sowie höhere Effektivität durch höherer Präzision. Zwischen (a) und (b) lassen sich nahezu beliebige Zwischenpositionen einordnen. Für ein KI-basiertes System ist i. d. R. eine an der eigentlichen Problemstellung ausgerichtete Austarierung zwischen naturgetreu-

er Nachbildung menschlicher Kognition und Nutzung informationstechnisch-spezifischer Überlegenheit gegenüber menschlicher Kognition angebracht. Beispiele, die sich Ausprägung (a) zuordnen lassen, sind beschrieben in: Hassabis et al. (2017), Tokic (2013) und Frantz (2003). Beispiele, die sich Ausprägung (b) zuordnen lassen, sind beschrieben in: Zimmermann et al. (2012), Timm und Lattner (2010) und Smirnov und Shilov (2010).

(II) Informationsverarbeitungsstruktur: Gegenwärtige KI-Ansätze beziehen sich sehr oft auf solche mit sog. (a) subsymbolischer Informationsverarbeitungstruktur. Subsymbolische Strukturen werden auch als konnektionistisch bezeichnet, da sie auf der Vernetzung einer Vielzahl kleiner abstrakter Informationsverarbeitungseinheiten beruhen. Hierbei stellen künstliche neuronale Netze die prominenteste Ausprägung dar. Diese haben die Eigenschaft, dass den künstlichen Neuronen als kleinste Informationsverarbeitungseinheiten i. d. R. keine Semantik zugeordnet werden kann. Die genaue Rolle der einzelnen künstlichen Neuronen sowie i. d. R. auch die Art und Weise ihrer Vernetzung ist üblicherweise weder a priori noch a posteriori bekannt. Die subsymbolische Informationsverarbeitung ist daher normalerweise praktisch nicht nachvollziehbar. Bei Unterbrechung der Verarbeitung kann aus dem Systemzustand i. Allg. kein Teilwissen abgeleitet werden. Dem werden sog. (b) symbolische Informationsverarbeitungsstrukturen gegenübergestellt. Symbolische Informationsverarbeitung entspricht logisch-mathematischer Schlussfolgerung. Die Informationsverarbeitungseinheiten sind semantisch aufgeladene Symbole bzw. Variablen in mathematischem Sinne, dessen Bedeutung bekannt ist. Dadurch kann im Prinzip jeder Verarbeitungsschritt symbolischer Informationsverarbeitung expliziert werden. Beispiele, die sich Ausprägung (a) zuordnen lassen, sind beschrieben in: Cottrel et al. (2012) und Zimmermann et al. (2012). Beispiele, die sich Ausprägung (b) zuordnen lassen, sind beschrieben in: Giesl (2010), Egly und Haller (2010) und Bench-Capon und Dunne (2007).

(III) Maschinelles Lernen: Eine klassische Zielstellung (a) maschinellen Lernens ist, dass IT automatisiert seine anhand einer Kennzahl gemessene Leistung für eine Aufgabenklasse durch steigende Erfahrung erhöht (Mitchell, 1997, S. 2). Steigende Erfahrung bedeutet in diesem Zusammenhang wiederholte Ausführung eines Computerprogramms für Probleme derselben Klasse. Der Einsatz maschinellen Lernens kann v. a. dann vorteilhaft sein, wenn er aufwändige oder gar unmögliche explizite Programmierung in angemessener Weise überflüssig macht. Gemäß der genannten Zielstellung kann maschinelles Lernen auch als Anwendung statistischer Methoden ausgeprägt werden (Hastie et al., 2013). Hierbei werden, der Natur der Statistik gemäß, stets sehr viele Daten benötigt. Dieser Nachteil trifft aber auf so gut wie alle maschinellen Lernverfahren zu. Der gegenwärtige Erfolg der meisten maschinellen Lernver-

fahren geht auf die Verfügbarkeit einer schieren Unmenge von Daten zurück. Die Verfahren selber müssen dabei nicht unbedingt kompliziert oder neuartig sein. Nach Auffassung der vorliegenden Arbeit bedarf ein IT-System (b) nicht zwangsläufig maschinellen Lernens, um als *KI*-System zu gelten. Maschinelles Lernen im Sinne der o. g. Definition ist von empirischen Daten abhängig. Intelligenz als ein besonderes Informationsverarbeitungsvermögen ist allerdings im Kern unabhängig davon zu betrachten. Die Verfügbarkeit von Daten macht nicht die Intelligenz deren Verarbeitung zu Informationen aus. Als intelligente Informationsverarbeitung kann eine solche bezeichnet werden, die komplexe Probleme angemessen löst (vgl. Untertitel von Luger, 2009). Komplexe Probleme sind – um mit Leveson (2011, S. 4) zu sprechen: – Probleme, welche (von Menschen) intellektuell schwer zu handhaben sind. (Die Schwierigkeit der Handhabung könnte dabei damit bestimmt werden, dass eine Vielzahl von Menschen entweder zu keiner, zu einer falschen, zu einer ungenauen oder nur nach unangemessen langer Zeit zu einer Lösung gelangen.) Beispiele, die sich Ausprägung (a) zuordnen lassen, sind beschrieben in: Tokic (2013), Cottrel et al. (2012) und Samuel (1959). Beispiele, die sich Ausprägung (b) zuordnen lassen, sind beschrieben in: Bernardini et al. (2018), Egly und Haller (2010) und Padgham und Lambrix (2005).

(IV) Formalisierbarkeit des zugrundeliegenden Sachverhalts: Damit IT Sachverhalte verarbeiten kann, müssen diese in eine digitale Repräsentation überführt werden. Letztlich arbeitet IT auf binär kodierten Ausdrücken. Ein für die Informatik von Anfang an elementares Problem ist daher die Frage, wie ein Sachverhalt so repräsentiert werden kann, dass zum einen die eindeutige Abbildung auf eine binäre Zeichenfolge und zum anderen eine zweifelsfreie Interpretation binär kodierter Zeichenfolgen möglich ist. (a) Formalisierbarkeit soll hier vorläufig als eine solche Überführbarkeit verstanden werden. Weiter unten wird auf das Verständnis von Formalisierungen noch einmal eingegangen. Es gibt auch Sachverhalte, die sich bislang (b) nicht in der soeben geforderten Weise adäquat formalisieren lassen. Damit sind nicht nur ideell-gedankliche Sachverhalte, wie Intuition und Liebe gemeint, sondern v. a. auch physikalisch-sinnliche Sachverhalte, wie der Vorgang des Sehens oder Hörens und sogar logisch-semantische Sachverhalte, wie der Vorgang des Verstehens geschriebenen Textes, selbst wenn er binär kodiert ist.

Bevor nun, wie oben angekündigt, erläutert werden soll, inwiefern die Dimensionen bewusst kontrastierend für die Argumentation der KI-Auffassung für die vorliegende Arbeit sind, sei betont, dass die Dimensionen tatsächlich dimensional, begrifflich voneinander unabhängig zu verstehen sind. Zwar ergeben sich natürlicherweise miteinander korrelierende Ausprägungen von Dimensionen. Keinesfalls bedingt aber eine Ausprägung einer Dimension eine bestimmte

Ausprägung einer anderen. Dies soll an drei exemplarisch ausgewählten Dimensionspaaren klar gemacht werden.

Zielstellung × Informationsverarbeitungsstruktur: Um menschliche Kognition nachzubilden, erscheinen künstliche neuronale Netze als subsymbolische Ansätze naheliegend. Allerdings kann sich die Nachbildung menschlicher Kognition auch auf rein logische Aspekte beziehen. Dann kann menschliche Kognition mit symbolischen Ansätzen untersucht werden. Die formale Logik befasst sich ja gerade mit der Aufstellung expliziter Regeln für das Denken als Bestandteil der menschlichen Kognition (vgl.: Luger, 2009, S. 9 ff.; Russell und Norvig, 2003, S. 194 f.; Kant, 1966, S. 22 f.). Umgekehrt werden spezifische Probleme sowohl mit symbolischen als auch mit subsymbolischen Ansätzen der KI zu lösen versucht, z. B. Probleme der Logistik (Timm und Lattner, 2010).

Informationsverarbeitungsstruktur × Maschinelles Lernen: Die Anwendung subsymbolischer Ansätze ist sehr stark mit maschinellem Lernen korreliert. Allerdings ist maschinelles Lernen kein zwingender Bestandteil subsymbolischer Informationsverarbeitung. Ein künstliches neuronales Netz könnte bspw. vollständig manuell so konfiguriert werden, dass es die geforderte Aufgabe angemessen löst. Dies ist z. B. für logische Grundoperationen recht einfach möglich (McCulloch und Pitts, 1943; Luger, 2009, S. 455 ff.). Umgekehrt gibt es auch maschinelle Lernverfahren für symbolische Ansätze, z. B. *generalization learning*, *rote learning* (Samuel, 1959) oder *naive-Bayes*-Klassifikatoren (Russell und Norvig, 2003, S. 718; Luger, 2009, S. 184 ff.).

Maschinelles Lernen × Formalisierbarkeit des zugrundeliegenden Sachverhalts: Aus mangelnder Formalisierbarkeit folgen nicht zwingend subsymbolische Ansätze. Dies ist zum einen damit begründet, dass mangelnde Formalisierbarkeit auch einen momentanen Mangel an Verständnis eines Sachverhalts widerspiegeln kann. Es ist z. B. momentan nicht vollends verstanden, wie einzelne Objekte auf einem Bild so zuverlässig erkannt werden können, wie es Menschen vermögen. Dennoch können symbolische Ansätze für dieses Problem angewendet werden, die dann eben unzuverlässiger sind. Ein Ansatz besteht bspw. darin, in einem Bild Objekte entlang hoher Farbgradienten abzugrenzen (Canny, 1986). Dieser Ansatz ist mathematisch-logisch klar explizierbar. Umgekehrt können subsymbolische Informationsverarbeitungsansätze auch für formalisierbare Probleme angewendet werden, z. B., wie im vorherigen Absatz erwähnt, für logische Grundoperationen, die an und für sich ja bereits formal vorliegen.

Für die übrigen Dimensionenspaare sei es nun dem interessierten Leser überlassen, sich von deren paarweisen Unabhängigkeit zu überzeugen. Trotz der prinzipiellen begrifflichen Unabhängigkeit besteht oftmals eine Art natürliche Korrelation zwischen bestimmten Dimensionsausprägungen. Gegenwärtig wird

KI sehr stark mit *Deep Learning* assoziiert (LeCun et al., 2015). Deep Learning wird v. a. für das Problem des Bilderkennens bzw. Bildverstehens eingesetzt. Der Ansatz beruht auf (I a) einer Nachbildung menschlicher, visueller Kognition, (II a) subsymbolischer Informationsverarbeitung und (III a) maschinellem Lernen, weil (IV b) das zugrundeliegende Problem bislang nicht hinreichend formalisiert ist. Der Lösungsansatz der vorliegenden Arbeit ordnet sich aber in allen Dimensionen genau der anderen Ausprägung zu. Es geht um (I b) das Lösen eines speziellen Problems mit (II b) symbolischer Informationsverarbeitung (III b) ohne maschinellem Lernen für ein (IV a) formalisierbares Problem.

Trotz dieser kontrastierenden Einordnung wurde die Problemstellung mit Methoden gelöst, die der KI zuzuordnen sind. Dass sich KI gegenwärtig verstärkt mit der automatisierten Verarbeitung nicht-formalisierter Sachverhalte befasst, heißt nicht, dass andere Fragestellungen zur Erzeugung künstlicher Intelligenz obsolet wären. Bei genauerer Überlegung, was denn Intelligenz für Maschinen wirklich ausmachen könnte, ist so etwas wie Bilderkennung sicherlich eine relevante Vorbedingung. Nach Auffassung der vorliegenden Arbeit ist dies aber nicht kennzeichnend für den Kern einer KI. Einige Tiere verfügen über höhere Sehfähigkeiten als Menschen und dennoch gelten sie dadurch nicht als intelligenter. Genauso wenig nimmt bei Menschen deren Intelligenz ab, wenn sie erblinden. Das Sehen ist eine Funktion eines Sinnesorgans, um den Sehenden ein möglichst genaues Bild ihrer Umwelt zu verschaffen. Die eigentliche Intelligenz, z. B. das Verstehen oder das Antizipieren[41] von Szenen, liegt im Verstandesapparat.

In diesem Sinne soll, wie oben versprochen, noch einmal kurz auf den Begriff der Formalisierbarkeit eingegangen werden. Oben wurde der Begriff informationstechnisch operational konkretisiert. Im weiteren Sinne aber bedeutet *Form*alisierung, dass ein Sachverhalt in eine bestimmte *Form* gebracht wird – und zwar in eine Form des *stringenten Denkens*. Stringentes, reines Denken bezeichnet hier die vom Sinnesapparat losgelöste Ebene des Denkens. Damit ist nicht gemeint, dass der Sinnesapparat überflüssig ist. Damit ist gemeint, dass das reine Denken auf Daten arbeitet, die aus dem Sinnesapparat als Perzeption kommen, aber genauso gut einer inneren Vorstellungskraft entspringen können. Logik und Mathematik ist reines Denken. Oftmals ist dies durch empirische Wahrnehmungen inspiriert. Die logisch-mathematische Aussagen sind aber davon unabhängig gültig.[42]

[41] Gerade das Antizipieren und *Planen* der Zukunft gilt als eine evolutionäre Ausprägung von Intelligenz.

[42] Die Aussage über die Länge der Diagonalen in einem Quadrat mit Seitenlänge 1 ist beispielsweise notwendigerweise wahr, obwohl der Wert $\sqrt{2}$ der Diagonalenlänge kein empirischer ist. Und doch ist die Aussage von erhebliche Bedeutung für die empirisch reale

Wie ein in eine Form des stringenten Denkens gebrachter Sachverhalt genau aufgeschrieben wird, ist für dessen Formalisierbarkeit nicht das Entscheidende. Es geht also nicht darum, dass formalisierte Sachverhalte einer bestimmten mathematischen Notationskonvention folgen müssen. (Da aber die Logik und Mathematik in diesem Sinne die reinste Form des Denkens darstellt, ist es kein Zufall, dass verschriftlichte Formalisierungen mathematischen Notationskonventionen folgen.) Das eigentlich Entscheidende ist, dass es möglich ist, einen Sachverhalt so zu beschreiben, dass darüber in mathematisch-logischer Form, d. h. *vernünftig* in rigorosem Sinne, geredet werden kann. Das „vernünftige Reden über einen Sachverhalt“ ist in der Auffassung der vorliegenden Arbeit Ausdruck von Intelligenz, wobei hier Reden auch nach innen gerichtet sein kann, also ein explizites Nachdenken darstellt. Mit vernünftig ist gemeint, dass dieses Nachdenken auf der rigorosen Nutzung des Verstandes beruht. Die Schwierigkeit näher zu spezifizieren, was wiederum mit rigoroser Nutzung des Verstandes gemeint ist, zeigt, dass auch hier noch nicht alles vollends begriffen ist und somit die KI diesbezüglich noch nicht abgeschlossen ist. Es sei darauf hingewiesen, dass das oben als vorläufig eingeführte Verständnis von Formalisierbarkeit auch in der eben erläuterten Hinsicht weiterhin zutrifft. Für IT-Systeme ist die Notationskonvention für das „vernünftige Reden“ über einen Sachverhalt letztlich die binäre Kodierung.

Im diesem Sinne erzeugt die vorliegende Arbeit eine Verfahrensweise, wie ein IT-System automatisiert über einen Sachverhalt, der bereits in die für IT passende Form eines künstlichen Denkens gebracht ist, nachdenken kann. Im Übrigen ist die Definition eines zu lösenden Problems sowie die Generierung eines Lösungsverfahrens bislang ausschließlich eine menschliche Leistung. Die Vor-Formalisierung durch einen Menschen kann daher, gegenwärtig, kaum ein Argument dagegen sein, dass ein „echtes“ KI-Verfahren zum Einsatz kommt. Auch in subsymbolischen Ansätzen mit maschinellem Lernen, wie beim Deep Learning, werden die Verfahrensweise und Problemlösungskriterien *formal vorgegeben* (Optimierung einer Kennzahl auf einer Graphstruktur, dem künstlichen neuronalen Netz).

Das Ziel der vorliegenden Arbeit lag darin, eine Verfahrensweise zu konstruieren, die als vernünftig bezeichnet werden kann. Dies ist mit der Auffassung von Russell und Norvig (2003, S. 28) konform, die für KI *rationales* (d. h. vernünftiges) Handeln in der Vordergrund stellen. Rationalität bedeutet dabei, dass in einer Situation die nach bestimmten Kriterien bestmögliche Handlung gewählt wird (ebd.). Für die vorliegende Arbeit wurde als Kriterium für

Welt. Die Bezeichnung von $\sqrt{2}$ als irrationale Zahl ist Übrigen in dem Sinne unglücklich, als dass die Zahl eigentlich nur rational, durch „reines“ Denken, begreifbar ist.

bestmögliche Handlungen das Maximin-Kriterium begründet verwendet. Zum vorliegenden Rationalitätsverständnis von Russell und Norvig (ebd.) gehört ebenso die Situation der Handlungswahl. Auch für die Problemstellung der vorliegenden Arbeit kann es sein, dass die Suche der Maximin-Lösung nicht die bestmögliche Handlung darstellt. Dies ist insb. der Fall, wenn das Herausfinden der exakten Lösung zu lange dauert, was bei Vorliegen kombinatorisch vieler Alternativen auch bei Ausführung durch IT regelmäßig eintritt. Dann müssen Verfahren bestimmt werden, die *vernünftigerweise* zu *zufriedenstellenden* Lösungen führen. Genau diese Art der Lösungssuche ist gemäß Luger (2009, S. 31) ein Kernbestandteil der KI-Methodik.

6 Zusammenfassung und Ausblick

Die vorliegende Arbeit lieferte einen Beitrag zur Herstellung von Adaptivität des urbanen Raums an individuelle Erfordernisse von Passanten. Dies soll die sichere Benutzbarkeit und somit die Inklusivität des urbanen Raumes erhöhen. Vor allem Passanten mit körperlichen Beeinträchtigungen meiden den urbanen Raum aufgrund mangelnder Benutzbarkeit. Eine typische Gruppe von Personen mit beeinträchtigter Mobilität zu Fuß sind hochbetagte Menschen. Zurückgehende motorische Fähigkeiten führen dazu, dass sie beim Gehen ein verstärktes Erfordernis nach Ausruhmöglichkeiten haben. Hieraus entsteht das Problem, die Verfügbarkeit geeigneter Sitzgelegenheiten zu koordinieren. Als besonders geeignete Sitzgelegenheiten werden Sitzgelegenheiten konzipiert, die durch Adaption ihrer Höhe der Sitzfläche Passanten aktive Aufsteh- und Hinsetzassistenz bieten können. Der Beitrag der vorliegenden Arbeit lässt sich wie folgt in einen (1.) konstruktionsorientierten Gesamtbeitrag, einen (2.) methodischen Kernbeitrag und einen (3.) übergeordneten methodischen Beitrag aufgeschlüsselt darstellen:

1. *Konstruktionsorientierter Gesamtbeitrag*: Adaptive Sitzgelegenheiten als SSOs zur Erhöhung der sicheren Benutzbarkeit des urbanen Raums für Passanten mit Mobilitätsbeeinträchtigungen.
   - Die Konstruktion der Adaptivität bzgl. Verfügbarkeit und die Konzeption von Adaptivität bzgl. Höhe der Sitzfläche trägt adaptive Sitzgelegenheiten als neuartige SSOs bei.
2. *Methodischer Kernbeitrag*: KI-Verfahren für die Allokation geeigneter Sitzgelegenheiten, um deren erfordernisgerechte Verfügbarkeit zu koordinieren.
   - Das Verfahren implementiert eine Heuristik für eine KI des SSO-Systems „Adaptive Sitzgelegenheiten“.
3. *Übergeordneter methodischer Beitrag*: Transfer eines Safety-Engineering-Ansatzes zur Erhöhung der sicheren Benutzbarkeit des urbanen Raums mittels SSOs.
   - Die Arbeit zeigt anhand des SSOs „Adaptive Sitzgelegenheit“ exemplarisch, wie der Safety-Engineering-Ansatz von Leveson (2011) für

die Safety-orientierte Konstruktion eines adaptiven urbanen Raumes angewendet werden kann.

Für den Lösungsansatz der vorliegeneden Arbeit wurde der Zusammenhang zugrunde gelegt, dass ein Safety-Vorfall genau dann eintritt, wenn ein kritischer Personenzustand und eine kritischer Umweltzustand zusammenkommen. Dieser Zusammenhang wurde insbesondere für die Konstruktion einer Safety-Funktion angewendet.

Die Problemstellung wurde mathematisch formuliert und es wurden Algorithmen zur Lösung der Problemstellung entwickelt. Die Zielgröße wird durch die entwickelte Safety-Funktion gegeben, welche nach wohlfahrtsökonomischen Kriterien aggregiert wird. Die simulationsbasierte Validierung des entwickelten KI-basierten Verfahrens belegte die Effektivität, v. a. im Vergleich zu einer intuitiven Benutzung der nächstmöglichen Sitzgelegenheit, sobald ein Pausenerfordernis auftritt.

Das entwickelte Verfahren ist unter den gesetzten und begründeten Annahmen effektiv. Es kann aber bspw. noch im Hinblick auf den Umgang mit nicht genau bekannten Ankunftszeiten an Sitzgelegenheiten sowie Verweildauern weiterentwickelt werden. Außerdem kann die implizit getroffene *Closed World Assumption* in eine *Open World Assumption* umgewandelt werden. Beides zieht tiefergreifende Überlegungen nach sich, die zwar methodisch interessant sind, den Rahmen der vorliegenden Arbeit aber gesprengt hätten.

Anhang: Quellcode (Java)

Nachfolgend ist der verwendete Quellcode aufgeführt, auf den in den Kapiteln 3 und 4 Bezug genommen wird. Dies soll der Nachvollziehbarkeit der Implementierung des entwickelten Verfahrens sowie auch der Reproduzierbarkeit der Simulationsergebnisse dienen.

Main.java

```
1  package main_package;
2
3  /**
4   *
5   * @author Marvin Hubl
6   * @since 2019-10-26
7   */
8
9  public class Main {
10
11     public static void main(String[] args) {
12
13         runSimulation();
14
15     }
16
17     public static void runSimulation() {
18
19         int maxPedestrianAmountEx  = 5;    // MAX. AMOUNT OF PEDESTRIANS
20                                            // FOR EXHAUSTIVE SOLUTION.
21
22         int maxPedestrianAmountAll = 15;   // MAX. AMOUNT OF PEDESTRIANS
23                                            // FOR NON-EXHAUSTIVE
24                                            // SOLUTIONS.
25
26         int minPedestrianAmount    = 1;    // MIN. AMOUNT OF PEDESTRIANS.
27
28         int runAmount              = 250; // NUMBER OF SIMULATION RUNS /
29                                            // DRAWINGS.
30
31         Director.executeSimulationCases(maxPedestrianAmountEx,
32                                         minPedestrianAmount,
33                                         runAmount,
34                                         true,
35                                         true,
36                                         true);
37
38         System.out.println("Wait 1 second...");
39         Director.sleep(1000);
40
41         Director.executeSimulationCases(maxPedestrianAmountAll,
```

```
42                                          maxPedestrianAmountEx + 1,
43                                          runAmount,
44                                          false,
45                                          true,
46                                          true);
47      }
48  }
```

Director.java

```
1   package main_package;
2
3   import java.util.Random;
4
5   import util.Logger;
6
7   /**
8    * Creates the validiation cases.
9    * @author Marvin Hubl
10   * @since 2019-10-18
11   */
12  public class Director {
13
14      /**
15       * Left and right boundary position of a scenario case, both
16       * initialized with 0.0 but are reset in the case creation methods.
17       * (They can be accessed from 'anywhere'.)
18       */
19      public static double LEFT_POS  = 0.0;
20      public static double RIGHT_POS = 0.0;
21
22      /**
23       * Critical level of fatigue for recording scoring variable x1
24       * (distance walked with critical degree of fatigue).
25       * (It can be accessed - and changed - from 'anywhere'.)
26       */
27      public static double CRITICAL_FATIGUE = 1.0;
28
29      /**
30       * The parameters for the gaussian-based sigmoid safety function,
31       * i.e. for the safety function w.r.t. to appropriate availability.
32       */
33      public static double Z_LE = -1.0;
34      public static double S_LE =  0.2;
35      public static double Z_RI =  1.0;
36      public static double S_RI =  0.75;
37
38      /**
39       * Weight factors for the composite safety function
40       */
41      public static double LAMBDA_A = 0.7;
42      public static double LAMBDA_B = 0.3;
43
44      /**
45       * Values for the safety function w.r.t. the possibility to use
46       * assistance.
47       */
48      public static double GAMMA_SIT_DOWN = 0.5;
49      public static double GAMMA_STAND_UP = 0.5;
50
```

```
51      /**
52       * Following four blocks:
53       * Parameters for the scoring function of the class ScoreRecord, the
54       * target variables x1 to x4 and weights. (Note this is another
55       * scoring function than in the class SafetyScoreRecord and
56       * potentially not used.)
57       * Remember: x1 = Distance that pedestrian walked with critical
58       *               fatigue (should be minimal)
59       *           x2 = Number of how often a pedestrian starts/ends using
60       *               an SUO while someone else is using it, hence
61       *               assistance is deactivated (should be minimal)
62       *           x3 = Duration for accomplishing the course (should be
63       *               minimal or ignored for the score)
64       *           x4 = Fatigue at the end of the course (should be
65       *               minimal)
66       *
67       * (They can be accessed from 'anywhere'.)
68       */
69      public static double X1_OPT = 0.0; // Equals x1_min, i.e. always 0.0
70      public static double X1_MIN = 0.0; // Always 0.0
71      public static double X1_MAX = 0.0; // --> Total distance of the
72                                         //     course
73      public static double X1_WEIGHT = 1.0;
74
75      public static double X2_OPT = 0.0; // Equals x2_min, i.e. always 0.0
76      public static double X2_MIN = 0.0; // Always 0.0
77      public static double X2_MAX = 0.0; // --> 2x number of SUOs
78      public static double X2_WEIGHT = 1.0;
79
80      public static double X3_OPT = 0.0; // Equals x3_min, i.e. always 0.0
81      public static double X3_MIN = 0.0; // Always 0.0
82      public static double X3_MAX = Double.MAX_VALUE; // --> ignore
83      public static double X3_WEIGHT = 0.0;
84
85      public static double X4_OPT = 0.0; // Equals x4_min, i.e. always 0.0
86      public static double X4_MIN = 0.0; // Always 0.0
87      public static double X4_MAX = 0.0; // --> 100%
88      public static double X4_WEIGHT = 1.0;
89
90      /**
91       * Number of maximum permitted expansions in the meta heuristic.
92       */
93      public static long C = Integer.MAX_VALUE;
94
95      /**
96       * Sets the basic settings for simulation cases:
97       *
98       * LEFT_POS --> 0.0,
99       * RIGHT_POS --> 48000.0,
100      * GAMMA_SIT_DOWN --> 0.5,
101      * GAMMA_STAND_UP --> 0.5,
102      * LAMBDA_A --> 0.7,
103      * LAMBDA_B --> 0.3,
104      * (Z_LE | S_LE) --> (-1 | 0.2 ),
105      * (Z_RI | S_RI) --> ( 1 | 0,75),
106      *  C --> 10,000.
107      *
108      */
109     public static void baseSettings() {
110
```

```
111         LEFT_POS = 0.0;
112         RIGHT_POS = 48000.0;
113
114         GAMMA_SIT_DOWN = 0.5;
115         GAMMA_STAND_UP = 0.5;
116
117         LAMBDA_A = 0.7;
118         LAMBDA_B = 0.3;
119
120         Z_LE = -1.0;
121         S_LE =  0.2;
122         Z_RI =  1.0;
123         S_RI =  0.75;
124
125         C = 2500;
126
127     }
128
129     /**
130      * Creates an array of SUOs for the simulation scenario:
131      *
132      * --> 3 SUOs with capacity as specified above and positions 8000.0,
133      *     26000.0, 45000.0.
134      * --> Seat capacity = 2, for all SUOs.
135      *
136      * @return Array of SUOs for a simulation scenario.
137      */
138     public static SUO_AdaptiveSeating[] createSUOs() {
139
140         int kappa = 2;   // Seat capacity (same for all SUOs).
141
142         SUO_AdaptiveSeating suo1 = new SUO_AdaptiveSeating();
143         suo1.position = 8000.0;
144         suo1.capacity = kappa;
145
146         SUO_AdaptiveSeating suo2 = new SUO_AdaptiveSeating();
147         suo2.position = 26000.0;
148         suo2.capacity = kappa;
149
150         SUO_AdaptiveSeating suo3 = new SUO_AdaptiveSeating();
151         suo3.position = 45000.0;
152         suo3.capacity = kappa;
153
154         return new SUO_AdaptiveSeating[] {suo1, suo2, suo3};
155     }
156
157     /**
158      * Creates an array of pedestrians based on the specified random
159      * value drawings and the specified run count.
160      * @param randomValueDrawings The values for initializing the
161      *        pedestrians.
162      * @param runCount The run count in a batch of simulation runs.
163      * @return Array of pedestrians for a simulation run.
164      */
165     public static Pedestrian[] createPedestrians(
166                         RandomValueDrawings randomValueDrawings,
167                         int                 runCount) {
168
169         int numberOfPedestrians = randomValueDrawings.directions.length;
170
```

```
171         Pedestrian[] pedestrians = new Pedestrian[numberOfPedestrians];
172
173         for (int i = 0; i < numberOfPedestrians; i++) {
174
175             pedestrians[i] = new Pedestrian();
176
177             pedestrians[i].direction =
178                         randomValueDrawings.directions[i][runCount];
179             // Start position depending on the pedestrian's walking
180             // direction.
181             if (randomValueDrawings.directions[i][runCount] == 1) {
182                 pedestrians[i].position = Director.LEFT_POS;
183             }
184             else { // direction is -1
185                 pedestrians[i].position = Director.RIGHT_POS;
186             }
187
188             pedestrians[i].normalVelocity =
189                         randomValueDrawings.velocities[i][runCount];
190
191             pedestrians[i].fatigueRate =
192                        randomValueDrawings.fatigueRates[i][runCount];
193
194             pedestrians[i].pauseTime =
195                         randomValueDrawings.pauseTimes[i][runCount];
196         }
197
198
199         return pedestrians;
200
201     }
202
203     /**
204      * Creates and returns an array of random value drawings. The
205      * lengths of the array will be the number of different amounts of
206      * pedestrians for a scenario execution resp. simulation run. The
207      * attributes of the random value drawings objects will be 2D arrays
208      * where the length of the first component is the number of
209      * pedestrian for the corresponing execution run and the length of
210      * the second component is the number of different value drawings
211      * (i.e.: usually the length of the first component will be
212      * different in the returned random value drawing object array and
213      * the the length of the second component will be the same for all).
214      *
215      * @param maxAmountOfPedestrians  The upper boundary for the number
216      *                                of pedestrians.
217      * @param minAmountOfPedestrians  The lower boundary for the number
218      *                                of pedestrians.
219      * @param numberOfRandomDrawings  The number of drawings per amount
220      *                                of pedestrians.
221      * @return Array of random value drawings.
222      */
223     public static RandomValueDrawings[] createRandomValueDrawings(
224                                             int maxAmountOfPedestrians,
225                                             int minAmountOfPedestrians,
226                                             int numberOfRandomDrawings ) {
227
228         int minVelocity = 60;  // Minimum random velocity in cm/s.
229         int maxVelocity = 140;  // Maximum random velocity in cm/s.
230
```

```
231         int minDistanceFatigue =  5000;  // Min. random distance until
232                                          // fatigue.
233         int maxDistanceFatigue = 25000;  // Max. random distance until
234                                          // fatigue.
235
236         int minRegenerationTime = 240; // Min. random regeneration time.
237         int maxRegenerationTime = 720; // Max. random regeneration time.
238
239         Random random = new Random();
240         int randomInt;
241
242         RandomValueDrawings[] randomValues = new RandomValueDrawings[
243                 maxAmountOfPedestrians - minAmountOfPedestrians + 1];
244
245         // n --> Amount of pedestrians (minus min. pedestrian amount).
246         for (int n = 0;
247              n < (maxAmountOfPedestrians - minAmountOfPedestrians + 1);
248              n++) {
249             // Instantiate the random value drawings record for the
250             // current amount of pedestrians.
251             randomValues[n] = new RandomValueDrawings(n +
252                                                       minAmountOfPedestrians,
253                                                       numberOfRandomDrawings);
254
255             // i --> i-th pedestrian in the vector of pedestrians
256             for (int i = 0; i < n + minAmountOfPedestrians; i++) {
257
258                 // r --> Round; number of random drawings per pedestrian
259                 //               amount (and method).
260                 for (int r = 0; r < numberOfRandomDrawings; r++) {
261                     // Draw a walking direction.
262                     randomInt = random.nextInt(2);
263                     if (randomInt == 0) {
264                         randomValues[n].directions[i][r] = -1;
265                     }
266                     else {
267                         randomValues[n].directions[i][r] = 1;
268                     }
269                     // Draw a walking velocity.
270                     randomInt = random.nextInt(maxVelocity -
271                                                minVelocity + 1) +
272                                                minVelocity;
273                     randomValues[n].velocities[i][r] = (double)
274                                                           randomInt;
275
276                     // Draw a fatigue rate.
277                     randomInt = random.nextInt(maxDistanceFatigue -
278                                                minDistanceFatigue + 1) +
279                                                minDistanceFatigue;
280                     randomValues[n].fatigueRates[i][r] =
281                                     randomValues[n].velocities[i][r] /
282                                                ((double) randomInt);
283
284                     // Draw a pause Time.
285                     randomInt = random.nextInt(maxRegenerationTime -
286                                               minRegenerationTime + 1) +
287                                               minRegenerationTime;
288                     randomValues[n].pauseTimes[i][r] = 1.0 /
289                                                ((double) randomInt);
290                 }
```

```
291             }
292         }
293
294         return randomValues;
295
296     }
297
298     /**
299      * Generates several simulation cases and executes them.
300      * Number of random drawings per amount of pedestrians and method
301      * (exhaustive, heuristic, intuitive): cf. 3rd param. below.
302      * Maximum amount of pedestrians. cf. 1st param. below.
303      * Minimum random velocity: 60 cm/s; Maximum random velocity:
304      * 140 cm/s. (Both set in: createRandomValueDrawings)
305      * Min. resp. max. random distance until fatigue: 5,000 cm resp.
306      * 25,000 cm. (Both set in: createRandomValueDrawings)
307      * Min. resp. max. random regeneration rate: 1/(240 s) resp.
308      * 1/(720 s). (Both set in: createRandomValueDrawings)
309      *
310      * @param maxAmountOfPedestrians The upper boundary for the number
311      *                               pf pedestrians.
312      * @param minAmountOfPedestrians The lower boundary for the number
313      *                               of pedestrians. This amount should
314                                     not be too high such that the
315                                     exhaustive solution can be
316                                     determined for comparisons.
317      * @param numberOfRandomDrawings The number of runs per amount of
318                                      pedestrians (n) and method
319                                      (exhaustive, heuristic, intuitive).
320      * @param exhaustive Executes exhaustive method, if true.
321      * @param heuristic  Executes heuristic method, if true.
322      * @param intuitive  Executes intuitive method, if true.
323      */
324     public static void executeSimulationCases(
325                                             int maxAmountOfPedestrians,
326                                             int minAmountOfPedestrians,
327                                             int numberOfRandomDrawings,
328                                             boolean exhaustive,
329                                             boolean heuristic,
330                                             boolean intuitive) {
331
332         long sleepTimeBetweenMethods = 1000; // Some time between
333                                              // executing the methods to
334                                              // make sure that the
335                                              // corresponding log files
336                                              // can be properly closed
337                                              // and opened.
338
339         Processor processor = new Processor();
340
341         StateNode    initialNode = null;
342         Pedestrian[] pedestrians = null;
343
344         // Parameters to set in each simulation run. These are set
345         // upfront so that each method (exhcaustive, heuristic,
346         // intuitive) can be executed for each n = amount of pedestrians
347         // by using three log files with one logger instance.
348         // Create as much random value drawings records as there are
349         // amounts of pedestrians.
350         RandomValueDrawings[] randomValues = null;
```

```
351
352         // Make base settings.
353         baseSettings();
354
355         randomValues = createRandomValueDrawings(maxAmountOfPedestrians,
356                                                  minAmountOfPedestrians,
357                                                  numberOfRandomDrawings);
358
359
360         // ===== EXHAUSTIVE ============================================
361         if (exhaustive) { // Note, that conditional code is not indented
362
363         System.out.println("Start exhaustive cases.");
364
365         Logger.getCSVInstance().writeCSVHeader();
366
367         for (int n = 0;
368                 n < (maxAmountOfPedestrians - minAmountOfPedestrians +
369                      1);
370                 n++) {
371             for (int run = 0; run < numberOfRandomDrawings; run++) {
372
373                 // TODO Console output: Check
374                 System.out.println("=== EXHAUSITVE:  n = " +
375                                    (n + minAmountOfPedestrians) +
376                                    " ; Run " + (run+1));
377
378                 initialNode = new StateNode();
379                 initialNode.suos = createSUOs();
380                 pedestrians = createPedestrians(randomValues[n], run);
381                 initialNode.pedestrians = pedestrians;
382                 initialNode.numberOfPedestriansEnRoute =
383                                                     pedestrians.length;
384
385                 processor.buildTreeExhaustively(initialNode);
386
387                 // It is necessary to reset the static counters for the
388                 // state nodes, pedestrians and SUOs after each run.
389                 StateNode.number            = 0;
390                 SUO_AdaptiveSeating.number = 0;
391                 Pedestrian.number           = 0;
392             }
393         }
394
395         Logger.getCSVInstance().close(false, true);
396         Logger.getInstance().close(true, false);
397
398         System.out.println("Exhaustive cases finished.");
399
400         }  // Note, that the corresponding code block was not indented.
401         // ===== END EXCHAUSTIVE ======================================
402
403         System.out.println("Wait " + sleepTimeBetweenMethods +
404                            " ms...");
405         sleep(sleepTimeBetweenMethods);
406
407         // ===== HEURISTIC ============================================
408         if (heuristic) { // Note, that conditional code is not indented.
409
410         System.out.println("Start heuristic cases.");
```

```
411
412         Logger.getCSVInstance().writeCSVHeader();
413
414         for (int n = 0;
415                 n < (maxAmountOfPedestrians - minAmountOfPedestrians +
416                     1);
417                 n++) {
418             for (int run = 0; run < numberOfRandomDrawings; run++) {
419
420                 // TODO Console output: Check
421                 System.out.println("=== HEURISTIC:  n = " +
422                                    (n + minAmountOfPedestrians) +
423                                    " ; Run " + (run+1));
424
425                 initialNode = new StateNode();
426                 initialNode.suos = createSUOs();
427                 pedestrians = createPedestrians(randomValues[n], run);
428                 initialNode.pedestrians = pedestrians;
429                 initialNode.numberOfPedestriansEnRoute =
430                                                     pedestrians.length;
431
432                 processor.buildTreeHeuristically(initialNode);
433
434                 // It is necessary to reset the static counters for the
435                 // state nodes, pedestrians and SUOs after each run.
436                 StateNode.number            = 0;
437                 SUO_AdaptiveSeating.number = 0;
438                 Pedestrian.number           = 0;
439             }
440         }
441
442         Logger.getCSVInstance().close(false, true);
443         Logger.getInstance().close(true, false);
444
445         System.out.println("Heuristic cases finished.");
446
447         }   // Note, that the corresponding code block was not indented.
448         // ===== END HEURISTIC ==========================================
449
450         System.out.println("Wait " + sleepTimeBetweenMethods +
451                            " ms...");
452         sleep(sleepTimeBetweenMethods);
453
454         // ===== INTUITIVE ==============================================
455         if (intuitive) { // Note, that conditional code is not indented.
456
457         System.out.println("Start intuitive cases.");
458
459         Logger.getCSVInstance().writeCSVHeader();
460
461         for (int n = 0;
462                 n < (maxAmountOfPedestrians - minAmountOfPedestrians +
463                     1);
464                 n++) {
465             for (int run = 0; run < numberOfRandomDrawings; run++) {
466
467                 // TODO Console output: Check
468                 System.out.println("=== INTUITIVE:  n = " +
469                                    (n + minAmountOfPedestrians) +
470                                    " ; Run " + (run+1));
```

```
471
472                 initialNode = new StateNode();
473                 initialNode.suos = createSUOs();
474                 pedestrians = createPedestrians(randomValues[n], run);
475                 initialNode.pedestrians = pedestrians;
476                 initialNode.numberOfPedestriansEnRoute =
477                                                     pedestrians.length;
478
479                 processor.buildTreeIntuitively(initialNode);
480
481                 // It is necessary to reset the static counters for the
482                 // state nodes, pedestrians and SUOs after each run.
483                 StateNode.number           = 0;
484                 SUO_AdaptiveSeating.number = 0;
485                 Pedestrian.number          = 0;
486             }
487         }
488
489         Logger.getCSVInstance().close(false, true);
490         Logger.getInstance().close(true, false);
491
492         System.out.println("Intuitive cases finished.");
493         } // Note, that the corresponding code block was not indented.
494         System.out.println();
495         System.out.println("All finished.");
496         // ===== END INTUITIVE ==========================================
497
498     }
499
500
501     /**
502      * Executes "Thread.sleep(millis)" and catches the corresponding
503      * exception.
504      * @param timeMillis Time in Millisecond for pausing the thread.
505      */
506     protected static void sleep (long millis) {
507
508         try {
509             Thread.sleep(millis);
510         } catch (InterruptedException e) {
511             e.printStackTrace();
512         }
513
514     }
515
516 }
```

Processor.java

```
1  package main_package;
2
3  import java.util.LinkedList;
4
5  import util.Logger;
6  import util.NumberFormatter;
7
8  /**
9   * Contains the 'intelligence' for the smart urban object (SUO)
10  * "adaptive seating".
11  * @author Marvin Hubl
```

```
12  * @since 2019-10-14
13  */
14 public class Processor {
15
16     private Logger.LogLevel logLevel = Logger.LogLevel.NONE;
17
18     /**
19      * Builds a state space tree by the logic of breadt-first-search
20      * (FIFO).
21      * @param initialStateNode The initial state as node.
22      */
23     public void buildTreeExhaustively(StateNode initialStateNode) {
24
25         Logger lg                 = Logger.getInstance();
26         Logger csvLg              = Logger.getCSVInstance();
27         int    iterationCount     = 0;  // Just counts iteration (e.g.
28                                         // for print outputs fromt time
29                                         // to time)
30         int    printOutputInterval = 1000; // For providing a print
31                                            // output each specified
32                                            // number of expansions.
33
34         // The initial environmental state is copied. The copy then is
35         // manipulated. The initial state node does not branch (only a
36         // 1-to-1 connection to the first state node). The first state
37         // node will be a copy of the initial node at the point in time
38         // t where the first pedestrian passes an SUO.
39
40         // The processed node list
41         LinkedList<StateNode> processList = new LinkedList<StateNode>();
42         // Children nodes after expanding a node
43         StateNode[] childrenNodes;
44
45         StateNode bestLeafNode   = null;
46         double    safetyScore    = 0.0;
47         double    maxSafetyScore = -(Double.MAX_VALUE);
48
49         initialStateNode.scoreRecord =  new ScoreRecord(
50                                 initialStateNode.pedestrians.length);
51
52         StateNode node               = initialStateNode.copy();
53         node.parent                  = initialStateNode;
54         initialStateNode.children[0] = node;
55         initialStateNode.children[1] = null;
56         // Up to here: connecting the initial node with the first state
57         // space branch node
58
59         // TODO Logging: Check here
60 //      lg.write(initialStateNode, logLevel);
61
62         // From here: State space construction by logic of breath-first-
63         // search (Nodes are processed according to the first-in-first-
64         // out (FIFO) principle.)
65         maxSafetyScore = -(Double.MAX_VALUE);
66         processList.add(node);
67         while (!(processList.isEmpty())) {
68             // TODO Console output: Check here
69             // Print output from time to time (cf. above).
70             // Note: "%" is modulo.
71             if (iterationCount % printOutputInterval == 0) {
```

```
72                  System.out.println("Expansions: " + iterationCount);
73              }
74              iterationCount++;
75
76              node = processList.poll();
77              // Note that only "expand" updates the node (and it also
78              // connects parent and child nodes).
79              childrenNodes = expand(node);
80
81              // TODO Logging: Check here
82  //          lg.write(node, logLevel);
83
84              if (childrenNodes[0] != null) {
85                  processList.add(childrenNodes[0]);
86              }
87              if (childrenNodes[1] != null) {
88                  processList.add(childrenNodes[1]);
89              }
90              // if no children then leaf node
91              if(childrenNodes[0] == null & childrenNodes[1] == null) {
92                  // Score path at leaf nodes
93  //                --> Commented out: Scoring with another scoring function
94  //                node.scoreRecord.calcScore(1);
95  //                if (node.scoreRecord.score > maxScore) {
96  //                    maxScore = node.scoreRecord.score;
97  //                    bestLeafNode = node;
98  //                }
99                  safetyScore = SafetyScoreRecord.aggregate(
100                                                     node.pedestrians);
101                 if (safetyScore > maxSafetyScore) {
102                     maxSafetyScore = safetyScore;
103                     bestLeafNode = node;
104                 }
105             }
106         }
107
108         // TODO Logging: Check here
109 //      --> Commented out: Scoring with another scoring function
110 //      -->           And: Variable leafNodes not existing anymore.
111 //      lg.write("Number of leaf nodes", leafNodes.size(),
112 //                Logger.LogLevel.ALL);
113 //      lg.write("Best leaf node", bestLeafNode, Logger.LogLevel.ALL);
114 //      lg.write("Max. walking duration (as by the 'best leaf node')",
115 //           NumberFormatter.formatDouble(bestLeafNode.scoreRecord.
116 //                                                       getMaxX(3)),
117 //                                                 Logger.LogLevel.ALL);
118 //      lg.write("Max. fatigue at accomplishing the course (as by the
119 //               + "'best leaf node')",
120 //            NumberFormatter.formatDouble(bestLeafNode.scoreRecord.
121 //                                                       getMaxX(4)),
122 //                                                 Logger.LogLevel.ALL);
123 //      lg.write("Path", bestLeafNode.pathInfo, Logger.LogLevel.ALL);
124         csvLg.writeCSVLine(
125                 String.valueOf(maxSafetyScore),
126                 "exhaustiv",
127                 String.valueOf(bestLeafNode.pedestrians. length),
128                 String.valueOf(iterationCount),
129                 String.valueOf(bestLeafNode.scoreRecord.getMaxX(1)),
130                 String.valueOf(bestLeafNode.scoreRecord.getMaxX(2)),
131                 String.valueOf(bestLeafNode.scoreRecord.getMaxX(3)),
```

```
132                     String.valueOf(bestLeafNode.scoreRecord.getMaxX(4)));
133         lg.write(bestLeafNode.pathInfo, Logger.LogLevel.TO_FILE);
134
135     }
136
137
138
139     /**
140      * Builds a state space tree by the logic of best-first-search.
141      * @param initialStateNode The initial state as node.
142      */
143     public void buildTreeHeuristically(StateNode initialStateNode) {
144
145         Logger lg                  = Logger.getInstance();
146         Logger csvLg               = Logger.getCSVInstance();
147         int    iterationCount      = 0; // Just counts iteration (e.g.
148                                         // for print outputs from time
149                                         // to time)
150         int    printOutputInterval = 1000;  // For providing a print
151                                             // output each specified
152                                             // number of expansions.
153
154         long expansionCount = 0; // Counts the number of expansions to
155                                  // control run time, if nececassary.
156                                  // Note:
157                                  // This will always have the same value
158                                  // as "iterationCount" (but
159                                  // "expansionCount" has a dedicated
160                                  // role which is why it is a dedicated
161                                  // variable. - The overhead is not
162                                  // noticeable.)
163
164         // The initial environmental state is copied. The copy then is
165         // manipulated. The initial state node does not branch (only a
166         // 1-to-1 connection to the first state node). The first state
167         // node will be a copy of the inital node at the point in time
168         // where the first pedestrian passes an SUO.
169
170         // The list of open nodes
171         LinkedList<StateNode> open = new LinkedList<StateNode>();
172         // Children nodes after expanding a node
173         StateNode[] childrenNodes;
174         boolean finished = false;
175
176         HeuristicValueComparator heuristicValueComparator =
177                                         new HeuristicValueComparator();
178
179         // The heuristically revealed goal node.
180         StateNode goalNode = null;
181
182         initialStateNode.scoreRecord =
183                         new ScoreRecord(initialStateNode.pedestrians.
184                                                                 length);
185
186         StateNode node                = initialStateNode.copy();
187         node.parent                   = initialStateNode;
188         initialStateNode.children[0] = node;
189         initialStateNode.children[1] = null;
190         // Up to here: connecting the initial node with the first state
191         // space branch node
```

```
192
193         // TODO Logging: Check here
194         lg.write(initialStateNode, logLevel);
195
196         // From here:  State space construction by the logic of
197         //             best-first-search
198         open.add(node);
199
200         // There is always a goal state. That is why the open list will
201         // never be empty before a goal is found.
202         expansionCount = 0;
203         finished = false;
204         while (!finished) {
205             // TODO Console output: Check here
206             // Print output from time to time (cf. above).
207             // Note: "%" is modulo.
208             if (iterationCount % printOutputInterval == 0) {
209                 System.out.println("Expansions: " + iterationCount +
210                                    "   Amount of nodes in open list: " +
211                                    open.size());
212             }
213             iterationCount++;
214
215             node = open.poll();
216             childrenNodes = expand(node);
217             expansionCount++;
218             if (childrenNodes[1] == null) {
219                 // By convention, the right hand side child node
220                 // represents the alternative that a pedestrian does not
221                 // sit down on an SUO. If this subsequent state had not
222                 // been available, then the expanded node has been a
223                 // goal state (there is no need to check for the left
224                 // hand side child node).
225                 finished = true;
226                 goalNode = node;
227             }
228             else {
229                 // Since there is only one path to each node, due to the
230                 // tree structure of the problem, there is no need to
231                 // check wether a child node is already in open or
232                 // closed: This will never be the case (after expansion)
233                 for (StateNode childNode : childrenNodes) {
234                     if (childNode != null) {
235                         applyHeuristicEvaluation(childNode);
236                         open.add(childNode);
237                     }
238                 }
239                 open.sort(heuristicValueComparator);
240
241                 // Control for running time if needed.
242                 if (expansionCount >=  Director.C -
243                         (node.pedestrians.length * node.suos.length)) {
244                     // Keep only the first state node in the open list.
245                     node = open.getFirst();
246                     open.clear();
247                     open.add(node);
248                 }
249             }
250         }
251
```

```
252         // TODO Logging: Check here
253 //      --> Commented out: Scoring with another scoring function
254 //      lg.write("Goal node", goalNode, Logger.LogLevel.ALL);
255 //      lg.write("Max. walking duration (as by the 'best leaf node')",
256 //           NumberFormatter.formatDouble(bestLeafNode.scoreRecord.
257 //                                                        getMaxX(3)),
258 //                                                  Logger.LogLevel.ALL);
259 //      lg.write("Max. fatigue at accomplishing the course (as by the
260 //                                                    + "'best leaf node')",
261 //            NumberFormatter.formatDouble(bestLeafNode.scoreRecord.
262 //                                                        getMaxX(4)),
263 //            Logger.LogLevel.ALL);
264 //      lg.write("Path", bestLeafNode.pathInfo, Logger.LogLevel.ALL);
265         csvLg.writeCSVLine(
266                 String.valueOf(SafetyScoreRecord.aggregate(
267                                                    goalNode.pedestrians)),
268                 "heuristisch",
269                 String.valueOf(goalNode.pedestrians.length),
270                 String.valueOf(iterationCount),
271                 String.valueOf(goalNode.scoreRecord.getMaxX(1)),
272                 String.valueOf(goalNode.scoreRecord.getMaxX(2)),
273                 String.valueOf(goalNode.scoreRecord.getMaxX(3)),
274                 String.valueOf(goalNode.scoreRecord.getMaxX(4)));
275         lg.write(goalNode.pathInfo, Logger.LogLevel.TO_FILE);
276 
277     }
278 
279     /**
280      * Builds a state space tree by an intuitive logic from a
281      * pedestrians' viewpoint: If a pedestrian is fatigued, passes an
282      * SUO where there is still a place to sit, then she uses it.
283      * @param initialStateNode The initial state as node.
284      */
285     public void buildTreeIntuitively(StateNode initialStateNode) {
286 
287         Logger lg                 = Logger.getInstance();
288         Logger csvLg              = Logger.getCSVInstance();
289         int    iterationCount     = 0; // Just counts iteration (e.g.
290                                        // for priint outputs from time
291                                        // to time)
292         int    printOutputInterval = 1;   // For providing a print
293                                           // output each specified
294                                           // number of expansions.
295 
296         // The initial environmental state is copied. The copy then is
297         // manipulated. The initial state node does not branch (only a
298         // 1-to-1 connection to the first state node). The first state
299         // node will be a copy of the inital node at the point in time
300         // where the first pedestrian passes an SUO.
301 
302         // The processed node list
303         LinkedList<StateNode> processList = new LinkedList<StateNode>();
304         // Children nodes after expanding a node
305         StateNode[] childrenNodes;
306 
307         // The pedestrian that causes an expansion event.
308         Pedestrian passingPedestrian = null;
309 
310         StateNode leafNode = null;
311 
```

```
312         initialStateNode.scoreRecord =
313                 new ScoreRecord(initialStateNode.pedestrians.length);
314
315         StateNode node = initialStateNode.copy();
316         node.parent = initialStateNode;
317         initialStateNode.children[0] = node;
318         initialStateNode.children[1] = null;
319         // Up to here: connecting the initial node with the first state
320         // space branch node
321
322         // TODO Logging: Check here
323         lg.write(initialStateNode, logLevel);
324
325         // From here:  State space construction by the intuitive logic
326         // that a  pedestrian uses an SUO if she is fatigued and passes
327         // an SUO where is still capacity.
328         processList.add(node);
329         while (!(processList.isEmpty())) {
330             // TODO Console output: Check here
331             // Print output from time to time (cf. above).
332             // Note: "%" is modulo.
333             if (iterationCount % printOutputInterval == 0) {
334                 System.out.println("Expansions: " + iterationCount);
335             }
336             iterationCount++;
337
338             node = processList.poll();
339             // Note that only "expand" updates the node (and it also
340             // connects parent and child nodes).
341             childrenNodes = expand(node);
342
343             // TODO Logging: Check here
344             lg.write(node, logLevel);
345
346             // Find out which pedestrian caused the expansion event
347             // (i.e. which pedestrian passed an SUO). This information
348             // is only known within the expansion procedure but can be
349             // reconstructed by investigating which pedestrian is
350             // sitting in the left child while moving in the right
351             // child. (If the left child is null then this information
352             // is not needed anyway.)
353             passingPedestrian = null;
354             if (childrenNodes[0] != null & childrenNodes[1] != null) {
355                 for (int i = 0; i < node.pedestrians.length; i++) {
356                     if (childrenNodes[0].pedestrians[i].state ==
357                             MovementState.USING_SUO &
358                         childrenNodes[1].pedestrians[i].state ==
359                             MovementState.WALKING) {
360                         passingPedestrian = node.pedestrians[i];
361                     }
362                 }
363             }
364
365             if (passingPedestrian != null &&
366                 passingPedestrian.fatigue >= Director.CRITICAL_FATIGUE)
367             {
368                 processList.add(childrenNodes[0]);
369             }
370             else if (childrenNodes[1] != null){
371                 processList.add(childrenNodes[1]);
```

```
372             }
373
374             // if no children then leaf node
375             if(childrenNodes[0] == null & childrenNodes[1] == null) {
376                 leafNode = node;
377             }
378         }
379
380         // TODO Logging: Check here
381 //      --> Commented out: Scoring with another scoring function
382 //      -->           And: Variable leafNodes not existing anymore.
383 //      lg.write("Number of leaf nodes", leafNodes.size(),
384 //                Logger.LogLevel.ALL);
385 //      lg.write("Best leaf node", bestLeafNode, Logger.LogLevel.ALL);
386 //      lg.write("Max. walking duration (as by the 'best leaf node')",
387 //           NumberFormatter.formatDouble(bestLeafNode.scoreRecord.
388 //                                                        getMaxX(3)),
389 //                                                   Logger.LogLevel.ALL);
390 //      lg.write("Max. fatigue at accomplishing the course (as by the
391 //               + "'best leaf node')",
392 //             NumberFormatter.formatDouble(bestLeafNode.scoreRecord.
393 //                                                        getMaxX(4)),
394 //                                                   Logger.LogLevel.ALL);
395 //      lg.write("Path", bestLeafNode.pathInfo, Logger.LogLevel.ALL);
396         csvLg.writeCSVLine(
397                         String.valueOf(SafetyScoreRecord.aggregate(
398                                                        leafNode.pedestrians)),
399                         "intuitiv",
400                         String.valueOf(leafNode.pedestrians.length),
401                         String.valueOf(iterationCount),
402                         String.valueOf(leafNode.scoreRecord.getMaxX(1)),
403                         String.valueOf(leafNode.scoreRecord.getMaxX(2)),
404                         String.valueOf(leafNode.scoreRecord.getMaxX(3)),
405                         String.valueOf(leafNode.scoreRecord.getMaxX(4)));
406         lg.write(leafNode.pathInfo, Logger.LogLevel.TO_FILE);
407
408     }
409
410     /**
411      * Applies the heuristic evalation (e) to the given state node
412      * w.r.t. the goal state (and writes the value in the attribute
413      * "node.heuristicValue"). The heuristic value estimates the maximum
414      * distances among the distances that the pedestrians walk with
415      * critical degree of fatigue.
416      * @param node The state node to be evaluated by means of the
417      *             heuristic.
418      */
419     public void applyHeuristicEvaluation(StateNode node) {
420         node.heuristicValue = costToCurrentState(node) +
421                               costToGoalState(node);
422     }
423
424     /**
425      * Evaluates the current state (g). The result are the cost from the
426      * initial state to the current state. Therefore, the evaluation
427      * compares the distances that each pedestrian has walked with
428      * crititcal degree of fatigue so far and returns the maximum
429      * distance among them.
430      * @param currentState The current state node.
431      * @return The evaluation of the current state: Maximum distance
```

```
432         *          among the distances that each pedestrian has walked with
433         *          critical degree of fatigue.
434         */
435        public double costToCurrentState(StateNode currentState) {
436
437            double maxDistance = -Double.MAX_VALUE;
438
439            for (Pedestrian p : currentState.pedestrians) {
440                if (p.fatiguedDistance > maxDistance) {
441                    maxDistance = p.fatiguedDistance;
442                }
443            }
444
445            return maxDistance;
446        }
447
448
449        /**
450         * Represents the actual heuristic (h). The result is an estimate of
451         * the costs from the current state to the aspired goal state.
452         * Therefore the function estimates the maximum distance among the
453         * distances that all pedestrians walk with critical fatigue from
454         * the current state to the goal state.
455         * @param node The current state node
456         * @return The actual heuristic value: Estimate of the maximum
457         *          distance among the distances that all pedestrians walk
458         *          with critical degree of fatigue from the current state to
459         *          the aspired goal state.
460         */
461        public double costToGoalState(StateNode currentState) {
462
463            double maxCriticalDistance = -Double.MAX_VALUE;
464            double[] criticalDistances = new double[currentState.
465                                                    pedestrians.length];
466            double idealDistance       = 0.0;
467            double lastRestPosition    = 0.0;
468            int    lastRestSUOIndex    = 0;
469            double idealNextPosition   = 0.0;
470            double minDeviation        = Double.MAX_VALUE;
471            double deviation           = 0.0;
472            int    nextRestSUOIndex    = 0;
473
474            // Confer Algorithm "Berechnung der Heuristik h(\tau)" in the
475            // dissertation text:
476
477            for (Pedestrian p : currentState.pedestrians) {
478
479                idealDistance     = p.normalVelocity / p.fatigueRate;
480
481                // Since in this implementation the boundary positions are
482                // considered separately than the suo positions, there is
483                // the need to check whether a pedestrian is at the starting
484                // point to determine her last (rest) position.
485                // First (if): "normal" case; second (else): initial case.
486                if (p.lastUsedSUOIndex >= 0) {
487                    lastRestSUOIndex  = p.lastUsedSUOIndex;
488                    lastRestPosition  = currentState.suos[lastRestSUOIndex].
489                                        position;
490                }
491                else {
```

```
492                 if (p.direction == 1) {
493                     lastRestPosition = Director.LEFT_POS;
494                     lastRestSUOIndex = -1; // Must be -1; cf. the head
495                                            //  of the while loop below.
496
497                 }
498                 else { // --> p.direction == - 1
499                     lastRestPosition = Director.RIGHT_POS;
500                     lastRestSUOIndex = currentState.suos.length;
501                         // Must be suos.length, cf. the head of the
502                         // while lopp below.
503                 }
504             }
505
506             idealNextPosition = lastRestPosition +
507                                 (p.direction * idealDistance);
508
509             criticalDistances[p.index] = 0.0;
510
511             while (Director.LEFT_POS <= idealNextPosition &
512                     idealNextPosition <= Director.RIGHT_POS) {
513
514                 minDeviation = Double.MAX_VALUE;
515                 for (int j = lastRestSUOIndex + p.direction;
516                      0 <= j & j < currentState.suos.length;
517                     j += p.direction) {
518
519                     deviation = Math.abs(currentState.suos[j].position -
520                                          idealNextPosition);
521
522                     if (deviation < minDeviation) {
523                         minDeviation = deviation;
524                         nextRestSUOIndex = j;
525                     }
526                 }
527
528                 // Consider the boundary positions separetely at the end
529                 if (p.direction == 1) {
530                     deviation = Math.abs(Director.RIGHT_POS -
531                                          idealNextPosition);
532                     if (deviation < minDeviation) {
533                         minDeviation = deviation;
534                         nextRestSUOIndex = currentState.suos.length;
535                         lastRestPosition = Director.RIGHT_POS;
536                     }
537                 }
538                 else { // --> p.direction == -1
539                     deviation = Math.abs(Director.LEFT_POS -
540                                          idealNextPosition);
541                     if (deviation < minDeviation) {
542                         minDeviation = deviation;
543                         nextRestSUOIndex = -1;
544                         lastRestPosition = Director.LEFT_POS;
545                     }
546                 }
547                 // End of separate consideration.
548                 // Note that the following if-clause is necessary due to
549                 // the seperate consideration of the boundary positions.
550                 if (0 <= nextRestSUOIndex &
551                     nextRestSUOIndex < currentState.suos.length) {
```

```
552
553                     lastRestPosition = currentState.
554                                         suos[nextRestSUOIndex].
555                                           position;
556                 }
557                 lastRestSUOIndex = nextRestSUOIndex;
558                 idealNextPosition = lastRestPosition +
559                                     (p.direction * idealDistance);
560
561                 if (p.direction *
562                         (idealNextPosition - lastRestPosition) > 0.0) {
563                     criticalDistances[p.index] += minDeviation;
564                 }
565             }
566             // The boundary positions must be considered separateley.
567             if (p.direction == 1) {
568                 if (Director.RIGHT_POS - lastRestPosition >
569                       idealDistance) {
570                     criticalDistances[p.index] += Director.RIGHT_POS -
571                                                   lastRestPosition;
572                 }
573             }
574             else { // --> p.direction == -1
575                 if (lastRestPosition - Director.LEFT_POS >
576                       idealDistance) {
577                     criticalDistances[p.index] += lastRestPosition -
578                                                   Director.LEFT_POS;
579                 }
580             }
581         }
582
583         maxCriticalDistance = -Double.MAX_VALUE;
584         for (int i = 0; i < currentState.pedestrians.length; i++) {
585             if (criticalDistances[i] > maxCriticalDistance) {
586                 maxCriticalDistance = criticalDistances[i];
587             }
588         }
589
590         return maxCriticalDistance;
591     }
592
593
594     /**
595      * Expands the input node by a dedicated logic. Thereby the state
596      * information of the input node is updated. The returned child
597      * nodes will be copies of the updated input node and will differ
598      * from the updated input node only in the information whether the
599      * passed pedestrian, causing the expansion event, sits down (left
600      * child) or not (right child).
601      * @param node The node to expand for creating a state space tree.
602      * @return Returns the two child nodes ([0] = left child,
603      *         [1] = right child; they can be "null" but the returned
604      *         array itself will not be "null").
605      */
606     public StateNode[] expand(StateNode node) {
607
608         @SuppressWarnings("unused")
609         Logger lg = Logger.getInstance();
610
611         BranchEvent de; // 3-tuple: (double, SUO_AdaptiveSeating,
```

```
612                     //          Pedestrian)
613         double      remainingWalkTime;
614         double      latestArrivalTime; // Needed - this will be the
615                                        // closure node.
616
617         StateNode leftChild;
618         StateNode rightChild;
619
620         @SuppressWarnings("unchecked")// Suppress Since LinkedList in
621                                       // the initialisation of the Array
622                                       // of LinkedLists is not/cannot be
623                                       // parametritized with Pedestrian.
624         // It will be necessary to have a list for each SUO that is
625         // sorted by the remaining use time of the pedestrians that use
626         // it to determine the amount of other pedestrians who use an
627         // SUO when a pedestrian  stands up.
628         // remainingUseTimeSortedLists[j] is the corresponding list for
629         // the j-th SUO. (For sorting the remainingUseTimeComparator is
630         // used.)
631         LinkedList<Pedestrian>[] remainingUseTimeSortedLists =
632                                       new LinkedList[node.suos.length];
633         RemainingUseTimeComparator remainingUseTimeComparator =
634                                       new RemainingUseTimeComparator();
635         int place = 0;
636         int occupancy = 0; // The amount of other pedestrians still
637                            // using an  SUO when a pedestrian stands up.
638
639         double fatigueCopy; // Needed for calculating x4, i.e. fatigue
640                             // at accomplishing the course
641         double untrimmedFatigueCopy; // Needed for calculating resp.
642                                      // untrimmed fatigue at
643                                      // accomplishing the course
644         double fatigueWalkingDistanceCopy; // Needed for calculating x1,
645                                            // i.e. the distance walked
646                                            // with a critical degree of
647                                            // fatigue
648
649         double offset = 0.001; // An offset to place pedestrians
650                                // slightly offset in walking direction
651                                // from their last position at the moment
652                                // of passing an SUO. This is very
653                                // crucial so that the pedestrian will
654                                // not be reconsidered for the SUO more
655                                // than one time. This would happen
656                                // because within the "timeToNextSUO"
657                                // method the ">=" operator needs to be
658                                // applied.
659
660         // Create the lists for each SUO sorted by the remaining use
661         // times: First, initialize the list. Second, add pedestrians
662         // who use an SUO in the corresponding list. Third, sort the
663         // lists.
664         for (int j = 0; j < remainingUseTimeSortedLists.length; j++) {
665             remainingUseTimeSortedLists[j] =
666                                           new LinkedList<Pedestrian>();
667         }
668         for (Pedestrian p : node.pedestrians) {
669             if (p.state == MovementState.USING_SUO ) {
670                 remainingUseTimeSortedLists[p.currentlyUsedSUO.index].
671                                                                 add(p);
```

```
672             }
673         }
674         for (LinkedList<Pedestrian> list : remainingUseTimeSortedLists)
675         {
676             list.sort(remainingUseTimeComparator);
677         }
678 
679         de = timeToNextSUO(node.stateTime, node.pedestrians, node.suos);
680 
681         // The case that nextPassedSUO == null is handled below. For the
682         // 'normal' case where nextPassedSUO != null, there is still a
683         // pedestrian passing an SUO
684         if (de.nextPassedSUO != null) {
685         // --> At least one pedestrian will pass at least one SUO
686         // --> Update environmental state
687             for (Pedestrian p : node.pedestrians) {
688 
689                 // Both may be needed below (if distance  will be
690                 // accomplished)
691                 fatigueCopy                = p.fatigue;
692                 untrimmedFatigueCopy       = p.untrimmedFatigue;
693                 fatigueWalkingDistanceCopy = p.fatiguedDistance;
694 
695                 // Check for the pedestrian whether she sits, walks or
696                 // first sits and then walks
697                 if (p.state == MovementState.USING_SUO) {
698                 // --> Pedestrian sits
699                     // Note that p.remainingUseTime had been calculated
700                     // within timeToNextSUO before
701                     if (p.remainingUseTime >= de.timeHorizon) {
702                     // --> Pedestrian sits the whole time
703                         estimateFatigueWhileUsingSUO(p, de.timeHorizon);
704                     }
705                     else {
706                     // --> Pedestrian sits but stands up within
707                     //     timeHorizon
708 
709                         // x2 in score record: Count how often a
710                         // pedestrian (starts or) ends using an SUO
711                         // (i.e. (sits down/) stands up) while another
712                         // one uses the same SUO; per pedestrian
713 
714                         // Check how many pedestrians currently using
715                         // the same SUO as p will stand up before p
716                         // stands up: Therefore retrieve the index of p
717                         // in the list sorted by currently remaining use
718                         // times of the SUO. The number of pedestrians
719                         // with a greater index in this sorted list is
720                         // the number of pedestrian who are still using
721                         // the SUO if p stands up from it.
722                         place     = remainingUseTimeSortedLists[
723                                         p.currentlyUsedSUO.index].
724                                             indexOf(p);
725                         occupancy = remainingUseTimeSortedLists[
726                                         p.currentlyUsedSUO.index].
727                                             size() - 1 - place;
728 
729                         if (occupancy > 0) {
730                             // TODO x2: ok
731                             node.scoreRecord.x2[p.index] += 1;
```

```
732                     }
733                     // TODO Ok: Safety-B-Score for standing up.
734                     p.safetyScoreRecord.
735                             addSafetyBScoreForStandingUp(occupancy);
736
737                     // Note that the occupancy of of the SUO is
738                     // decreased only here.
739                     p.currentlyUsedSUO.occupancy -= 1;
740                     p.currentlyUsedSUO = null;
741                     // State change here: from USING_SUO to WALKING
742                     p.lastStateAlterationTime = node.stateTime +
743                                                 p.remainingUseTime;
744                     p.state = MovementState.WALKING;
745                     // By assumption, pedestrians stands up only if
746                     // she is not fatigued anymore.
747                     p.fatigue          = 0.0;
748                     p.untrimmedFatigue = 0.0;
749                     remainingWalkTime  = de.timeHorizon -
750                                          p.remainingUseTime;
751                     projectPosition(p, remainingWalkTime);
752                     // Both may be needed below (if distance  will
753                     // be accomplished)
754                     fatigueCopy                 = p.fatigue;
755                     untrimmedFatigueCopy        = p.untrimmedFatigue;
756                     fatigueWalkingDistanceCopy = p.fatiguedDistance;
757                     estimateFatigueWhileWalking(p,
758                                                 remainingWalkTime,
759                                                 true);
760
761                 }
762             }
763             else if (p.state == MovementState.WALKING) {
764             // --> Pedestrian walks the whole time
765                 projectPosition(p, de.timeHorizon);
766                 // Both may be needed below (if distance  will be
767                 // accomplished)
768                 fatigueCopy = p.fatigue;
769                 untrimmedFatigueCopy = p.untrimmedFatigue;
770                 fatigueWalkingDistanceCopy = p.fatiguedDistance;
771                 estimateFatigueWhileWalking(p,
772                                             de.timeHorizon,
773                                             true);
774
775             }
776             // Check, whether pedestrian has accomplished the course
777             // for both walking directions seperately
778             if (p.state != MovementState.DISTANCE_ACCOMPLISHED) {
779                 if ((p.direction == 1) &
780                         (Director.RIGHT_POS <= p.position)) {
781                     node.numberOfPedestriansEnRoute -= 1;
782                     p.state = MovementState.DISTANCE_ACCOMPLISHED;
783                     // Record time of arrival
784                     p.lastStateAlterationTime = node.stateTime +
785                           projectWalkingTime(Director.RIGHT_POS, p);
786                     p.position = Director.RIGHT_POS + offset;
787                                                 // slighlty exceed
788
789                     // x3 in score record: Time needed for
790                     // accomplishing the distance; per pedestrian
791                     // TODO x3: ok
```

```
792                         node.scoreRecord.x3[p.index] =
793                                                 p.lastStateAlterationTime;
794
795                         // x4 in score record: Degree of fatigue at
796                         // accomplishing the distance; per pedestrian
797                         p.fatigue = fatigueCopy;
798                         p.untrimmedFatigue = untrimmedFatigueCopy;
799                         p.fatiguedDistance = fatigueWalkingDistanceCopy;
800                         // (Second argument is the walking time to the
801                         //  end pos.)
802                         estimateFatigueWhileWalking(p,
803                              p.lastStateAlterationTime - node.stateTime,
804                              true);
805                         // TODO x4: ok
806                         node.scoreRecord.x4[p.index] = p.fatigue;
807
808                         // TODO Ok: Final Safety-A-Score for
809                         //          accomplishing the course
810                         p.safetyScoreRecord.addSafetyAScore(p, true);
811
812                     }
813                     if ((p.direction == -1) &
814                          (Director.LEFT_POS >= p.position)) {
815                         node.numberOfPedestriansEnRoute -= 1;
816                         p.state = MovementState.DISTANCE_ACCOMPLISHED;
817                         // Record time of arrival
818                         p.lastStateAlterationTime = node.stateTime +
819                                 projectWalkingTime(Director.LEFT_POS, p);
820                         p.position = Director.LEFT_POS - offset;
821                                                 // slightly exceed
822
823                         // x3 in score record: Time needed for
824                         // accomplishing the distance; per pedestrian
825                         // TODO x3: ok
826                         node.scoreRecord.x3[p.index] =
827                                                 p.lastStateAlterationTime;
828
829                         // x4 in score record: Degree of fatigue at
830                         // accomplishing the distance; per pedestrian
831                         p.fatigue = fatigueCopy;
832                         p.untrimmedFatigue = p.untrimmedFatigue;
833                         p.fatiguedDistance = fatigueWalkingDistanceCopy;
834                         // (Second argument is the walking time to the
835                         //   end pos.)
836                         estimateFatigueWhileWalking(p,
837                              p.lastStateAlterationTime - node.stateTime,
838                              true);
839                         // TODO x4: ok
840                         node.scoreRecord.x4[p.index] = p.fatigue;
841
842                         // TODO Ok: Final Safety-A-Score for
843                         //          accomplishing the course
844                         p.safetyScoreRecord.addSafetyAScore(p, true);
845                     }
846                 }
847                 // x1 in score record: Distance walked with critical
848                 // degree of fatigue; per pedestrian
849                 // TODO x1: ok
850                 node.scoreRecord.x1[p.index] = p.fatiguedDistance;
851             }
```

```
852             node.stateTime += de.timeHorizon; // Update state time
853                                               // (must be here)
854         }
855         else {
856         // --> de.nextPassedSUO == null
857         // ==> No pedestrian will pass another SUO anymore
858         // --> Create closure node.
859             latestArrivalTime = -(Double.MAX_VALUE);
860             for (Pedestrian p : node.pedestrians) {
861                 // Check for the pedestrian whether she is sitting or
862                 // walking
863                 if (p.state == MovementState.USING_SUO) {
864                 // --> Pedestrian is sitting (and will stand up after
865                 //     remainingUseTime which had been calculated in
866                 //     "timeToNextSUO")
867
868                     // Check how many pedestrians currently using the
869                     // same SUO as p will stand up before p stands up:
870                     // Therefore retrieve the index of p in the list
871                     // sorted by currently remaining use times of the
872                     // SUO. The number of pedestrians with a greater
873                     // index in this sorted list is the number of
874                     // pedestrian who are still using the SUO if p
875                     // stands up from it.
876                     place = remainingUseTimeSortedLists[
877                                     p.currentlyUsedSUO.index].indexOf(p);
878                     occupancy = remainingUseTimeSortedLists[
879                             p.currentlyUsedSUO.index].size() - 1 - place;
880
881                     // x2 in score record: Count how often a pedestrian
882                     // (starts or) ends using an SUO (i.e. (sits down/)
883                     // stands up) while another one uses the same SUO;
884                     // per pedestrian
885                     if (occupancy > 0) {
886                         // TODO x2: ok
887                         node.scoreRecord.x2[p.index] += 1;
888                     }
889                     // TODO Ok: Safety-B-Score for standing up.
890                     p.safetyScoreRecord.addSafetyBScoreForStandingUp(
891                                                             occupancy);
892
893                     // Note that the occupancy of of the SUO is
894                     // decreased only here.
895                     p.currentlyUsedSUO.occupancy -= 1;
896                     p.currentlyUsedSUO = null;
897                     // By assumption, pedestrians stands up only if she
898                     // is not fatigued anymore.
899                     p.fatigue          = 0.0;
900                     p.untrimmedFatigue = 0.0;
901                     // Core difference to the branch with
902                     // nextPassedSUO != null:
903                     // use walk time to the end position instead of
904                     // timeHorizon
905                     if (p.direction == 1) {
906                         remainingWalkTime = projectWalkingTime(
907                                                 Director.RIGHT_POS, p);
908                         p.position = Director.RIGHT_POS + offset;
909                                                     // slightly exceed
910                     }
911                     else {
```

```
912                         remainingWalkTime = projectWalkingTime(
913                                                 Director.LEFT_POS, p);
914                         p.position = Director.LEFT_POS - offset;
915                                                 // slightly exceed
916                     }
917                     estimateFatigueWhileWalking(p, remainingWalkTime,
918                                                 true);
919 
920                     // Pedestrian has accomplished the course.
921                     p.state = MovementState.DISTANCE_ACCOMPLISHED;
922                     node.numberOfPedestriansEnRoute -= 1;
923                     // Record time of arrival
924                     p.lastStateAlterationTime = node.stateTime +
925                                                 p.remainingUseTime +
926                                                 remainingWalkTime;
927                     // latest arrival time needed for the node's state
928                     // time
929                     if(p.lastStateAlterationTime > latestArrivalTime) {
930                         latestArrivalTime = p.lastStateAlterationTime;
931                     }
932 
933                     // x3 in score record: Time needed for accomplishing
934                     // the distance; per pedestrian
935                     // TODO x3: ok
936                     node.scoreRecord.x3[p.index] =
937                                                 p.lastStateAlterationTime;
938 
939                     // x4 in score record: Degree of fatigue at
940                     // accomplishing the distance; per pedestrian
941                     // TODO x4: ok
942                     node.scoreRecord.x4[p.index] = p.fatigue;
943 
944                     // TODO Ok: Final Safety-A-Score for accomplishing
945                     //          the course
946                     p.safetyScoreRecord.addSafetyAScore(p, true);
947 
948                 }
949                 else if (p.state == MovementState.WALKING) {
950                 // --> Pedestrian walks
951                     // Core difference to the branch with
952                     // nextPassedSUO != null: use walk time to the end
953                     // position instead of timeHorizon
954                     if (p.direction == 1) {
955                         remainingWalkTime = projectWalkingTime(
956                                                 Director.RIGHT_POS, p);
957                         p.position = Director.RIGHT_POS + offset;
958                                                 // slightly exceed
959                     }
960                     else {
961                         remainingWalkTime = projectWalkingTime(
962                                                 Director.LEFT_POS, p);
963                         p.position = Director.LEFT_POS - offset;
964                                                 // slightly exceed
965                     }
966                     estimateFatigueWhileWalking(p, remainingWalkTime,
967                                                 true);
968 
969 
970                     // Pedestrian has accomplished the course.
971                     p.state = MovementState.DISTANCE_ACCOMPLISHED;
```

```
972                     node.numberOfPedestriansEnRoute -= 1;
973                     // Record time of arrival
974                     p.lastStateAlterationTime = node.stateTime +
975                                                 remainingWalkTime;
976                     // latest arrival time needed for the node's state
977                     // time
978                     if(p.lastStateAlterationTime > latestArrivalTime) {
979                         latestArrivalTime = p.lastStateAlterationTime;
980                     }
981
982                     // x3 in score record: Time needed for accomplishing
983                     // the distance; per pedestrian
984                     // TODO x3: ok
985                     node.scoreRecord.x3[p.index] =
986                                                 p.lastStateAlterationTime;
987
988                     // x4 in score record: Degree of fatigue at
989                     // accomplishing the distance; per pedestrian
990                     // TODO x4: ok
991                     node.scoreRecord.x4[p.index] = p.fatigue;
992
993                     // TODO Ok: Final Safety-A-Score for accomplishing
994                     //          the course
995                     p.safetyScoreRecord.addSafetyAScore(p, true);
996                 }
997                 // x1 in score record: Distance walked with critical
998                 // degree of fatigue; per pedestrian
999                 // TODO x1: ok
1000                node.scoreRecord.x1[p.index] = p.fatiguedDistance;
1001            }
1002            node.stateTime = latestArrivalTime;
1003        }
1004
1005        // Now, make the branches: Left branch is by convention always
1006        // the state altertative to use the SUO (if possible); right
1007        // branch to walk on. There are two abort criteria: (1.) SUO is
1008        // fully occupied (abort left branch); (2.) All pedestrians have
1009        // accomplished the course (abort all branches).
1010        if (node.numberOfPedestriansEnRoute > 0) {
1011        // --> not all pedestrians have arrived, yet
1012        //    (nextPassedSUO is not null)
1013            if (de.nextPassedSUO.occupancy == de.nextPassedSUO.capacity)
1014            {
1015            // --> capacity must not and cannot be exceeded
1016                leftChild = null;
1017            }
1018            else {
1019            // --> the pedestrian at an SUO can use the SUO
1020
1021                leftChild = node.copy(); // does not copy parent and
1022                                         // children but set them to
1023                                         // "null"
1024                leftChild.parent = node;
1025
1026                leftChild.pathInfo +=
1027                            "\r\n" + de.nextPassingPedestrian.id +
1028                            " --> " + de.nextPassedSUO.id + " (at " +
1029                            NumberFormatter.formatDouble(
1030                                    de.timeHorizon + node.stateTime) +
1031                            " s)";
```

```
1032
1033                // The copy's currentlyUsedSUO must refer to an SUO in
1034                // the suos-array of the copy:
1035                for (int i = 0; i < leftChild.pedestrians.length; i++) {
1036                    if (node.pedestrians[i].currentlyUsedSUO != null) {
1037                        leftChild.pedestrians[i].currentlyUsedSUO =
1038                                leftChild.suos[node.pedestrians[i].
1039                                        currentlyUsedSUO.index];
1040                    }
1041                }
1042
1043
1044
1045                // x2 in score record: Count how often a pedestrian
1046                // starts (or ends) using an SUO (i.e. sits down
1047                // (/stands up)) while another one uses the same SUO;
1048                // per pedestrian
1049                if (de.nextPassedSUO.occupancy > 0) {
1050                    // TODO x2: ok
1051                    node.scoreRecord.x2[de.nextPassingPedestrian.index]
1052                                                                    += 1;
1053                }
1054
1055                // In the left child: occupy SUO with the pedestrian
1056                // being there and record the SUO in the pedestrian.
1057
1058                leftChild.suos[de.nextPassedSUO.index].occupancy += 1;
1059                leftChild.pedestrians[de.nextPassingPedestrian.index].
1060                      currentlyUsedSUO = leftChild.
1061                                         suos[de.nextPassedSUO.index];
1062                leftChild.pedestrians[de.nextPassingPedestrian.index].
1063                             state = MovementState.USING_SUO;
1064                leftChild.pedestrians[de.nextPassingPedestrian.index].
1065                             lastUsedSUOIndex = de.nextPassedSUO.index;
1066                leftChild.pedestrians[de.nextPassingPedestrian.index].
1067                   lastStateAlterationTime = node.stateTime;
1068                // Set the pedestrian exactly to the SUO position with a
1069                // slight offset to avoid that she will be considered
1070                // after standing up again for using this SUO (because
1071                // distance 0.0 would be a minimum distance to a next
1072                // SUO when applying ">=" operator)
1073                if (de.nextPassingPedestrian.direction == 1) {
1074                    leftChild.
1075                      pedestrians[de.nextPassingPedestrian.index].
1076                         position = de.nextPassedSUO.position + offset;
1077                }
1078                else {
1079                    leftChild.
1080                      pedestrians[de.nextPassingPedestrian.index].
1081                        position = de.nextPassedSUO.position - offset;
1082                }
1083
1084                // TODO Ok: Safety-B-Score for sitting down.
1085                // Note that the occupancy value has already been
1086                // increased above.
1087                leftChild.pedestrians[de.nextPassingPedestrian.index].
1088                        safetyScoreRecord.addSafetyBScoreForSittingDown(
1089                          leftChild.
1090                            pedestrians[de.nextPassingPedestrian.index].
1091                              currentlyUsedSUO.occupancy - 1);
```

```
1092
1093                // TODO Ok: Safety-A-Score (for using an SUO)
1094                leftChild.pedestrians[de.nextPassingPedestrian.index].
1095                        safetyScoreRecord.addSafetyAScore(
1096                  leftChild.pedestrians[de.nextPassingPedestrian.index],
1097                                                                  false);
1098
1099            }
1100            rightChild = node.copy();
1101            rightChild.parent = node;
1102            // The copy's "currentlyUsedSUO" must refer to an SUO in the
1103            // suos-array of the copy:
1104            for (int i = 0; i < rightChild.pedestrians.length; i++) {
1105                if (node.pedestrians[i].currentlyUsedSUO != null) {
1106                    rightChild.pedestrians[i].currentlyUsedSUO =
1107                                    rightChild.suos[node.pedestrians[i].
1108                                                    currentlyUsedSUO.index];
1109                }
1110            }
1111            // Set the pedestrian exactly to the current position with a
1112            // slight offset. This is important if she's at an SUO, that
1113            // she will not be reconsidered for this SUO (because
1114            // distance 0.0 would be a minimum distance to a next SUO
1115            // when applying ">=" operator)
1116            if (de.nextPassingPedestrian.direction == 1) {
1117                rightChild.pedestrians[de.nextPassingPedestrian.index].
1118                        position = de.nextPassedSUO.position + offset;
1119            }
1120            else {
1121                rightChild.pedestrians[de.nextPassingPedestrian.index].
1122                        position = de.nextPassedSUO.position - offset;
1123            }
1124
1125        }
1126        else {
1127        // --> all pedestrians have arrived (close state space tree)
1128            leftChild = null;
1129            rightChild = null;
1130        }
1131
1132        node.children[0] = leftChild;
1133        node.children[1] = rightChild;
1134        return new StateNode[] {leftChild, rightChild};
1135    }
1136
1137    /**
1138     * Calculates who passed which SUO when. Calculated is the first
1139     * point in time, where a pedestrian passes an SUO after the point
1140     * in time given by the "currentStateTime"-parameter.
1141     * @param currentStateTime The point in time as starting time for
1142     *                         the calculation.
1143     * @param pedestrians The list of pedestrians as encapsulated data
1144     *                    types.
1145     * @param suos The list of SUOs as encapsulated data types
1146     * @return BranchEvent.timeHorizon: point in time where the next SUO
1147     *         is passed or Double.MAX_VALUE if no pedestrian will pass
1148     *         an SUO;
1149     *         BranchEvent.nextPassedSUO: the next SUO what is passed by
1150     *                                    any pedestrian;
1151     *         BranchEvent.pedestrian: the next pedestrian who passes
```

```
1152      *                                  any SUO.
1153      */
1154     public BranchEvent timeToNextSUO(double currentStateTime,
1155                                      Pedestrian[] pedestrians,
1156                                      SUO_AdaptiveSeating[] suos) {
1157
1158         BranchEvent          decEv = new BranchEvent();
1159         Pedestrian           thePedestrian;
1160         SUO_AdaptiveSeating theNextSUO;
1161         double               minDuration;
1162         double               fatigueResetCopy; // Necessary, for
1163                                  // resetting p.fatigue which is
1164                                  // potentially overwritten below
1165         double untrimmedFatigueResetCopy; // Necessary, for resetting
1166                                           // p.untrimmedFatigue which is
1167                                           // potentially overwritten
1168                                           // below
1169         double minDistance;
1170         SUO_AdaptiveSeating nextSUO;
1171         double duration;
1172
1173         thePedestrian = null;
1174         theNextSUO = null;
1175         minDuration = Double.MAX_VALUE;
1176         for (Pedestrian p : pedestrians) {
1177             p.remainingUseTime        = 0.0;
1178             fatigueResetCopy          = p.fatigue;
1179             untrimmedFatigueResetCopy = p.untrimmedFatigue;
1180             // If pedestrian is sitting, then calculate the remaing
1181             // sitting time
1182             if (p.state == MovementState.USING_SUO) {
1183                 // It is always: remainingUseTime >= 0, cf. "expand".
1184                 p.remainingUseTime = projectRemainingUseTime(p,
1185                                                              currentStateTime);
1186                 p.fatigue = 0.0; // After elapse of remainingUseTime the
1187                                  // fatigue is by assumption 0 again.
1188                                  // Note that fatigue may be
1189                                  // incorporated in projecting the
1190                                  // walking time.
1191             }
1192             // Conduct the following calculations if the pedestrian is
1193             // moving or sitting.
1194             if (p.state != MovementState.DISTANCE_ACCOMPLISHED) {
1195                 // The following loop is for finding the SUO that is
1196                 // nearest to the currently consided pedestrian.
1197                 minDistance = Double.MAX_VALUE;
1198                 nextSUO = null;
1199                 for (SUO_AdaptiveSeating suo : suos) {
1200                     // Consider only SUOS in walking direction; then the
1201                     // first part of the conditions is true; note that
1202                     // the operator ">=" is crucial. In "expand" after
1203                     // each node "openend" by a pedestrian, the
1204                     // pedestrian is slightly offset to avoid that she
1205                     // will be reconsidered there because of the
1206                     // inclusion of "=". The inclusion of "=" is crucial
1207                     // if several pedestrians reach an SUO at the same
1208                     // time.
1209                     if (((p.direction * suo.position) >=
1210                          (p.direction * p.position)) &
1211                         (Math.abs(suo.position - p.position) <
```

```
1212                            minDistance)) {
1213
1214                        nextSUO = suo;
1215                        minDistance = Math.abs(suo.position -
1216                                               p.position);
1217
1218                    }
1219                }
1220                // nextSUO = null would mean: pedestrian passed all SUOs
1221                if (nextSUO != null) {
1222                    // Note that the remainingUseTime had been set above
1223                    // (cf. just after loop entry and just after first
1224                    // if-condition)
1225                    duration = p.remainingUseTime +
1226                               projectWalkingTime(nextSUO.position, p);
1227                    if (duration < minDuration) {
1228                        minDuration = duration;
1229                        theNextSUO = nextSUO;
1230                        thePedestrian = p;
1231                    }
1232                }
1233                p.fatigue = fatigueResetCopy; // Reset fatigue and
1234                                              // untrimmed fatigue for
1235                                              // the forthcoming
1236                                              // calculation steps
1237                                              // (outside this method).
1238                p.untrimmedFatigue = untrimmedFatigueResetCopy;
1239
1240            }
1241        }
1242        decEv.timeHorizon = minDuration;
1243        decEv.nextPassedSUO = theNextSUO;
1244        decEv.nextPassingPedestrian = thePedestrian;
1245        return decEv;
1246    }
1247
1248    /**
1249     * Projects how long a pedestrian needs for accomplishing the
1250     * distance up to the position given the "targetPosition"-parameter.
1251     * This calculation could incorporate several factors, like fatigue
1252     * or topography of the course.
1253     * However, the calculation here is kept simple.
1254     * @param targetPosition The position to where the walking time
1255     *                       shall be projected. At this position there
1256     *                       is usually an SUO or an end of the course.
1257     * @param pedestrian The pedestrian for whom the walking time shall
1258     *                   be calculated.
1259     * @return The duration (in seconds) needed for the "pedestrian" to
1260     *         reach the "targetPosition".
1261     */
1262    public double projectWalkingTime(double targetPosition,
1263                                     Pedestrian pedestrian) {
1264
1265        double time = 0.0;
1266        double distance = Math.abs(targetPosition -
1267                                   pedestrian.position);
1268        time = distance / pedestrian.normalVelocity;
1269
1270        return time;
1271
```

```
1272        }
1273
1274        /**
1275         * Projects the remaining time of a pedestrian using an SUO. If the
1276         * pedestrian given by the parameter "p" is not in the movement
1277         * state "USING_SUO" then 0 is returned (that means, if it is not
1278         * checked somewhere else, whether the pedestrian is using an SUO
1279         * the output 0 can either mean, that the pedestrian is not using an
1280         * SUO at the given time or that the remaining use time is 0 indeed)
1281         * @param p The pedestrian.
1282         * @param currentTime The current state time.
1283         * @return The projected use time for the pedestrian. If the
1284         *         pedestrian's movement state is not "USING_SUO" then 0 is
1285         *         returned (0 is also returned if the remaining use time is
1286         *         actually 0).
1287         */
1288        public double projectRemainingUseTime(Pedestrian pedestrian,
1289                                              double currentTime) {
1290
1291            double remUseTime;
1292
1293            if (pedestrian.state == MovementState.USING_SUO) {
1294                // In brackets: time since start using the SUO
1295                // (i.e. since sitting down)
1296                remUseTime = pedestrian.pauseTime -
1297                             (currentTime -
1298                              pedestrian.lastStateAlterationTime);
1299                // in the method "expand" it is ensured that the maximum
1300                // sitting time is the pause time (i.e. the term in brackets
1301                // never be a higher value than pause time and hence always
1302                // remUseTime >= 0).
1303            }
1304            else {
1305                remUseTime = 0.0;
1306            }
1307
1308            return remUseTime;
1309
1310        }
1311
1312        /**
1313         * Calculates the position of a pedestrian, given that she walks
1314         * throughout the time specified by "walkingTime". The pedestrian's
1315         * movement state is not checked within this method. The position
1316         * is written in the specified pedestrian object.
1317         * @param pedestrian The pedestrian whose position is calculated.
1318         *                   The new position is directly set to this
1319         *                   pedestrian object.
1320         * @param walkingTime The duration for which the position progress
1321         *                    is calculated.
1322         */
1323        public void projectPosition(Pedestrian pedestrian,
1324                                    double walkingTime) {
1325
1326            double posDiff = pedestrian.normalVelocity * walkingTime;
1327            pedestrian.position += (pedestrian.direction) * posDiff;
1328
1329        }
1330
1331        /**
```

```
1332      * Estimates the degree of fatigue for the specified sitting time.
1333      * The movement state of the pedestrian is not checked.
1334      * The degree of fatigue will decrease but never be less than 0.
1335      * (The untrimmed fatigue will never be less than 0, too).
1336      * The new degree of fatigue is written in the specified pedestrian
1337      * object (as well as the untrimmed degree of fatigue).
1338      * @param pedestrian The pedestrian whose degree of fatigue shall be
1339      *                   estimated. The new degree of fatigue is
1340      *                   directly set to this pedestrian object.
1341      * @param usingTime The duration how long the pedestrians uses an
1342      *                  SUO "adaptive seating", i.e. how long she sits.
1343      */
1344     public void estimateFatigueWhileUsingSUO(Pedestrian pedestrian,
1345                                              double usingTime) {
1346         double newFatigue = pedestrian.fatigue -
1347                             (pedestrian.recoveryRate * usingTime);
1348
1349
1350         if (newFatigue < 0.0) {
1351             pedestrian.fatigue = 0.0;
1352             pedestrian.untrimmedFatigue = 0.0;
1353         }
1354         else {
1355             pedestrian.fatigue = newFatigue;
1356             pedestrian.untrimmedFatigue = newFatigue;
1357         }
1358     }
1359
1360     /**
1361      * Estimates the degree of fatigue for the specified walking time.
1362      * The movement state of the pedestrian is not checked.
1363      * The degree of fatigue will increase but never be higher than 1.
1364      * The new degree of fatigue is written in the specified pedestrian
1365      * object. There is also an untrimmed fatigue value written in the
1366      * pedestrian object. The untrimmed fatigue value can be higher than
1367      * 1. If specified, the method also calculates the distance that the
1368      * pedestrian walked at a critical degree of fatigue.
1369      * @param pedestrian The pedestrian whose degree of fatigue shall be
1370      *                   estimated. The new degree of fatigue is
1371      *                   directly set to this pedestrian object.
1372      * @param walkingTime The duration how long the pedestrians is
1373      *                    walking.
1374      * @param wFatiguedDistance If true then also the distance that the
1375      *                          pedestrian walks with a critical
1376      *                          degree of fatigue is updated in
1377      *                          the specified pedestrian.
1378      */
1379     public void estimateFatigueWhileWalking(Pedestrian pedestrian,
1380                                             double walkingTime,
1381                                             boolean wFatiguedDistance) {
1382
1383         double newFatigue =  pedestrian.fatigue +
1384                              pedestrian.fatigueRate * walkingTime;
1385
1386         pedestrian.untrimmedFatigue = pedestrian.untrimmedFatigue +
1387                                       pedestrian.fatigueRate *
1388                                       walkingTime;
1389
1390         double criticalWalkingTime; // Time that pedestrian walks with a
1391                                     // critical degree of fatigue
```

```
1392
1393            if (wFatiguedDistance) {
1394                // Calculate also the distance that the pedestrian walks
1395                // with a critical degree of fatigue
1396                criticalWalkingTime = 0.0;
1397                if (pedestrian.fatigue >= Director.CRITICAL_FATIGUE) {
1398                    criticalWalkingTime = walkingTime; // the whole time is
1399                                                       // critical
1400                }
1401                else if (newFatigue >= Director.CRITICAL_FATIGUE) {
1402                    // Note that pedestrian.fatigue has not been trimmed to
1403                    // at max. 1.0. So, the difference in brackets is the
1404                    // fatigue increase, if there was no upper bound.
1405                    criticalWalkingTime = (newFatigue -
1406                                           Director.CRITICAL_FATIGUE) /
1407                                           pedestrian.fatigueRate;
1408                }
1409                pedestrian.fatiguedDistance += criticalWalkingTime *
1410                                               pedestrian.normalVelocity;
1411            }
1412
1413
1414            if (newFatigue > 1.0) {
1415                pedestrian.fatigue = 1.0;
1416            }
1417            else {
1418                pedestrian.fatigue = newFatigue;
1419            }
1420        }
1421
1422    }
```

StateNode.java

```
1  package main_package;
2
3  import util.NumberFormatter;
4
5  /**
6   * The nodes of the state space tree.
7   * @author Marvin Hubl
8   * @since 2019-10-11
9   *
10  */
11 public class StateNode {
12
13      /**
14       * Node identifier.
15       */
16      public int id;
17
18      /**
19       * For determination of the "id".
20       */
21      public static int number = 0;
22
23      /**
24       * The child nodes of this node in a tree.
25       */
26      public StateNode[] children;
```

```
27
28      /**
29       * The parent node of this node in a tree.
30       */
31      public StateNode parent;
32
33      /**
34       * The point in time to which the state is represented in this state
35       * node.
36       */
37      public double stateTime;
38
39      /**
40       * The state of the SUOs represented in this state node.
41       */
42      public SUO_AdaptiveSeating[] suos;
43
44      /**
45       * The state of the pedestrians represented in this state node.
46       */
47      public Pedestrian[] pedestrians;
48
49      /**
50       * Number of pedestrians who have not yet accomplished the course.
51       */
52      public int numberOfPedestriansEnRoute;
53
54      /**
55       * The value after a heuristic evaluation of the node
56       * (heuristcValue = e = g + f)
57       */
58      public double heuristicValue;
59
60      /**
61       * Records relevant information for scoring a path evententually.
62       */
63      public ScoreRecord scoreRecord;
64
65      /**
66       * Containts 'verbal' information about the path leading to the node
67       */
68      public String pathInfo;
69
70      /**
71       * Initializations:
72       * id --> running number,
73       * children --> {null, null},
74       * parent --> null,
75       * stateTime --> 0.0,
76       * suos --> null,
77       * pedestrians --> null,
78       * numberOfPedestriansEnRoute --> 0,
79       * heuristicValue --> 0.0,
80       * scoreRecord --> null,
81       * pathInfo --> "Init."
82       */
83      public StateNode() {
84          number++;
85          id = number;
86          children = new StateNode[]{null, null};
```

```
87          parent = null;
88          stateTime = 0.0;
89          suos = null;
90          pedestrians = null;
91          numberOfPedestriansEnRoute = 0;
92          heuristicValue = 0.0;
93          scoreRecord = null;
94          pathInfo = "Init.";
95      }
96
97      /**
98       * Creates a copy of this state node by 'cloning' attribute values,
99       * i.e. by creating new attribute instances in the copy with the
100      * same value as in this state node. (The heuristic value is also
101      * copied, as it is considered to be more sensible than setting it
102      * to, e.g., 0.0.)The "id" is not copied but created as usual by the
103      * constructer. The items in "children" as well as the "parent" are
104      * set to "null" in the copy (however the children-array is not null
105      * but an empty array). For the "suos" and "pedestrians" the
106      * "copy()"-methods of the respective objects are elementwisely
107      * executed. For the "scoreRecord" the "copy()"-method of the
108      * respective object is executed.
109      * @return A copy of this state node.
110      */
111     public StateNode copy() {
112
113         StateNode copy = new StateNode();
114
115         copy.children = new StateNode[] {null, null};
116         copy.parent = null;
117         copy.stateTime = this.stateTime;
118
119         copy.suos = new SUO_AdaptiveSeating[this.suos.length];
120         for (int i = 0; i < copy.suos.length; i++) {
121             copy.suos[i] = this.suos[i].copy();
122         }
123         copy.pedestrians = new Pedestrian[this.pedestrians.length];
124         for (int i = 0; i < copy.pedestrians.length; i++) {
125             copy.pedestrians[i] = this.pedestrians[i].copy();
126         }
127         copy.numberOfPedestriansEnRoute =
128                                         this.numberOfPedestriansEnRoute;
129
130         copy.heuristicValue = this.heuristicValue;
131
132         if (this.scoreRecord != null) {
133             copy.scoreRecord = this.scoreRecord.copy();
134         }
135
136         copy.pathInfo = this.pathInfo;
137
138         return copy;
139     }
140
141     public String toString() {
142
143         // "\r\n" creates line break depending on the operating system
144         // (unlike only "\n")
145
146         String output = "STATE NODE\r\n" +
```

```
147                         "Id                          : " +
148                            String.valueOf(id) + "\r\n" +
149                         "State time                  : " +
150                            NumberFormatter.formatDouble(stateTime) +
151                            " s\r\n" +
152                         "No. of pedestrians en route: " +
153                            String.valueOf(numberOfPedestriansEnRoute) +
154                               "\r\n";
155
156         if (parent != null) {
157             output +=   "Parent (ID)                 : " +
158                          String.valueOf(parent.id) + "\r\n";
159         }
160         else {
161             output +=   "Parent                      : null\r\n";
162         }
163         if (children[0] != null) {
164             output +=   "Left child (ID)             : " +
165                         String.valueOf(children[0].id) + "\r\n";
166         }
167         else {
168             output +=   "Left child                  : null\r\n";
169         }
170         if (children[1] != null) {
171             output +=   "Right child (ID)            : " +
172                         String.valueOf(children[1].id) + "\r\n";
173         }
174         else {
175             output +=   "Right child                 : null\r\n";
176         }
177
178         if (pedestrians != null) {
179             for (Pedestrian p : pedestrians) {
180                 if (p != null) {
181                     output += p.id + ", " + p.state + ", pos. = " +
182                               NumberFormatter.formatDouble(p.position) +
183                               " cm\r\n";
184                 }
185                 else {
186                     output += "Pedestrian is 'null'\r\n";
187                 }
188             }
189         }
190         else {
191             output += "Pedestrians are 'null'\r\n";
192         }
193
194         if (suos != null) {
195             for (int i = 0; i < suos.length; i++) {
196                 if (suos[i] != null) {
197                     output += suos[i].id + ", pos. = " +
198                         NumberFormatter.formatDouble(suos[i].position) +
199                               " cm" + ", occupied by " +
200                               String.valueOf(suos[i].occupancy);
201                 }
202                 else {
203                     output += "SUO is 'null'";
204                 }
205                 if (i != suos.length - 1) {
206                     output += "\r\n";
```

```
207                 }
208             }
209         }
210         else {
211             output += "SUOs are 'null'";
212         }
213 
214         output += "\r\n";
215         if (scoreRecord != null) {
216             output += scoreRecord.toString();
217         }
218         else {
219             output += "No score record.";
220         }
221 
222         return output;
223 
224     }
225 
226     public boolean equals(Object otherObject) {
227         if (otherObject instanceof StateNode) {
228             StateNode otherNode = (StateNode) otherObject;
229             return this.id == otherNode.id;
230         }
231         else {
232             return false;
233         }
234     }
235 
236 }
```

BranchEvent.java

```
1  package main_package;
2 
3  import util.NumberFormatter;
4 
5  /**
6   * A 3-Tupel Tripel (for the return value of "timeToNextSUO").
7   * @author Marvin Hubl
8   * @since 2018-22-08
9   */
10 
11 public class BranchEvent {
12 
13     /**
14      * A Duration [in seconds].
15      */
16     public double timeHorizon;
17 
18     /**
19      * A smart urban object "adaptive seating".
20      */
21     public SUO_AdaptiveSeating nextPassedSUO;
22 
23     /**
24      * A pedestrian.
25      */
26     public Pedestrian nextPassingPedestrian;
27 
```

```
28      /**
29       * Initializations:
30       * timeHorizon --> max. value of double-type (not infinity!),
31       * nextPassedSUO --> null,
32       * pedestrian --> null.
33       */
34      public BranchEvent() {
35          timeHorizon = Double.MAX_VALUE;
36          nextPassedSUO = null;
37          nextPassingPedestrian = null;
38      }
39
40      public String toString() {
41
42          String output;
43
44          output =        "BRANCH EVENT\r\n" +
45                          "Time horizon                : " +
46                          NumberFormatter.formatDouble(timeHorizon) +
47                          " s\r\n";
48          if (nextPassedSUO == null) {
49              output += "Next passed SUO              : null\r\n";
50          }
51          else {
52              output += "Next passed SUO (ID)         : " +
53                          nextPassedSUO.id + "\r\n";
54          }
55          if (nextPassingPedestrian == null) {
56              output += "Next passing pedestrian      : null\r\n";
57          }
58          else {
59              output += "Next passing pedestrian (ID): " +
60                          nextPassingPedestrian.id;
61          }
62
63
64          return output;
65      }
66
67  }
```

SUO_AdaptiveSeating.java

```
1   package main_package;
2
3   import util.NumberFormatter;
4
5   /**
6    * A smart urban object (SUO) "adaptive seating".
7    * @author MarvinHubl
8    * @since 2018-08-22
9    */
10  public class SUO_AdaptiveSeating {
11
12      /**
13       * Identifier.
14       */
15      public String id;
16
17      /**
```

```
18      * For determination of the attribute values for "id" and "index".
19      */
20     public static int number = 0;
21
22     /**
23      * Index for direct access in array structures.
24      */
25     public int index;
26
27     /**
28      * Position of the SUO (one-dimensional) [in cm].
29      */
30     public double position;
31
32     /**
33      * "True", if the SUO is adaptive.
34      */
35     public boolean adaptive;
36
37     /**
38      * Number of pedestrians who can use the SUO simultaneously.
39      */
40     public int capacity;
41
42     /**
43      * Number of pedestrians who are using the SUO at a given point in
44      * time.
45      */
46     public int occupancy;
47
48     /**
49      * Initializations:
50      * id --> "SUO-" + running no. as string (begun with 1),
51      * index --> running no. (begun with 0),
52      * position --> 0.0,
53      * adaptive --> true,
54      * capacity --> 2,
55      * occupancy --> 0.
56      */
57     public SUO_AdaptiveSeating() {
58         number++;
59         id = "SUO-" + String.valueOf(number);
60         index = number - 1;
61         position = 0.0;
62         adaptive = true;
63         capacity = 2;
64         occupancy = 0;
65     }
66
67     /**
68      * Creates a copy of this SUO by 'cloning' all the attribute values,
69      * i.e. by creating new attribute instances in the copy with the
70      * same value as in this SUO.
71      * @return A copy of this state node.
72      */
73     public SUO_AdaptiveSeating copy() {
74
75          SUO_AdaptiveSeating copy = new SUO_AdaptiveSeating();
76          number--; // "number" is increased by instantiation but shall
77                    // not be increased by copying.
```

```
78
79          copy.id = this.id;
80          copy.index = this.index;
81          copy.position = this.position;
82          copy.adaptive = this.adaptive;
83          copy.capacity = this.capacity;
84          copy.occupancy  = this.occupancy;
85
86          return copy;
87
88      }
89
90      public String toString() {
91
92          String output;
93
94          output = "SUO ADAPTIVE SEATING\r\n" +
95                   "Id       : " + id + "\r\n" +
96                   "Index    : " + String.valueOf(index) + "\r\n" +
97                   "Position : " + NumberFormatter.formatDouble(position)
98                                   + " cm \r\n" +
99                   "Adaptive : " + String.valueOf(adaptive) + "\r\n" +
100                  "Capacity : " + String.valueOf(capacity) + "\r\n" +
101                  "Occupancy: " + String.valueOf(occupancy);
102
103         return output;
104
105     }
106
107 }
```

Pedestrian.java

```
1  package main_package;
2
3  import util.NumberFormatter;
4
5  /**
6   * A pedestrian.
7   * @author MarvinHubl
8   * @since 2019-10-12
9   */
10 public class Pedestrian {
11
12     /**
13      * Identifier.
14      */
15     public String id;
16
17     /**
18      * For determination of the attribute values for "id" and "index".
19      */
20     public static int number = 0;
21
22     /**
23      * Index for direct access in array structures.
24      */
25     public int index;
26
27     /**
```

```
28      * Position of the pedestrian (one-dimensional) [in cm].
29      */
30     public double position;
31
32     /**
33      * Walking direction of the pedestrian.
34      * This must be either +1 or -1 !
35      */
36     public int direction;
37
38     /**
39      * "True", if the pedestrian is handicapped in some way.
40      * @deprecated Not used anywhere (not needed)
41      */
42     public boolean handicap;
43
44     /**
45      * Current SUO or "null", if no SUO is used currently.
46      */
47     public SUO_AdaptiveSeating currentlyUsedSUO;
48
49     /**
50      * The index of the last used SUO by this pedestrian. If the
51      * pedestrian has not used any SUO, this index has a negative value
52      * (-1).
53      */
54     public int lastUsedSUOIndex;
55
56     /**
57      * Point in time, of the last, previous movement state of the
58      * pedestrian [in seconds].
59      */
60     public double lastStateAlterationTime;
61
62     /**
63      * The state of the pedestrian (walking, using an SUO, or "out of
64      * system").
65      */
66     public MovementState state;
67
68     /**
69      * The pedestrian's degree of fatigue in percent, i.e. within
70      * [0, 1].
71      */
72     public double fatigue;
73
74     /**
75      * This reflects the fatigue of the pedestrian, if it could grow
76      * above the value of 1.0 and is used to calculate the distance that
77      * a rest comes to late.
78      */
79     public double untrimmedFatigue;
80
81     /**
82      * Fatigue increase per time unit [in seconds] while walking in
83      * percent, i.e. within [0, 1]. (Inverse value of the pauseTime.)
84      */
85     public double fatigueRate;
86
87     /**
```

```
88          * Fatigue decrease per time unit [in seconds] while using an SUO
89          * in percent, i.e. in [0 ,1]. (The recovery rate is the inverse
90          * value of the pause time.)
91          */
92         public double recoveryRate;
93
94         /**
95          * Normal walking velocity without fatigue (expected value) [in
96          * cm/sec].
97          */
98         public double normalVelocity;
99
100        /**
101         * Residual walking velocity if the pedestrian's fatigue is 100% [in
102         * cm/s].
103         * @deprecated Not used anywhere.
104         */
105        public double terminalVelocity;
106
107        /**
108         * The duration how long the pedestrian uses an SUO "adaptive
109         * seating" [in seconds]. (Reciprocal of the recovery rate.)
110         */
111        public double pauseTime;
112
113        /**
114         * If the pedestrian uses an SUO, the remaining use time
115         * [in seconds].
116         */
117        public double remainingUseTime;
118
119        /**
120         * The total distance that the pedestrian walked at critical degree
121         * of fatigue.
122         */
123        public double fatiguedDistance;
124
125        /**
126         * The record for the safety scoring of used SUOs (and for the final
127         * waypoint).
128         */
129        public SafetyScoreRecord safetyScoreRecord;
130
131        /**
132         * Initializations:
133         * id --> "Pedestrian" + running no. as String (begun with 1),
134         * index --> running no. (begun with 0),
135         * position --> 0.0
136         * direction --> 1
137         * handicap --> true,
138         * currentlyUsedSUO --> null,
139         * lastUsedSUOIndex --> -1,
140         * lastStateAlterationTime --> 0.0,
141         * state --> MovementState.WALKING,
142         * fatigue --> 0.0,
143         * untrimmedFatigue --> 0.0,
144         * fatigueRate --> (1/600) = 0.001666...,
145         * recoveryRate --> (1/300) = 0.003333...,
146         * normalVelocity --> 100.0,
147         * terminalVelocity --> 100.0,
```

```
148         * pauseTime --> 300.0,
149         * remainingUseTime --> 0.0,
150         * fatiguedDistance --> 0.0.
151         * safetyScoreRecord --> new SafetyScoreRecord
152         */
153        public Pedestrian() {
154            number++;
155            id = "Pedestrian-" + String.valueOf(number);
156            index = number - 1;
157            direction = 1;
158            handicap = true;
159            currentlyUsedSUO = null;
160            lastUsedSUOIndex = -1;
161            lastStateAlterationTime = 0.0;
162            state = MovementState.WALKING;
163            fatigue = 0.0;
164            untrimmedFatigue = 0.0;
165            fatigueRate = (1.0/600.0);
166            recoveryRate = (1.0/300.0);
167            normalVelocity = 100.0;
168            terminalVelocity = 100.0;
169            pauseTime = 300.0;
170            remainingUseTime = 0.0;
171            fatiguedDistance = 0.0;
172            safetyScoreRecord = new SafetyScoreRecord();
173        }
174
175        /**
176         * Creates a copy of this pedestrian state node by 'cloning'
177         * attribute values, i.e. by creating new attribute instances in the
178         * copy with the same value as in this pedestrian object.
179         * Note that the reference to the "currentlyUsedSUO" of this
180         * instance is also used for the copy instance, i.e. no new instance
181         * is created. (hence, most often it is necessary to "manually" set
182         * the copy's "currentlyUsedSUO" to another reference referring to
183         * the SUO copy; therefore the index of the referred SUO may be
184         * useful).
185         * @return A copy of this state node.
186         */
187        public Pedestrian copy() {
188
189            Pedestrian copy = new Pedestrian();
190            number--; // "number" is increased by instantiation but shall
191                      // not be increased by copying.
192
193            copy.id = this.id;
194            copy.index = this.index;
195            copy.position = this.position;
196            copy.direction = this.direction;
197            copy.handicap = this.handicap;
198            copy.currentlyUsedSUO = this.currentlyUsedSUO;
199            copy.lastUsedSUOIndex = this.lastUsedSUOIndex;
200            copy.lastStateAlterationTime = this.lastStateAlterationTime;
201            copy.state = this.state;
202            copy.fatigue = this.fatigue;
203            copy.untrimmedFatigue = this.untrimmedFatigue;
204            copy.fatigueRate = this.fatigueRate;
205            copy.recoveryRate = this.recoveryRate;
206            copy.normalVelocity = this.normalVelocity;
207            copy.terminalVelocity = this.terminalVelocity;
```

```
208         copy.pauseTime = this.pauseTime;
209         copy.remainingUseTime = this.remainingUseTime;
210         copy.fatiguedDistance = this.fatiguedDistance;
211         copy.safetyScoreRecord = this.safetyScoreRecord.copy();
212 
213         return copy;
214     }
215 
216     public String toString() {
217 
218         String output;
219 
220         output =  "PEDESTRIAN\r\n" +
221                   "Id                          : " + id + "\r\n" +
222                   "Index                       : " +
223                      String.valueOf(index) + "\r\n" +
224                   "Position                    : " +
225                      String.valueOf(position) + " cm\r\n" +
226                   "Direction                   : " +
227                      String.valueOf(direction) + "\r\n" +
228                   "Handicap                    : " +
229                      String.valueOf(handicap) + "\r\n";
230         if (currentlyUsedSUO == null) {
231             output += "Currently used SUO        : null\r\n";
232         }
233         else {
234             output += "Currently used SUO (ID)   : " +
235                       currentlyUsedSUO.id + "\r\n";
236         }
237         output += "Last used SUO index       : " + lastUsedSUOIndex +
238                   "\r\n" +
239                   "State                       : " +
240                      String.valueOf(state) + "\r\n" +
241                   "Fatigue                     : " +
242                      NumberFormatter.formatDouble(fatigue) + "\r\n" +
243                   "Fatigue rate                : " +
244                      NumberFormatter.formatDouble(fatigueRate) +
245                      " /s\r\n" +
246                   "Recovery Rate               : " +
247                      NumberFormatter.formatDouble(recoveryRate) +
248                      " /s\r\n" +
249                   "Normal velocity             : " +
250                      NumberFormatter.formatDouble(normalVelocity) +
251                   " cm/s \r\n" +
252                   "Terminal velocity           : " +
253                      NumberFormatter.formatDouble(terminalVelocity) +
254                      " cm/s \r\n" +
255                   "Pause time                  : " +
256                      NumberFormatter.formatDouble(pauseTime) +
257                      " s\r\n" +
258                   "Last state alteration time: " +
259                      NumberFormatter.formatDouble(
260                                           lastStateAlterationTime) +
261                   " s\r\n" +
262                   "Remaining use time          : " +
263                      NumberFormatter.formatDouble(remainingUseTime) +
264                      " s\r\n" +
265                   "Fatigued distance           : " +
266                      NumberFormatter.formatDouble(fatiguedDistance);
267 
```

```
268
269         return output;
270
271     }
272 }
```

MovementState.java

```
1  package main_package;
2
3  /**
4   * The state of movement.
5   * WALKING: Pedestrian is walking,
6   * USING_SUO: Pedestrian is currently using a smart urban object (e.g.
7   * sits on it).
8   * DISTANCE_ACCOMPLISHED: Pedestrian has finished the distance to go.
9   * @author Marvin Hubl
10  * @since 2018-08-22
11  */
12 public enum MovementState {
13
14     WALKING,
15     USING_SUO,
16     DISTANCE_ACCOMPLISHED;
17
18 }
```

SafetyScoreRecord.java

```
1  package main_package;
2
3  import java.util.ArrayList;
4
5  /**
6   * Containts relevant information to score a path with the safety
7   * functions.
8   * @author Marvin Hubl
9   * @since 2019-10-11
10  *
11  */
12 public class SafetyScoreRecord {
13
14     /**
15      *
16      * The scores for the appropriate place of a used SUO.
17      * Note that safetyAScores[j] must belong to safetyBScores[j].
18      */
19     public ArrayList<Double> safetyAScores;
20
21     /**
22      * The scores for the possibility of using assistance of a used SUO.
23      * Note that safetyBScores[j]  must belong to safetyAScores[j].
24      */
25     public ArrayList<Double> safetyBScores;
26
27     /**
28      * This is the scoring applied to the last waypoint.
29      */
30     public double finalSafetyAScore;
```

```
31
32      /**
33       * Initialization:
34       * safetyAScores --> new ArrayList<Double>(),
35       * safetyBScores --> new ArrayList<Double>(),
36       * finalSafetyScore --> 0.0.
37       */
38      public SafetyScoreRecord() {
39          safetyAScores = new ArrayList<Double>();
40          safetyBScores = new ArrayList<Double>();
41          finalSafetyAScore = 0.0;
42      }
43
44      /**
45       * Calculates the overall safety score. Basically, this is an
46       * average score of the composed safety score. The composed safety
47       * score composes the safety score for the availability at the right
48       * place and for the usability of assitance.
49       * @return The overall safety score  (which is an averaged composed
50       *         safety score).
51       */
52      public double calculateOverallScore() {
53          double result = 0.0;
54
55          // TODO Console output: Check
56  //      System.out.println("----------------------------");
57
58          for (int j = 0; j < safetyAScores.size(); j++) {
59              result += Director.LAMBDA_A * safetyAScores.get(j) +
60                        Director.LAMBDA_B * safetyBScores.get(j);
61          }
62
63          result += finalSafetyAScore;
64
65  //      System.out.println("final safety-a-score = " +
66  //                         finalSafetyAScore);
67  //      System.out.println("Result before averaging = " + result);
68
69          result = (result) / ((double) (safetyAScores.size() + 1));
70
71          // TODO Console output: Check
72  //      System.out.println("Result after averaging = " + result);
73
74          return result;
75      }
76
77      /**
78       * Calculates a safety score for the appropriate place of a used SUO
79       * (or endpoint of the course)
80       * @param pedestrian The pedestrian who is just to start using an
81       *                   SUO (or accomplishes the course)
82       * @param accomplishing This parameter is true if the pedestrian is
83       *                      just accomplishing the course. Then the
84       *                      safety score is set to the
85       *                      "finalSafetyAScore".
86       */
87      public void addSafetyAScore(Pedestrian pedestrian,
88                                  boolean accomplishing) {
89
90          double timeDeviation = (Director.CRITICAL_FATIGUE -
```

```
91                                  pedestrian.untrimmedFatigue) /
92                                  pedestrian.fatigueRate;
93
94          double distanceDeviation = timeDeviation *
95                                     pedestrian.normalVelocity;
96
97          double maxDistance = pedestrian.normalVelocity /
98                               pedestrian.fatigueRate;
99
100         double normalizedDistanceDeviation = distanceDeviation /
101                                              maxDistance;
102
103         double safetyAScore =
104                             safetyFunctionA(normalizedDistanceDeviation);
105
106         if (!accomplishing) {
107             safetyAScores.add(safetyAScore);
108         }
109         else {
110             finalSafetyAScore = safetyAScore;
111         }
112
113     }
114
115     /**
116      * Calculates a safety score for the usability of the assistance for
117      * sitting downn (only).
118      *
119      * @param numberOfOtherUsers The amount of other pedestrian who had
120      *                           been using the SUO while a pedestrian
121      *                           started to use it.
122      */
123     public void addSafetyBScoreForSittingDown(int numberOfOtherUsers) {
124         double safetyBScore1 = 0.0;
125
126         if (numberOfOtherUsers == 0) {
127             safetyBScore1 = Director.GAMMA_SIT_DOWN;
128         }
129         else {
130             safetyBScore1 = 0.0;
131         }
132         safetyBScores.add(safetyBScore1);
133     }
134
135     /**
136      * Calculates a safety score for the usability of the assistance for
137      * standing up (only).
138      *
139      * @param numberOfOtherUsers The amount of other pedestrian who had
140      *                           been using the SUO while a pedestrian
141      *                           finished with using it.
142      */
143     public void addSafetyBScoreForStandingUp(int numberOfOtherUsers) {
144         double safetyBScore2 = 0.0;
145         double safetyBScore1;
146         double safetyBScore;
147
148         if (numberOfOtherUsers == 0) {
149             safetyBScore2 = Director.GAMMA_STAND_UP;
150         }
```

```
151         else {
152             safetyBScore2 = 0.0;
153         }
154         safetyBScore1 = safetyBScores.remove(safetyBScores.size() - 1);
155         safetyBScore = safetyBScore1 + safetyBScore2;
156 
157         safetyBScores.add(safetyBScore);
158     }
159 
160     /**
161      * Implements the gaussian-based sigmoid safety function, i.e. the
162      * safety function w.r.t. to adaptivity of availability.
163      * @param z The indicator for availabiliy at an appropriate place.
164      * @return The safety value w.r.t. to adaptivity of availability.
165      */
166     public double safetyFunctionA(double z) {
167 
168         double result = 0.0;
169 
170         /**
171          * Parameters for the function.
172          */
173         double w_le = (Math.abs(Director.Z_LE)) /
174                 (Math.sqrt(-(Math.log(Math.pow(Director.S_LE, 2.0)))));
175         double w_ri = (Math.abs(Director.Z_RI)) /
176                 (Math.sqrt(-(Math.log(Math.pow(Director.S_RI, 2.0)))));
177 
178         if (z < 0) {
179             result = Math.exp(-(Math.pow(z, 2.0)) * 0.5 *
180                                             (1 / (Math.pow(w_le, 2.0))));
181         }
182         if (z == 0) {
183             result = 1.0;
184         }
185         if (z > 0) {
186             result = Math.exp(-(Math.pow(z, 2.0)) * 0.5 *
187                                             (1 / (Math.pow(w_ri, 2.0))));
188         }
189 
190         return result;
191 
192     }
193 
194     /**
195      * Applies an aggregation logic of pedestrians' safety score
196      * records. The aggregation logic here is to determine the minimum
197      * overall safety score among the given pedestrians
198      * @param pedestrians The pedestrians among whom the safey scores
199      *                    shall be aggregated, i.e. among whom the
200      *                    minimum overall safety score shall be
201      *                    determined.
202      * @return The Minimum overall safety score among the given
203      *         pedestrians.
204      */
205     public static double aggregate(Pedestrian... pedestrians) {
206 
207         double minScore = Double.MAX_VALUE;
208         double score;
209 
210         for (Pedestrian p : pedestrians) {
```

```
211
212             score = p.safetyScoreRecord.calculateOverallScore();
213
214             //TODO Console output: check
215 //          System.out.println("Pedestrian-" + (p.index+1) +
216 //                             "'s safety score = " + score);
217
218             if (score < minScore) {
219                 minScore = score;
220             }
221
222         }
223
224         return minScore;
225     }
226
227     /**
228      * Creates a copy of this safety score record by cloning all the
229      * attribute values, i.e. by creating new attribute instances in the
230      * copy with the
231      * @return A copy of this instance.
232      */
233     public SafetyScoreRecord copy() {
234
235         SafetyScoreRecord copy = new SafetyScoreRecord();
236         double entry;
237
238         for (int j = 0; j < this.safetyAScores.size(); j++) {
239             entry = (double) this.safetyAScores.get(j);
240             copy.safetyAScores.add((Double) entry);
241
242             entry = (double) this.safetyBScores.get(j);
243             copy.safetyBScores.add((Double) entry);
244         }
245
246         copy.finalSafetyAScore = this.finalSafetyAScore;
247
248         return copy;
249
250     }
251
252     // TODO Implement SafetyScore.toString()
253     public String toString() {
254
255         String string = "SAFETY SCORE RECORD\r\n";
256
257         return string;
258     }
259
260
261 }
```

ScoreRecord.java

```
1 package main_package;
2
3 import java.util.Arrays;
4
5 import util.NumberFormatter;
6
```

```
7  /**
8   * Containts relevant information to assess paths (i.e. to calculate a
9   * score for the paths in the state space tree).
10  * @author Marvin Hubl
11  * @since 2018-08-22
12  *
13  */
14 public class ScoreRecord {
15
16     /**
17      * Refers to a leaf node for back tracing the path in the state
18      * space tree. (Always null at the the moment, because a ScoreRecord
19      * is implemented as attribute of the nodes; hence a
20      * 'back-reference' is not necessary.)
21      */
22     public StateNode leafNode;
23
24     /**
25      * The score assigned to the path (referenced by the 'leafNode')
26      */
27     public double score;
28
29     /**
30      * Variable 1: Distance per pedestrian that she walked with a
31      * critical fatigue.
32      */
33     public double[] x1;
34
35     /**
36      * Variable 2: Amount per pedestrian how often she interacted with
37      * an SUO, while another individual was using it, too.
38      */
39     public int[] x2;
40
41     /**
42      * Variable 3: Duration per pedestrian for accomlishing the course.
43      */
44     public double[] x3;
45
46     /**
47      * Variable 4: Degree of fatigue per pedestrian at the end of the
48      * course.
49      */
50     public double[] x4;
51
52     /**
53      * Initialization:
54      * leadNode --> null,
55      * score --> 0.0,
56      * x1 --> null,
57      * x2 --> null,
58      * x3 --> null,
59      * x4 --> null.
60      */
61     public ScoreRecord() {
62         leafNode = null;
63         score = 0.0;
64         x1 = null;
65         x2 = null;
66         x3 = null;
```

```
67          x4 = null;
68      }
69
70      /**
71       * Initialization:
72       * leadNode --> null,
73       * score --> 0.0,
74       * x1 --> new double[numberOfPedestrians], filled with 0.0,
75       * x2 --> new int[numberOfPedestrians], filled with 0,
76       * x3 --> new double[numberOfPedestrians], filled with
77       *        POSITIVE_INFINITY,
78       * x4 --> new double[numberOfPedestrians], filled with NaN.
79       * @param numberOfPedestrians The number of pedestrians for
80       *                            calculating
81       */
82      public ScoreRecord(int numberOfPedestrians) {
83          leafNode = null;
84          score = 0.0;
85          x1 = new double[numberOfPedestrians];
86          x2 = new int[numberOfPedestrians];
87          x3 = new double[numberOfPedestrians];
88          x4 = new double[numberOfPedestrians];
89
90          Arrays.fill(x1, 0.0);
91          Arrays.fill(x2, 0);
92          Arrays.fill(x3, Double.POSITIVE_INFINITY);
93          Arrays.fill(x4, Double.NaN);
94      }
95
96
97      public String toString() {
98
99          String string = "SCORE RECORD\r\n";
100
101 //------The leaf node reference is not necessary, see above at the
102 //------declaration of the leafNode attribute.
103 //          if (leafNode == null) {
104 //              string += "Leaf node     : null\r\n";
105 //          }
106 //          else {
107 //              string += "Leaf node (ID): " + leafNode.id + "\r\n";
108 //          }
109 //          string += "Score         : " +
110 //                    NumberFormatter.formatDouble(score) +
111 //                   "\r\n";
112
113          string += "x1            : ";
114          for (int i = 0; i < x1.length; i++) {
115              string += NumberFormatter.formatDouble(x1[i]) + " ; ";
116          }
117          string += "\r\n";
118
119          string += "x2            : ";
120          for (int i = 0; i < x2.length; i++) {
121              string += String.valueOf(x2[i]) + " ; ";
122          }
123          string += "\r\n";
124
125          string += "x3            : ";
126          for (int i = 0; i < x3.length; i++) {
```

```
127             string += NumberFormatter.formatDouble(x3[i]) + " ; ";
128         }
129         string += "\r\n";
130 
131         string += "x4            : ";
132         for (int i = 0; i < x4.length; i++) {
133             string += NumberFormatter.formatDouble(x4[i]) + " ; ";
134         }
135 
136         return string;
137 
138     }
139 
140     /**
141      * Creates a copy of this score record by cloning all the attribute
142      * values, i.e. by creating new attribute instances in the copy with
143      * the same values as in this score record. Note that for the leaf
144      * node "only" the reference is used, not a "hard" copy of the
145      * referenced instance.
146      * @return A copy of this score record.
147      */
148     public ScoreRecord copy() {
149 
150         ScoreRecord copy = new ScoreRecord();
151 
152         copy.leafNode = this.leafNode;
153         copy.score = this.score;
154         copy.x1 = new double[this.x1.length];
155         copy.x2 = new int[this.x2.length];
156         copy.x3 = new double[this.x3.length];
157         copy.x4 = new double [this.x4.length];
158         // Arrays for all x have the same length (viz. the number of
159         // pedestrians)
160         for (int i = 0; i < this.x1.length; i++) {
161             copy.x1[i] = this.x1[i];
162             copy.x2[i] = this.x2[i];
163             copy.x3[i] = this.x3[i];
164             copy.x4[i] = this.x4[i];
165         }
166 
167         return copy;
168 
169     }
170 
171     /**
172      * Calculates the (attribute value) score for this score record.
173      * @param mode 1 if score shall be calculated by a linear function,
174      *             2 if score shall be calculated by a quadratic
175      *             function. For all other parameter values, the score
176      *             is calculated with a linear function, too
177      */
178     public void calcScore(int mode) {
179 
180         // Objective will be to minimize the maximium variable values
181         // among all pedestrians
182 
183         double maxX1 = -(Double.MAX_VALUE); // max. walked distance with
184                                             // critical fatigue among
185                                             // all pedestrians
186 
```

```
187         int maxX2 = -(Integer.MAX_VALUE); // max. number of using an SUO
188                                           // while someone else is using
189                                           // it, among all pedestrians
190
191         // Scores for the variables
192         double s1;
193         double s2;
194         // Note that variables x3 and x4 are not scored but treated only
195         // as depended variables.
196
197         // Find the max. values among all pedestrians for the objective
198         // variables.
199         maxX1 = getMaxX(1);
200         maxX2 = (int) getMaxX(2);
201
202         if (mode == 2) {
203             s1 = quadraticScore(maxX1, Director.X1_OPT,
204                                        Director.X1_MIN,
205                                        Director.X1_MAX);
206             s2 = quadraticScore((double) maxX2, Director.X2_OPT,
207                                                 Director.X2_MIN,
208                                                 Director.X2_MAX);
209         }
210         else {
211             s1 = linearScore(maxX1, Director.X1_OPT,
212                                     Director.X1_MIN,
213                                     Director.X1_MAX);
214             s2 = linearScore((double) maxX2, Director.X2_OPT,
215                                              Director.X2_MIN,
216                                              Director.X2_MAX);
217         }
218
219         score = s1 * Director.X1_WEIGHT +
220                 s2 * Director.X2_WEIGHT;
221     }
222
223
224     /**
225      * Finds the maximum value of a variable among all pedestrians. The
226      * parameter specifies for 'which' variable vector x1 to x4.
227      * Note that the return value is always double and must potentially
228      * be casted to integer for x2.
229      * @param which 1 => x1, 2 => x2, ... 4 => x4.
230      * @return The maximium value for the specified variable vector.
231      */
232     public double getMaxX(int which) {
233
234         double maxX = -(Double.MAX_VALUE);
235
236         if (which == 1) {
237             for (double x : x1) {
238                 if (x > maxX) {
239                     maxX = x;
240                 }
241             }
242         }
243         else if (which == 2) {
244             for (int x : x2) {
245                 if ((double) x > maxX) {
246                     maxX = (double) x;
```

```
247                 }
248             }
249         }
250         else if (which == 3) {
251             for (double x : x3) {
252                 if (x > maxX) {
253                     maxX = x;
254                 }
255             }
256         }
257         else if (which == 4) {
258             for (double x : x4) {
259                 if (x > maxX) {
260                     maxX = x;
261                 }
262             }
263         }
264         else {
265             maxX = Double.NaN;
266         }
267 
268         return maxX;
269 
270     }
271 
272     /**
273      * Calculate a score as described in
274      * Hubl (2018) "Adaption Rule for Simultaneous Use of Smart Urban
275      * Objects from a Fairness Perspective". In: IEEE 20th Conference on
276      * Business Informatics (CBI 2018), p. 93.
277      * @param x The value to score
278      * @param xopt The optimal value (score is 1.0)
279      * @param xmin The minimum feasible value (score is 0.0)
280      * @param xmax The maximum feasible value (score is 0.0)
281      * @return The score for x given the specified function.
282      */
283     private double linearScore(double x,
284                                double xopt, double xmin, double xmax) {
285 
286         double s = 0.0;
287 
288             if (xmin <= x & x < xopt & xopt != xmin) {
289                 s = (x-xmin) / (xopt-xmin);
290             }
291             else if (xopt < x & x <= xmax & xopt != xmax) {
292                 s = (x-xmax) / (xopt-xmax);
293             }
294             else if (x == xopt) {
295                 s = 1.0;
296             }
297             else {
298                 s = Double.NaN;
299             }
300 
301         return s;
302     }
303 
304     /**
305      * Calculate a score as described in
306      * Hubl et al. (2015) "Coordination of Just-in-Time Deliveries with
```

```
307         * Multi- attribute Auctions". In: 12th International Conference on
308         * Wirtschaftsinformatik (WI 2015), p. 99
309         * (but with Lambda = 1)
310         * @param x The value to score
311         * @param xopt The optimal value (score is 1.0)
312         * @param xmin The minimum feasible value (score is 0.0)
313         * @param xmax The maximum feasible value (score is 0.0)
314         * @return The score for x given the specified function.
315         */
316        @SuppressWarnings("unused")
317        private double quadraticScore(double x, double xopt, double xmin,
318                                      double xmax) {
319
320            double s = 0.0;
321
322                if (xmin <= x & x < xopt) {
323                    s = (1 - Math.pow((x-xopt)/(xopt-xmin), 2.0));
324                }
325                else if (xopt < x & x <= xmax) {
326                    s = (1 - Math.pow((x-xopt)/(xopt-xmax), 2.0));
327                }
328                else if (x == xopt) {
329                    s = 1.0;
330                }
331                else {
332                    s = Double.NaN;
333                }
334
335            return s;
336        }
337
338    }
```

RandomValueDrawings.java

```
1   package main_package;
2
3   import util.NumberFormatter;
4
5   /**
6    * This class encapsulates a specified number of values and is used to
7    * simplify data structures for maintaining random drawings for
8    * simulation cases.
9    *
10   * The class has several attributes, where each is a 2D-array.
11   * In theses arrays, the first component's index is the index of a
12   * pedestrian and the second component's index is the index of a draw.
13   * That is the array containts for various pedestrians values for
14   * multiple drawings.
15   *
16   * For example: velocities[i][r] = The r-th randomly drawn walking
17   *                                 velocity for the i-th pedestrian.
18   *
19   *
20   * @author Marvin Hubl
21   * @since 2019-10-18
22   */
23  public class RandomValueDrawings {
24
25      /**
```

```
26      * Vector for the pedestrians' walking directions.
27      */
28     public int[][]    directions;
29
30     /**
31      * Vector for the pedestrians' walking velocities [cm/s].
32      */
33     public double[][] velocities;
34
35     /**
36      * Vector for the pedestrians' fatigue rates [1/s].
37      */
38     public double[][] fatigueRates;
39
40     /**
41      *  Vector for the pedestrians' pause times [s].
42      */
43     public double[][] pauseTimes;
44
45     /**
46      * Constructor.
47      * @param numberOfPedestrians Number of pedestrian in a simulation
48      *                            case.
49      * @param numberOfDrawingsOfPedestrians Number of drawings per
50      *                                      pedestrian.
51      */
52     public RandomValueDrawings (int numberOfPedestrians,
53                                 int numberOfDrawingsPerPedestrian) {
54
55         directions    = new int[numberOfPedestrians]
56                                [numberOfDrawingsPerPedestrian];
57
58         velocities    = new double[numberOfPedestrians]
59                                   [numberOfDrawingsPerPedestrian];
60
61         fatigueRates = new double[numberOfPedestrians]
62                                  [numberOfDrawingsPerPedestrian];
63
64         pauseTimes    = new double[numberOfPedestrians]
65                                   [numberOfDrawingsPerPedestrian];
66     }
67
68     public String toString() {
69
70         StringBuilder sb = new StringBuilder();
71
72         sb.append("=== RandomValueDrawings ===\r\n");
73
74         for (int i = 0; i < directions.length; i++) {
75
76             sb.append("Walking directions: ");
77             for (int j = 0; j < directions[i].length; j++) {
78                 sb.append(String.valueOf(directions[i][j]) + " ; ");
79             }
80             sb.append("\r\n");
81
82             sb.append("Walking velocities: ");
83             for (int j = 0; j < velocities[i].length; j++) {
84                 sb.append(NumberFormatter.
85                           formatDouble(velocities[i][j]) +
```

```
86                              " ; ");
87              }
88              sb.append("\r\n");
89
90              sb.append("Fatigue rates      : ");
91              for (int j = 0; j < fatigueRates[i].length; j++) {
92                  sb.append(NumberFormatter.
93                              formatDouble(fatigueRates[i][j]) +
94                              " ; ");
95              }
96              sb.append("\r\n");
97
98              sb.append("Pause times        : ");
99              for (int j = 0; j < pauseTimes[i].length; j++) {
100                 sb.append(NumberFormatter.
101                             formatDouble(pauseTimes[i][j]) +
102                             " ; ");
103             }
104             sb.append("\r\n");
105             if (i != directions.length - 1) {
106                 sb.append("------------------------------\r\n");
107             }
108         }
109         sb.append("==============================");
110
111         return sb.toString();
112     }
113
114 }
```

HeuristicValueComparator.java

```
1  package main_package;
2
3  import java.util.Comparator;
4
5  /**
6   * Comparator to sort by heuristic value. The heuristic value is the
7   * maximum among the distances that all pedestrians have walked with a
8   * critical degree of fatigue.
9   * @author Marvin Hubl
10  * @since 2019-10-13
11  */
12 public class HeuristicValueComparator implements Comparator<StateNode> {
13
14
15     @Override
16     public int compare(StateNode node1, StateNode node2) {
17         int result;
18
19         if (node1.heuristicValue < node2.heuristicValue) {
20             result = -1;
21         }
22         else if (node1.heuristicValue == node2.heuristicValue) {
23             result = 0;
24         }
25         else {
26             result = 1;
27         }
28
```

```
29          return result;
30      }
31
32  }
```

RemainingUseTimeComparator.java

```
1   package main_package;
2
3   import java.util.Comparator;
4
5   /**
6    *
7    * @author Marvin Hubl
8    * @since 2019-10-15
9    */
10  public class RemainingUseTimeComparator implements
11                                                  Comparator<Pedestrian> {
12
13      @Override
14      public int compare(Pedestrian p1, Pedestrian p2) {
15
16          int result;
17
18          if (p1.remainingUseTime < p2.remainingUseTime) {
19              result = -1;
20          }
21          else if (p1.remainingUseTime == p2.remainingUseTime) {
22              result = 0;
23          }
24          else {
25              result = 1;
26          }
27
28          return result;
29      }
30
31  }
```

Logger.java

```
1   package util;
2
3   import java.io.File;
4   import java.io.FileWriter;
5   import java.io.IOException;
6   import java.io.PrintWriter;
7   import java.text.DateFormat;
8   import java.util.GregorianCalendar;
9   import java.util.Locale;
10  import java.util.StringTokenizer;
11
12  /**
13   * Logs results and potentially intermediate states in a file and prints
14   * the information potentially also on screen
15   * @author Marvin Hubl
16   * @since 2019-10-14
17   *
18   */
```

```
19 public class Logger {
20
21     /**
22      * Specifies a log level (--> input parameter for write):
23      * NONE      --> Logger shall not log anythig.
24      * ALL       --> Logger shall log to log file and to console.
25      * TO_FILE   --> Logger shall log only to log file but not to
26      *               console.
27      * TO_SCREEN --> Logger shall log only to console but not to log
28      *               file.
29      */
30     public enum LogLevel {
31         NONE,
32         ALL,
33         TO_SCREEN,
34         TO_FILE
35     }
36
37     /**
38      * The log file and needed writers.
39      */
40     private File         file        = null;
41     private FileWriter   fileWriter  = null;
42     private PrintWriter  printWriter = null;
43
44     private GregorianCalendar calendar;
45
46     /**
47      * For retrieving the logger via "getInstance()".
48      */
49     private static Logger theInstance = null;
50     /**
51      * For retrieving the csv file logger via "getCSVInstance()".
52      */
53     private static Logger theCSVInstance = null;
54
55     /**
56      * The real time in milliseconds at the instatiation point in time
57      * of a logger object.
58      */
59     private final long millisAtStart = System.currentTimeMillis();
60
61     /**
62      * Creates a new log file.
63      */
64     public Logger() {
65         theInstance = this;
66         init(false);
67     }
68
69     /**
70      * Creates a new log instance or csv logging instance.
71      * @param If this is true the logger will create a CSV-File.
72      */
73     public Logger(boolean isCSV) {
74         if (isCSV) {
75             theCSVInstance = this;
76         }
77         else {
78             theInstance = this;
```

```
79          }
80          init(isCSV);
81      }
82
83      /**
84       * Returns the single instance for logging.
85       * @return The instance for logging.
86       */
87      public static Logger getInstance() {
88          if (theInstance == null) {
89              theInstance = new Logger();
90          }
91          return theInstance;
92      }
93
94
95      /**
96       * Returns the single instance for logging in a file with comma
97       * seperated values.
98       * @return The instance of a logger that writes in a csv-file.
99       */
100     public static Logger getCSVInstance() {
101         if (theCSVInstance == null) {
102             theCSVInstance = new Logger(true);
103         }
104         return theCSVInstance;
105     }
106
107     /**
108      * Initaliazes the logger.
109      * @param If this is true a CSV-File is created.
110      */
111     private void init(boolean isCSV) {
112
113
114         calendar                  = new GregorianCalendar();
115         String          timestamp = null;
116         String          day       = null;
117         String          month     = null;
118         String          year      = null;
119         String          hours;
120         String          minutes;
121         String          seconds;
122         StringTokenizer tokenizer = null;
123
124         timestamp = DateFormat.getDateTimeInstance(
125                                 DateFormat.MEDIUM,
126                                 DateFormat.MEDIUM,
127                                 Locale.GERMAN).
128                                   format(calendar.getTime());
129
130         System.out.println("Logger instantiation " + timestamp);
131         System.out.println();
132
133         tokenizer = new StringTokenizer(timestamp, ".: ");
134         day       = tokenizer.nextToken();
135         month     = tokenizer.nextToken();
136         year      = tokenizer.nextToken();
137         hours     = tokenizer.nextToken();
138         minutes   = tokenizer.nextToken();
```

```
139         seconds   = tokenizer.nextToken();
140
141         timestamp = year + "-" + month + "-" + day + "-" +
142                     hours + "-" + minutes + "-" + seconds;
143
144         if (isCSV) {
145             file = new File("logs/" + timestamp +".csv");
146         }
147         else {
148             file = new File("logs/" + timestamp +".log");
149         }
150         try {
151             fileWriter = new FileWriter(file, true);
152         } catch (IOException e) {
153             e.printStackTrace();
154         }
155         printWriter = new PrintWriter(fileWriter, true);
156
157
158         printWriter.println("Start at: " + year + "-" + month + "-" +
159                             day + " " + hours + ":" + minutes + ":" +
160                             seconds);
161         printWriter.println();
162
163     }
164
165     /**
166      * Writes information in the log file and prints it also to screen
167      * if wanted.
168      * @param headerInformation Some meta-information (e.g. the calling
169      *                          object). The current real time and
170      *                          milleseconds since start are always
171      *                          logged automatically (and need not to be
172      *                          as headerInformation).
173      * @param contentObject The actual content to log using the
174      *                      "toString()"- method of the object.
175      * @param logLevel LogLevel as specified above.
176      */
177     public void write(String headerInformation, Object contentObject,
178                       LogLevel logLevel) {
179
180         String content;
181         long elapsedMillis = System.currentTimeMillis() - millisAtStart;
182
183         if (contentObject == null) {
184             content = "Content is null";
185         }
186         else {
187             content = contentObject.toString();
188         }
189
190         if (logLevel == LogLevel.ALL | logLevel == LogLevel.TO_FILE) {
191
192             if (headerInformation != null && headerInformation != "") {
193                 printWriter.println("====== " + elapsedMillis +
194                         " ms after start - " + headerInformation);
195             }
196             else {
197                 printWriter.println("====== " + elapsedMillis +
198                         " ms after start");
```

```
199             }
200             printWriter.println(content);
201             printWriter.println("======");
202             printWriter.println();
203 
204         }
205 
206         if (logLevel == LogLevel.ALL | logLevel == LogLevel.TO_SCREEN) {
207             if (headerInformation != null && headerInformation != "") {
208                 System.out.println("====== " + elapsedMillis +
209                                    " ms after start - " +
210                                    headerInformation);
211             }
212             else {
213                 System.out.println("====== " + elapsedMillis +
214                                    " ms after start");
215             }
216             System.out.println(content);
217             System.out.println("======");
218             System.out.println();
219         }
220 
221     }
222 
223     /**
224      * Calls "write('null', contentObject, LogLevel.TO_FILE)"
225      * @param contentObject see: "write(String, Object, Boolean)"
226      */
227     public void write(Object contentObject) {
228         write(null, contentObject, LogLevel.TO_FILE);
229     }
230 
231     /**
232      * Calls "write('null', contentObject, logLevel)"
233      * @param contentObject see: "write(String, Object, Boolean)"
234      * @param toScreen see: "write(String, Object, LogLevel)"
235      */
236     public void write(Object contentObject, LogLevel logLevel) {
237         write(null, contentObject, logLevel);
238     }
239 
240     /**
241      * Writes any string to the log output.
242      * @param s The string to write
243      * @param @param logLevel LogLevel as specified above.
244      */
245     public void write(String s, LogLevel logLevel) {
246 
247         long elapsedMillis = System.currentTimeMillis() - millisAtStart;
248 
249 
250         if (logLevel == LogLevel.ALL | logLevel == LogLevel.TO_FILE) {
251 
252             printWriter.println("====== " + elapsedMillis +
253                                 " ms after start");
254 
255             printWriter.println(s);
256             printWriter.println("======");
257             printWriter.println();
258 
```

```
259         }
260
261
262         if (logLevel == LogLevel.ALL |
263             logLevel == LogLevel.TO_SCREEN) {
264
265             System.out.println("====== " + elapsedMillis +
266                                " ms after start");
267
268             System.out.println(s);
269             System.out.println("======");
270             System.out.println();
271         }
272
273     }
274
275     /**
276      * Calls write(s, LogLevel.TO_FILE).
277      * @param s The String to write.
278      */
279     public void write(String s) {
280         write(s, LogLevel.TO_FILE);
281     }
282
283
284     /**
285      * Writes a set of strings in a CSV-file. The strings will be
286      * separated by a comma and the last string will be endend with a
287      * semicolon.
288      * @param s A list of string, supposed to be a line in a csv file.
289      */
290     public void writeCSVLine(String... values) {
291
292         StringBuilder sb = new StringBuilder();
293
294         for (int s = 0; s < values.length; s++) {
295
296             sb.append(values[s]);
297             if (s < values.length - 1) {
298                 sb.append(",");
299             }
300             else {
301                 sb.append(";");
302             }
303         }
304
305         printWriter.println(sb.toString());
306
307     }
308
309     /**
310      * Writes the table header for the CSV log file:
311      * 1. Maximum minimum safety score (among all pedestrians),
312      * 2. Solution method (exhaustive/heuristic/intuitively),
313      * 3. Amount of pedestrians,
314      * 4. Number of iterations (within the solution method),
315      * 5. Maximum distance with critical degree of fatigue (among the
316      *    pedestrians),
317      * 6. Maximum number of times, how ofte assitance for sitting down
318      *    or standing up could not be used by a pedestrian,
```

```
319      * 7. Maximum duration to accomplish the course,
320      * 8. Maximum degree of fatigue at the end of the course.
321      */
322     public void writeCSVHeader() {
323
324         writeCSVLine("Safety-Score",
325                      "Verfahren",
326                      "Anz. d. Passanten",
327                      "Anzahl der Expansionen",
328                      "Max. Distanz mit f = 1",
329                      "Max. Anz. d. Assistenz-Nichtnutzbarkeit",
330                      "Max. Dauer",
331                      "Max. Ermuedungsgrad am Ende");
332     }
333
334     /**
335      * Closes the log file and the corresponding data streams. If
336      * specified it sets the singelton logger instance or csv log file
337      * instance to null, such that "getInstance()" resp.
338      * "getCSVInstance" would return a new instance, hence create a new
339      * file).
340      * @param theInstanceToNull If true, the singleton logger instance
341      *                          is set to null.
342      * @param theCSVInstanceToNull If true, the log file instance is set
343      *                             to null.
344      */
345     public void close(boolean theInstanceToNull,
346                       boolean theCSVInstanceToNull) {
347
348         calendar = new GregorianCalendar();
349
350         System.out.println();
351         System.out.println("Logger closing " +
352                            DateFormat.getDateTimeInstance(
353                                       DateFormat.MEDIUM,
354                                       DateFormat.MEDIUM,
355                                       Locale.GERMAN).
356                                           format(calendar.getTime()));
357
358         printWriter.println();
359         printWriter.println("End at: " +
360                             DateFormat.getDateTimeInstance(
361                                       DateFormat.MEDIUM,
362                                       DateFormat.MEDIUM,
363                                       Locale.GERMAN).
364                                           format(calendar.getTime()));
365
366         printWriter.close();
367         try {
368             fileWriter.close();
369         } catch (IOException e) {
370             e.printStackTrace();
371         }
372
373         if (theInstanceToNull) {
374             theInstance = null;
375         }
376         if (theCSVInstanceToNull) {
377             theCSVInstance = null;
378         }
```

```
379
380     }
381
382 }
```

NumberFormatter.java

```
1  package util;
2
3  import java.text.DecimalFormat;
4  import java.util.Locale;
5
6  /**
7   * Formats numbers as strings.
8   * @author Marvin Hubl
9   * @since 2014-05-29
10  *
11  */
12 public class NumberFormatter {
13
14      /**
15       * The required formatter
16       */
17      private static DecimalFormat formatter =
18                      (DecimalFormat) DecimalFormat.getInstance(Locale.US);
19
20      /**
21       * Formats a number of type double.
22       * @param number The number to format.
23       * @return The formatted number as string.
24       */
25      public static String formatDouble(double number) {
26
27          String result = "";
28
29          if (number == Double.MAX_VALUE) {
30              result = "Double.MAX_VALUE = ";
31          }
32
33          formatter.applyPattern("0.0#####");
34
35          result += formatter.format(number);
36
37          return result;
38
39      }
40
41 }
```

Literatur

Adamo, Diane E., Bernard J. Martin und Susan H. Brown (2007). „Age-Related Differences in Upper Limb Proprioceptive Acuity“. In: *Perceptual and Motor Skills* 104, S. 1297–1309. DOI: 10.2466/pms.104.4.1297-1309.

Adler, M., H.-J. Koldehoff, V. Meuser, S. Scheuer, H. Müller-Arnecke, A. Windel und T. Bleyer (2010). *Ergonomiekompendium: Anwendung ergonomischer Regeln und Prüfung der Gebrauchstauglichkeit von Produkten.* Hrsg. von BAuA (Bundesanstalt für Arbeitsschutz und Arbeitsmedizin). Deutschland.

Agostinelli, Forst, Stephen McAleer, Alexander Shmakov und Pierre Baldi (2019). „Solving the Rubik's cube with deep reinforcement learning and search“. In: *Nature Machine Intelligence* 1, S. 356–363. DOI: 10.1038/s42256-019-0070-z.

Ajzen, Icek (1991). „The Theory of Planned Behavior“. In: *Organizational Behavior and Human Decision Processes* 50, S. 179–211. DOI: 10.1016/0749-5978(91)90020-T.

Albrechtsen, Eric (2003). „Security vs Safety“. In: *Techn Rep. of the Dept. of Industrial Economics and Technology Management, Norwegian University of Science and Technology.*

Alchian, Armen A. (1953). „The Meaning of Utility Measurement“. In: *The American Economic Review* 43.1, S. 26–50.

Andrews, A. Williams, Michael W. Thomas und Richard W. Bohannon (1996). „Normative Values for Isometric Muscle Force Measurements Obtained With Hand-held Dynamometers“. In: *Physical Therapy* 76.3, S. 248–259. DOI: 10.1093/ptj/76.3.248.

Anía, Basilio J., Vera J. Suman, Virgil F. Fairbanks, Diana M. Rademacher und L. Joseph Melton (1997). „Incidence of Anemia in Older People: An Epidemionologic Study in a Well Defined Population“. In: *Journal of the American Geriatrics Society* 45, S. 825–831. DOI: 10.1111/j.1532-5415.1997.tb01509.x.

Atzori, Luigi, Antonio Iera und Giacomo Morabito (2010). „The Internet of Things: A survey“. In: *Computer Networks* 54.15, S. 2787–2805. DOI: 10.1016/j.comnet.2010.05.010.

Barin, K. (1989). „Evaluation of a Generalized Model of Human Postural Dynamics and Control in the Sagittal Plane“. In: *Biological Cybernetics* 61, S. 37–50. DOI: 10.1007/BF00204758.

Beckermann, Ansgar (1977). *Gründe und Ursachen: Zum vermeintlich grundsätzlichen Unterschied zwischen mentalen Handlungserklärungen und wissenschaftlich-kausalen Erklärungen*. Kronberg: Scriptor-Verlag.

Bench-Capon, T.J.M. und Paul E. Dunne (2007). „Argumentation in artificial intelligence“. In: *Artificial Intelligence* 171.10-15, S. 619–641. DOI: 10.1016/j.artint.2007.05.001.

Benocci, Marco, Elisabetta Farella und Luca Benini (2011). „A context-aware smart seat“. In: *2011 4th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI)*. Savelletri di Fasano, Italy, S. 104–109. DOI: 10.1109/IWASI.2011.6004697.

Berardelli, A., M. Hallet, J. C. Rothwell, R. Agostino, M. Manfredi, P. D. Thompson und C. D. Marsden (1996). „Single-joint rapid arm movements in normal subjects and in patients with motor disorders“. In: *Brain* 119, S. 661–674. DOI: 10.1093/brain/119.2.661.

Bernardini, Sara, Fabio Fagnani und David E. Smith (2018). „Extracting mutual exclusion invariants from lifted temporal planning domains“. In: *Artificial Intelligence* 258, S. 1–65. DOI: 10.1016/j.artint.2018.01.004.

Bohannon, Richard W. (1997). „Comfortable and maximum walking speed of adults aged 20-79 years: reference values and determinants“. In: *Age and Ageing* 26, S. 15–19. DOI: 10.1093/ageing/26.1.15.

Borkan, Gary A., David E. Hults, Stephen G. Gerzof, Alan H. Robbins und Cynthia K. Silbert (1983). „Age Changes in Body Composition Revealed by Computer Tomography“. In: *Journal of Gerontology* 38.6, S. 673–677. DOI: 10.1093/geronj/38.6.673.

Boutilier, Craig, Yoav Shoham und Michael P. Wellman (1997). „Economic principles of multi-agent systems“. In: *Artificial Intelligence* 94, S. 1–6.

Brandt, Tobias, Wolfgang Ketter, Lutz M. Kolbe, Dirk Neumann und Richard T. Watson (2016). „Call for Papers: Issue 3/2018: Smart Cities and Digitized Urban Management“. In: *Business & Information Systems Engineering* 58.6, S. 437–438. DOI: 10.1007/s12599-016-0452-2.

Brooks, Susan V. und John A. Faulkner (1994). „Skeletal muscle weakness in old age: underlying mechanisms“. In: *Medicine and science in sports and exercise* 26.4, S. 432–439.

Browne, Cameron, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samotharkis und Simon Colton (2012). „A Suvery on Monte Carlo Tree

Search Methods“. In: *IEEE Transactions on Computational Intelligence and AI in Games* 4.1. DOI: 10.1109/TCIAIG.2012.2186810.

Campbell, M. J., A. J. McComas und F. Petito (1973). „Physiological changes in ageing muscles“. In: *Journal of Neurology, and Psychiatry* 36, S. 174–182. DOI: 10.1136/jnnp.36.2.174.

Canny, John (1986). „A Computational Approach to Edge Detection“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6, S. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

Carnap, Rudolf (1959). *Induktive Logik und Wahrscheinlichkeit*. Hrsg. von Wolfgang Stegmüller. Wien: Springer.

Chen, X-, E. Santos-Neto und M. Ripeanu (2012). „Crowdsourcing for on-street smart parking“. In: *Proc. of the 2nd ACM int. symp. on Design and analysis of intelligent vehicular networks and applications*, S. 1–8. DOI: 10.1145/2386958.2386960.

Coninx, Kristof, Rutger Claes, Stijn Vandael, Niels Leemput, Tom Holvoet und Geert Deconick (2014). „Anticipatory Coordination of Electric Vehicle Allocation to Fast Charging Infrastructure“. In: *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection. PAAMS 2014. Lecture Notes in Computer Science*. Hrsg. von Y. Demezeau, F. Zambonelli, J. M. Corchado und J. Bajo. Bd. 8473. Springer, Cham, S. 74–85. DOI: 10.1007/978-3-319-07551-8_7.

Cottrel, Marie, Madalina Olteanu, Fabrice Rossi, Joseph Rynkiewicz und Nathalie Villa-Vialaneix (2012). „Neural Networks for Complex Data“. In: *Künstliche Intelligenz* 26, S. 373–380. DOI: 10.1007/s13218-012-0207-2.

Cremonesi, Paolo, Antonella Di Rienzo und Franca Garzotto (2015). „Personalized interactive public screens“. In: *Proc. of the 4th Workshop on Interacting with Smart Objects (at ACM IUI 2015)*, S. 10–15.

Criqui, Michael H., Julie O. Denenberg, Robert D. Langer und Arnost Fronek (1997). „The epidemiology of peripheral arterial disease: importance of identifying the population at risk“. In: *Vascular Medicine* 2, S. 221–226. DOI: 10.1177/1358863X9700200310.

Cunha, Marcio und Hugo Fuks (2015). „AmbLEDs: Implicit I/O for AAL Systems“. In: *Proceedings of the 4th Workshop on Interacting with Smart Objects (at ACM IUI 2015)*, S. 6–9.

Darling, W. G., J. D. Cooke und S. H. Brown (1989). „Control of Simple Arm Movements in Elderly Humans“. In: *Neurobiology of Aging* 10, S. 149–157. DOI: 10.1016/0197-4580(89)90024-9.

Debus, E. S., G. Torsello, T. Schmitz-Rixen, T. Hupp, W. Lang, T. Noppeney, A. Oberhuber und R. T. Grundmann (2013). „Manifestation und

Prävention der Arteriosklerose“. In: *Gefässchirurgie* 18, S. 644–651. DOI: 10.1007/s00772-013-1235-4.

Dechter, Rina und Judae Pearl (1985). „Generalized Best-First Search Strategies and the Optimality of A*“. In: *Journal of the Association for Computing Machinery* 32.3, S. 505–536. DOI: 10.1145/3828.3830.

Dekel, Ammon, Yitzhak Simon, Hilal Dar, Ezri Tarazi, Oren Rabinowitz und Yoav Sterman (2005). „Adding Playful Interaction to Public Spaces“. In: *Proc. of the 1st Int. Conf. on Intelligent Technologies for Interactive Entertainment (INTETAIN 2005)*. Berlin, Heidelberg: Springer, S. 225–229. DOI: 10.1007/11590323_24.

Doherty, Timothy J., Anthony A. Vandervoort, Albert W. Taylor und William F. Brown (1993). „Effects of motor unit losses on strength in older men and women“. In: *Journal of Applied Physiology* 74.2, S. 868–874. DOI: 10.1152/jappl.1993.74.2.868.

Duden online (2020). *Intuition*. URL: https://www.duden.de/node/72049/revision/72085 (besucht am 23. 07. 2020).

Egly, Uwe und Leopold Haller (2010). „A SAT Solver for Circuits Based on the Tableau Method“. In: *Künstliche Intelligenz* 24, S. 15–23. DOI: 10.1007/s13218-010-0008-4.

Era, Pertti, Marianna Schroll, Henriette Ytting, Ingrid Gause-Nilsson, Eino Heikkinen und Bertil Steen (1996). „Postural Balance and Its Sensory-Motor Correlates in 75-Year-Old Men and Women: A Cross-National Comparative Study“. In: *Journal of Gerontology* 51A.2, S. M53–M63. DOI: 10.1093/gerona/51A.2.M53.

Espinola-Klein, C. (2011). „Periphere arterielle Verschlusskrankheit“. In: *Internist* 52, S. 549–561. DOI: 10.1007/s00108-011-2814-7.

Evans, Wiliam J. und Wayne W. Campbell (1993). „Sarcopenia and Age-Related Changes in Body Composition and Functional Capacity“. In: *Journal of Nutrition* 123.suppl_2, S. 465–468. DOI: 10.1093/jn/123.suppl_2.465.

Ferber, Jacques (2001). *Multiagentensysteme: Eine Einführung in die Verteilte Künstliche Intelligenz*. Hrsg. von Stefan Kirn. München: Addison-Wesley.

Fernández, Lara, Hayo A. Breinbauer und Paul Hinckley Delano (2015). „Vertigo and dizziness in the elderly“. In: *Frontiers in Neurology* 6.144. DOI: 10.3389/fneur.2015.00144.

Foell, Stefan, Gerd Kortuem, Reza Rawassizadeh, Marcus Handte, Umer Iqbal und Pedro Marrón (2014). „Micro-navigation for Urban Bus Passengers: Using the Internet of Things to Improve the Public Transport Experience“. In: *Proceedings of the First International Conference on IoT in Urban Space (Urb-IoT'149*, S. 1–6. DOI: 10.4108/icst.urb-iot.2014.257373.

Frantz, Roger (2003). „Herbert Simon. Artificial Intelligence as a framework for understanding intuition“. In: *Journal of Economic Psychology* 24, S. 265–277. DOI: 10.1016/S0167-4870(02)00207-6.

Frontera, Walter R., Dongwon Suh, Lisa S. Krivickas, Virginia A. Hughes, Richard Goldstein und Ronenn Roubenoff (2000). „Skeletal muscle fiber quality in older men and women“. In: *American Journal of Physiology-Cell Physiology* 279, S. C611–C618. DOI: 10.1152/ajpcell.2000.279.3.C611.

Galganski, Michele E., Andrew H. Fuglevand und Roger M. Enoka (1993). „Reduced Control of Motor Output in a Human Hand Muscle of Elderly Subjects During Submaximal Contractions“. In: *Journal of Neuropsychology* 69, S. 6. DOI: 10.1152/jn.1993.69.6.2108.

García-Magariño, Iván, Guillermo Palacios-Navarro, Raquel Lacuesta und Jaime Lloret (2018). „ABSCEV: An agent-based simulation framework about smart transportation for reducing waiting times in charging electric vehicles“. In: *Computer Networks* 138, S. 119–135. DOI: 10.1016/j.comnet.2018.03.014.

Geng, Y. und C. G. Cassandras (2013). „New ‘smart parking’ system based on resource allocation and reservations“. In: *IEEE Transactions on Intelligent Transportation Systems* 14.3, S. 1129–1139. DOI: 10.1109/TITS.2013.2252428.

Giesl, Jürgen (2010). „Current Trends in Automated Deduction“. In: *Künstliche Intelligenz* 24, S. 11–13. DOI: 10.1007/s13218-010-0011-9.

Gosselin, Luc E., Christopher Adams, Tiffani A. Cotter, Richard J. McCormick und D. Paul Thomas (1998). „Effect of exercise training on passive stiffness in locomotor skeletal muscle: role of extracellular matrix“. In: *Journal of Applied Physiology* 85.3, S. 1011–1016. DOI: 10.1152/jappl.1998.85.3.1011.

Gosselin, Luc E., Daniel A. Martinez, Arthur C. Vailas und Gary C. Sieck (1994). „Passive length-force properties of senescent diaphragm: relationship with collagen characteristic“. In: *Journal of Applied Physiology* 76.6, S. 2680–2685. DOI: 10.1152/jappl.1994.76.6.2680.

Gusrialdi, Azwirman, Zhihua Qu und Marwan A. Simaan (2017). „Distributed Scheduling and Cooperative Control for Charging of Electric Vehicles at Highway Service Stations“. In: *IEEE Transactions on Intelligent Transportation Systems* 18.10, S. 2713–2727. DOI: 10.1109/TITS.2017.2661958.

Hammi, Badis, Rida Khatoun, Sherali Zeadally, Achraf Fayad und Lyes Khoukhi (2018). „IoT technologies for smart cities“. In: *IET Networks* 7.1, S. 1–13. DOI: 10.1049/iet-net.2017.0163.

Harada, N. D., V. Chiu und A. L. Stewart (1999). „Mobility-Related Function in Older Adults: Assessment With a 6-Minute Walk Test“. In: *Archives of*

Physical Medicine and Rehabilitation 80, S. 837–841. DOI: 10.1016/S0003-9993(99)90236-8.

Haralick, Robert M. und Linda G. Shapiro (1979). „The Consistent Labeling Problem: Part I“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence1* PAMI-1.2, S. 173–184. DOI: 10.1109/TPAMI.1979.4766903.

— (1980). „The Consistent Labeling Problem: Part II“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-2.3, S. 193–203. DOI: 10.1109/TPAMI.1980.4767007.

Hassabis, Demis, Dharshan Kumaran, Christopher Summerfield und Matthew Botvinick (2017). „Neuroscience-Inspired Artificial Intelligence“. In: *Neuron* 95, S. 245–258. DOI: 10.1016/j.neuron.2017.06.011.

Hastie, Trevor, Robert Tibshirani und Jerome Friedman (2013). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2. Aufl. Springer.

Hayflick, Leonard (1965). „The Limited In Vitro Lifetime of Human Diploid Cell Strains“. In: *Experimental Cell Research* 37, S. 614–636. DOI: 10.1016/0014-4827(65)90211-9.

— (1985). „Theories of Biological Aging“. In: *Experimental Gerontology* 20, S. 145–159. DOI: 10.1016/0531-5565(85)90032-4.

Helmert, Malter, Patrick Haslum, Jörg Hoffmann und Raz Nissim (2014). „Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces“. In: *Journal of the ACM2* 61.3, Article 16. DOI: 10.1145/2559951.

Hillenbrand, Martin (2012). *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik / Elektronik Architekturen von Fahrzeugen*. Karlsruhe: KIT Scientific Publishing.

Hubl, Marvin (2018). „Adaption rule for simultaneous use of smart urban objects from a fairness perspective“. In: *Proc. of the 20th Int. Conf. on Business Informatics (CBI 2018)*, S. 89–98. DOI: 10.1109/CBI.2018.00019.

— (2019). „An Adaptive Park Bench System to Enhance Availability of Appropriate Seats for the Elderly: A Safety Engineering Approach for Smart City“. In: *Proceedings of the 21st IEEE Conference on Business Informatics (CBI 2019)*, S. 373–382. DOI: 10.1109/CBI.2019.00049.

Hubl, Marvin, Philipp Skowron und Michael Aleithe (2018). „Towards a Supportive City with Smart Urban Objects in the Internet of Things: The Case of Adaptive Park Bench and Adaptive Light“. In: *Position Papers of the 2018 Federated Conf. on Computer Science and Information Systems*. Hrsg. von M. Ganzha, L. Maciaszek und M. Paprzycki, S. 51–58. DOI: 10.15439/2018F118.

Huldtgren, Alina, Christina Katsimerou, Andre Kuijsters, Judith A. Redi und Ingrid E. J. Heynderickx (2015). „Design Considerations for Adaptive Lighting to Improve Senior's Mood". In: *Proceedings of the 13th International Conference on Smart Homes and Health Telematics (ICOST 2015)*. Springer (LNCS), S. 15–26. DOI: 10.1007/978-3-319-19312-0_2.

Hurley, Michael V., Joanne Rees und Di J. Newham (1998). „Quadriceps function, proprioceptive acuity and functional performance in healthy young, middle-aged and elderly subjects". In: *Age and Ageing* 27, S. 55–62. DOI: 10.1093/ageing/27.1.55.

Hytönen, Malta, Ilmari Pyykkö, Heikki Aalto und Jukka Starck (1993). „Postural Control and Age". In: *Acta Oto-Laryngologica* 113.2, S. 119–122. DOI: 10.3109/00016489309135778.

Kant, Immanuel (1966). *Kritik der reinen Vernunft*. Stuttgart: Reclam.

Kerrigan, D. Casey, Mary K. Todd, Ugo Della Croce, Lewis A. Lipsitz und James J. Collins (1998). „Biomechanical Gait Alterations Independent of Speed in the Healthy Elderly: Evidence for Specific Limiting Impairments". In: *Archives of Physical Medicine and Rehabilitation* 79, S. 317–322. DOI: 10.1016/S0003-9993(98)90013-2.

Kinch, Sofie, Erik Grönvall, Marianna Graves Petersen und Majken Kirkegaard Rasmussen (2014). „Encounters on a Shape-changing Bench: exploring atmospheres and social behaviour in situ". In: *Proc. of the 8th Int. Conf. on Tangible, Embedded and Embodied Interaction (TEI'14)*, S. 233–240. DOI: 10.1145/2540930.2540947.

Kirn, Stefan (1996). *Gestaltung von Multiagenten-Systemen: Ein organisationszentrierter Ansatz*. Westfälische Wilhems-Universität Münster (Habilitationsschrift).

— (2002). „Kooperierende intelligente Softwareagenten". In: *Wirtschaftsinformatik* 44, S. 53–63. DOI: 10.1007/BF03251465.

Kothiyal, K. und S. Tettey (2001). „Anthropometry for Design for the Elderly". In: *International Journal of Occupational Safety and Ergonomics* 7.1, S. 15–34. DOI: 10.1080/10803548.2001.11076474.

Kuilman, Thomas, Chrysiis Michaloglou, Wolter J. Mooi und Daniel S. Peeper (2010). „The essence of senescence". In: *Genes & Development* 24, S. 2463–2470. DOI: 10.1101/gad.1971610.

Kumar, Priyan Malarvizhi, Ushadevi Gandhi, R. Varatharajan, Gunasekaran Manogaran, Jidhesh R. und Thanjai Vadivel (2017). „Intelligent face recognition and navigation system using neural learning for smart security in Internet of Things". In: *Cluster Computing*. DOI: 10.1007/s10586-017-1323-4.

Laborie, Philippe (2003). „Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results“. In: *Artificial Intelligence* 143, S. 151–188. DOI: 10.1016/S0004-3702(02)00362-4.

LeCun, Yann, Yoshua Bengio und Geoffrey Hinton (2015). „Deep learning“. In: *Nature* 521, S. 436–444. DOI: 10.1038/nature14539.

Leveson, Nancy G. (2011). *Engineering a Safer World: Systems Thinking Applied to Safety*. Massachusetts: MIT Press.

Lexell, Jan, Charles C. Taylor und Michael Sjöström (1988). „What is the cause of the ageing atrophy? Total number, size and proportion of different fiber types studied in whole vastus lateralis muscle from 15- to 83-year-old men“. In: *Journal of the Neurological Sciences* 84, S. 275–294. DOI: 10.1016/0022-510x(88)90132-3.

Li, Shancang, Li Da Xu und Shanshan Zhao (2015). „The internet of things: a survey“. In: *Information Systems Frontiers* 17, S. 243–259. DOI: 10.1007/s10796-014-9492-7.

López, Tomás Sánches, Damith Chinthana Ranasinghe, Bela Patkai und Duncan McFarlane (2011). „Taxonomy, technology and applications of smart objects“. In: *Information Systems Frontiers* 13, S. 281–300. DOI: 10.1007/s10796-009-9218-4.

Lord, Stephen R., Russel D. Clark und Ian W. Webster (1991). „Postural Stability and Associated Physiological Factors in a Population of Aged Persons“. In: *Journal of Gerontology: Medical Sciences* 46.3, S. M69–76. DOI: 10.1093/geronj/46.3.m69.

Lorenzen, Paul (1994). „Konstruktivismus“. In: *Journal for General Philosophy of Science / Zeitschrift für allgemeine Wissenschaftstheorie* 25.1, S. 125–133.

— (2000). *Lehrbuch der konstruktiven Wissenschaftstheorie*. Stuttgart: J.B. Metzler. DOI: 10.1007/978-3-476-02758-0.

Lüder, Anna und Irina Böckelmann (2011). „Visuelle Leistungen unter dem Aspekt Alter“. In: *Zentralblatt für Arbeitsmedizin, Arbeitsschutz und Ergonomie* 61.10, S. 328–336. DOI: 10.1007/BF03345013.

Luger, George F. (2009). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 6th Editio. Boston: Pearson Education.

Lynch, N. A., E. J. Metter, R. S. Lindle, J. L. Fozard, J. D. Tobin, T. A. Roy, J. L. Fleg und B. G. Hurley (1999). „Muscle quality. I. Age-associated differences between arm and leg muscle groups“. In: *Journal of Applied Physiology* 86.1, S. 188–194. DOI: 10.1152/jappl.1999.86.1.188.

Mattern, Friedemann und Christian Flörkemeier (2010). „Vom Internet der Computer zum Internet der Dinge“. In: *Informatik-Spektrum* 33.2, S. 107–121. DOI: 10.1007/s00287-010-0417-7.

McCulloch, Warren S. und Walter Pitts (1943). „A Logical Calculus of the Ideas Immament in Nervous Activity“. In: *Bulletin of Mathematical Biophysics* 5, S. 115–133. DOI: 10.1007/BF02478259.

Menz, Nadja, Petra Hoeppner, Jens Tiermann und Frank Koußen (2015). „Safety und Security aus dem Blickwinkel der öffentlichen IT“. In: *White Paper of the Fraunhofer-Institut für Offene Kommunikationssysteme FOKUS*.

Metivier, Fabien, Sylvain J. Marchais, Alain P. Guerin, Bruno Pannier und Gérard M. London (2000). „Pathophysiology of anaemia: focus on the heart and blood vessels“. In: *Nephrology Dialysis Transplantation* 15.3, S. 14–18. DOI: 10.1093/oxfordjournals.ndt.a027970.

Minne, H. W., M. Pfeifer, B. Begerow und Pollhähne W (2002). „Osteoporose“. In: *Der Orthopäde* 31.7, S. 681–699. DOI: 10.1007/s00132-002-0348-3.

Mitchell, Tom M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

Müller, Jörg, Florian Arlt, Daniel Michelis und Albrecht Schmidt (2010). „Requirements and design space for interactive public displays“. In: *Proceedings of the 18th ACM international conference on Multimedia (MM'10)*, S. 1285–1294. DOI: 10.1145/1873951.1874203.

Myerson, Roger B. (1979). „Incentive Compatibility and the Bargaining Problem“. In: *Econometrica* 47.1, S. 61–74. DOI: 10.2307/1912346.

— (1984). „Two-Person Bargaining Problems with Incomplete Information“. In: *Econometrica* 52.2, S. 461–488. DOI: 10.2307/1911499.

Neirotti, Paolo, Alberto De Marco, Anna Corinna Cagliono, Giulio Mangano und Francesco Scorrano (2014). „Current trends in Smart City initiatives: Some stylised facts“. In: *Cities* 38, S. 25–36. DOI: 10.1016/j.cities.2013.12.010.

Nonaka, Hisako, Katsumi Mita, Makoto Watakabe, Kumi Akataki, Nobuharu Suzuki, Testuo Okuwa und Kyonosuke Yabe (2002). „Age-related changes in the interactive mobility of the hip and knee joints: a geometrical analysis“. In: *Gait and Posture* 15, S. 236–243. DOI: 10.1016/S0966-6362(01)00191-6.

OECD, Hrsg. (2001). *Ageing and Transport: Mobility Needs and Safety Issues*. Paris: OCED Publications. DOI: 10.1787/9789264195851-en.

Okada, Miyo, Atsuro Ueki, Niclas Jonasson, Masato Yamanouchi, Cristian Norlin, Hideki Sunahara, Joakim Formo, Mikael Anneroth und Masa Inakage (2016). „Autonomous Cooperation of Social Things: Designing a System for things with Unique Personalities in IoT“. In: *6th Internatio-*

nal Conference on the Internet of Things (IoT'16). Stuttgart, Germany, S. 35–42. DOI: 10.1145/2991561.2991574.

Olovnikov, A. M. (1973). „A Theory of Marginotomy: The Inclompete Copying of Template Margin in Enzymic Synthesis of Polynucleotides and Biological Significance of the Phenomenon". In: *Journal of Theoretical Biology* 41, S. 181–190. DOI: 10.1016/0022-5193(73)90198-7.

Oxford Dictionaries online (2020a). *safety*. URL: https://en.oxforddictionaries.com/definition/safety (besucht am 23. 07. 2020).

— (2020b). *security*. URL: https://en.oxforddictionaries.com/definition/security (besucht am 23. 07. 2020).

Padgham, Lin und Patrick Lambrix (2005). „Formalisations of Capabilities for BDI-Agents". In: *Autonomous Agents and Multi-Agent Systems* 10, S. 249–271. DOI: 10.1007/s10458-004-4345-2.

Peterka, R. J. und F. O. Black (1990). „Age-related changes in human posture control: motor coordination tests". In: *Journal of Vestibular Research* 1, S. 87–96.

Pfohl, Hans-Christian (2004). *Logistiksysteme*. Berlin, Heidelberg: Springer.

Piacquadio, Paulo Giovanni (2017). „A Fairness Justification of Utilitarianism". In: *Econometrica* 85.4, S. 1261–1276. DOI: 10.3982/ECTA14151.

Pisciottano, M. V. C., S. S. Pinto, V. L. Szejnfeld und C. H. M. Castro (2014). „The relationship between lean mass, muscle strength and physical ability in independent healthy elderly women from the community". In: *The Journal of Nutrition, Health & Aging* 18.5, S. 554–558. DOI: 10.1007/s12603-013-0414-z.

Ploenes, C. (2019). „So wird die Schaufensterkrankheit abgeklärt". In: *MMW Fortschritte der Medizin* 5.161, S. 46–48. DOI: 10.1007/s15006-019-0274-5.

Poulsen, Esben Skouboe, Ann Morrison, Hans Jorgen Andersen und Ole B. Jensen (2013). „Responsive Lighting: 'The city becomes alives'". In: *Proc. of the 15th Int. Conf. on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'13)*, S. 217–226. DOI: 10.1145/2493190.2493218.

Rawls, John (1999). *A Theory of Justice*. Revised Edition. Cambridge, Massachusetts: The Belknap Press of Harvard University Press.

Ray, Debraj und Kaoru Ueda (1996). „Egalitarianism and Incentives". In: *Journal of Economic Theory* 71.2, S. 324–348. DOI: 10.1006/jeth.1996.0124.

Rayes, Ammar und Samer Salam (2017). *Internet of Things–From Hype to Reality: The Road to Digitization*. Cham: Springer. DOI: 10.1007/978-3-319-99516-8.

Rice, C. L., D. A. Cunningham, D. H. Paterson und M. S. Lefcoe (1989). „Arm and leg composition determined by computed tomography in young and elderly men“. In: *Clinical Physiology* 9, S. 207–220. DOI: 10.1111/j.1475-097X.1989.tb00973.x.

Rinkenauer, Gerhard (2008). „Motorische Leistungsfähigkeit im Alter“. In: *Leistungsfähigkeit und Mobilität im Alter*. Hrsg. von Bernhard Schlag. TÜV Media GmbH TÜV Rheinland Group, S. 143–180.

Ronsky, J. L., B. M. Nigg und V. Fisher (1995). „Correlation between physical activity and the gait characteristics and ankle joint flexibility of the elderly“. In: *Clinical Biomechanics* 10.1, S. 41–49. DOI: 10.1016/0268-0033(95)90436-D.

Roos, Martin R., Charles L. Rice, Denise M. Connelly und Anthony A. Vandervoort (1999). „Quadriceps Muscle Strength, Contractile Properties, and Motor Unit Firing Rates in Young and Old Men“. In: *Muscle Nerve* 22, S. 1094–1103. DOI: 10.1002/(SICI)1097-4598(199908)22:8<1094::AID-MUS14>3.0.CO;2-G.

Ropohl, Günter (2009). *Allgemeine Technologie: Eine Systemtheorie der Technik*. 3. Auflage. KIT Scientific Publishing.

Rosenberg, Irwin H. (1997). „Sarcopenia: Origins and Clinical Relevance“. In: *The Journal of Nutrition* 127.5, 990S–991S. DOI: 10.1093/jn/127.5.990S.

Russell, Stuart und Peter Norvig (2003). *Artificial intelligence: A modern approach*. 2. Aufl. New Jersey: Prentice Hall.

Samuel, A. L. (1959). „Some Studies in Machine Learning Using the Game of Checkers“. In: *IBM Journal*, S. 211–229. DOI: 10.1147/rd.33.0210.

Saß, A. C., S. Wurm und C. Scheidt-Nave (2010). „Alter und Gesundheit: Eine Bestandsaufnahme aus Sicht der Gesundheitsberichterstattung“. In: *Bundesgesundheitsblatt* 53, S. 404–416. DOI: 10.1007/s00103-010-1049-4.

Shepard, Roy J. (1999). „Age and Physical Work Capacity“. In: *Experimental Aging Research* 25.4, S. 331–343. DOI: 10.1080/036107399243788.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Drieesche, Julian Schrittwieser, Ionannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel und Demis Hassabis (2016). „Mastering the game of Go with deep neural networks and tree search“. In: *Nature* 529, S. 484–503. DOI: 10.1038/nature16961.

Skowron, Philipp, Michael Aleithe, Susanne Wallrafen, Marvin Hubl, Julian Fietkau und Bogdan Franczyk (2019). „Smart Urban Design Space“. In:

Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS'19). IEEE, S. 493–496. DOI: 10.15439/2019F80.

Sloane, Philip D., Robert W. Baloh und Vicente Honrubia (1989). „The Vestibular System in the Elderly: Clinical Implications". In: *American Journal of Otolaryngology* 10, S. 422–429. DOI: 10.1016/0196-0709(89)90038-0.

Sloane, Philip D., Remy R. Coeytaux, Rainer S. Beck und John Dallara (2001). „Dizziness: State of Science". In: *Annals of internal medicine* 134, S. 823–832. DOI: 10.7326/0003-4819-134-9_part_2-200105011-00005.

Smirnov, Alexander und Nikolay Shilov (2010). „AI-based Approaches to Solving a Dynamic Logistics Problem". In: *Künstliche Intelligenz* 24, S. 143–147. DOI: 10.1007/s13218-010-0028-0.

Stachowiak, Herbert (1973). *Allgemeine Modelltheorie*. Wien: Springer.

Stegmüller, Wolfgang (1958). „Carnaps Auffassung der induktiven Logik". In: *Induktive Logik und Wahrscheinlichkeit*. Hrsg. von Wolfgang Stegmüller. Wien: Springer. Kap. Einleitung, S. 1–11.

Stevens, Joseph C. und Kenneth K. Choo (1996). „Spatial Acuity of the Body Surface over the Life Span". In: *Somatosensory & Motor Research* 13.2, S. 153–166. DOI: 10.3109/08990229609051403.

Swash, M. und Kathleen P. Fox (1972). „The Effect of Age on Human Skeletal Muscle: Studies of the Morphology and Innervation of Muscle Spindles". In: *Journal of the neurological Sciences* 16, S. 417–432. DOI: 10.1016/0022-510X(72)90048-2.

Thomas, D. P., R. J. McCormick, S. D. Zimmermann, R. K. Vadlamudi und L. E. Gosselin (1992). „Aging- and training-induced alterations in collagen characteristics of rat left ventricle and papillary muscle". In: *American Journal of Physiology-Heart and Circulatory Physiology* 263.3, H778–H783. DOI: 10.1152/ajpheart.1992.263.3.H778.

Thomson, William (1983). „Problems of Fair Division and the Egalitarian Solution". In: *Journal of Economic Theory* 31, S. 211–226. DOI: 10.1016/0022-0531(83)90074-1.

— (2011). „Fair Allocation Rules". In: *Handbook of Social Choice and Welfare*. Hrsg. von Kenneth J. Arrow, Amartya Sen und Kotaro Suzumura. Bd. 2. Elsevier BV. Kap. 21, S. 393–506. DOI: 10.1016/S0169-7218(10)00021-3.

Timm, Ingo J. und Andreas D. Lattner (2010). „Künstliche Intelligenz in der Logistik". In: *Künstliche Intelligenz* 24, S. 99–103. DOI: 10.1007/s13218-010-0022-6.

Tokic, Michel (2013). „Reinforcement Learning: Psychologische und neurobiologische Aspekte". In: *Künstliche Intelligenz* 27, S. 213–219. DOI: 10.1007/s13218-013-0261-4.

Traunmueller, Martin und Ava Fatah gen. Schieck (2013). „Introducing the Space Recommender System: How crowd-sourced voting can enrich Urban Exploration in the digital Era“. In: *Proceedings of the 6th International Conference on Communities and Technologies (C&T'13)*. Munich, S. 149–156. DOI: 10.1145/2482991.2482995.

U.S. Department of Transportation (1997). *Improving Transportation for a Maturing Society*. Washington, DC: Office of the Assistant Secretary for Transportation Policy.

Valko, Yulia, Richard F. Lewis, Adrian J. Priesol und Daniel M. Merfeld (2012). „Vestibular Labyrinth Contributions to Human Whole-Body Motion Discrimination“. In: *The Journal of Neuroscience* 32.39, S. 13537–13542. DOI: 10.1523/JNEUROSCI.2157-12.2012.

van Eersel, Gerdien G., Gabriela V. Koppenol-Gonzalez und Julian Reiss (2019). „Extrapolation of Experimental Results through Analogical Reasoning from Latent Classes“. In: *Philosophy of Science* 86, S. 219–235. DOI: 10.1086/701956.

Vandervoort, Anthony A., Bert M. Chesworth, David A. Cunningham, Don H. Paterson, Peter A- Rechnitzer und John J. Koval (1992). „Age and Sex Effects on Mobility of the Human Ankle“. In: *Journal of Gerontology* 47.1, S. M17–21. DOI: 10.1093/geronj/47.1.M17.

Varian, Hal R. (1974). „Equity, Envy, and Efficiency“. In: *Journal of Economic Theory* 9, S. 63–91. DOI: 10.1016/0022-0531(74)90075-1.

VDI, Hrsg. (2000). *VDI-Richtlinie 3780 (Technikbewertung: Begriffe und Grundlagen)*. Berlin: Beuth Verlag.

Vogel, Daniel und Ravin Balakrishnan (2004). „Interactive Public Ambient Displays: Transitioning from Implicit to Explicit, Public to Personal, Interaction with Multiple Users“. In: *Proceedings of the 17th annual ACM symposium in User interface software and technology*, S. 137–146. DOI: 10.1145/1029632.1029656.

Wachtel, Ellen, Alice Maroudas und Rosa Schneiderman (1995). „Age-related changes in collagen packing of human articular cartilage“. In: *Biochimica et Biophysica Acta* 1243, S. 239–243. DOI: 10.1016/0304-4165(94)00134-J.

Weinert, Briant T. und Poala S. Timiras (2003). „Invited Review: Theories of aging“. In: *Journal of Applied Physiology* 95, S. 1706–1716. DOI: 10.1152/japplphysiol.00288.2003.

Weiser, Mark (1991). „The Computer for the 21st Century“. In: *Scientific American* 265.3, S. 94–105. DOI: 10.1038/scientificamerican0991-94.

Winter, David A., Aftab E. Patla, James S. Frank und Sharon E. Walt (1990). „Biomechanical Walking Pattern Changes in the Fit and Healthy Elderly“. In: *Physical Therapy* 70.6, S. 340–347. DOI: 10.1093/ptj/70.6.340.

Wolpert, David H. und William G. Macready (1996). *No Free Lunch Theorems for Search*. Techn. Ber. SFI-TR-95-02-010. Santa Fe: The Santa Fe Institute.

— (1997). „No Free Lunch Theorems for Optimizations“. In: *IEEE Transactions on Evolutionary Computation* 1.1, S. 67–82. DOI: 10.1109/4235.585893.

Wooldridge, Michael (2009). *An Introduction to MultiAgent Systems*. 2. Aufl. Chichester, West Sussex: Wiley.

Wu, Fang-Jing, Yu-Fen Kao und Yu-Chee Tseng (Aug. 2011). „From wireless sensor networks towards cyber physical systems“. In: *Pervasive and Mobile Computing* 7.4, S. 397–413. DOI: 10.1016/j.pmcj.2011.03.003.

Wu, Ge (1998). „The Relation Between Age-Related Changes in Neuromusculoskeletal System and Dynamic Postural Response to Balance Disturbance“. In: *Journal of Gerontology* 53A.4, S. M320–M326. DOI: 10.1093/gerona/53a.4.m320.

Xu, Li Da, Wu He und Shancang Li (2014). „Internet of Things in Industries: A Survey“. In: *IEEE Transactions on Industrial Informatics* 10.4, S. 2233–2243. DOI: 10.1109/TII.2014.2300753.

Yan, G., W. Yang, D. B. Rawant und S. Olariu (2011). „SmartParking: A secure and intelligent parking system“. In: *IEEE Intelligent Transportation Systems Magazine* 3.1, S. 18–30. DOI: 10.1109/MITS.2011.940473.

Yang, J., J. Portilla und T. Riesgo (2012). „Smart parking service based on Wireless Sensor Networks“. In: *38th Annual Conf. on IEEE Industrial Electronics Society (IECON 2012)*, S. 6029–6034. DOI: 10.1109/IECON.2012.6389096.

Yengin, Duygu (2012). „Characterizing Welfare-egalitarian Mechanisms with Solidarity When Valuations are Private Information“. In: *The B.E. Journal of Theoretical Economics* 12.1. DOI: 10.1515/1935-1704.1789.

Zacher, J. und A. Gursche (2001). „Diagnostik der Arthrose“. In: *Der Orthopäde* 30.11, S. 841–847. DOI: 10.1007/s001320170020.

Zanella, Andrea, Nicola Bui, Angelo Castellani, Lorenzo Vanelista und Michele Zorzi (2014). „Internet of Things for Smart Cities“. In: *IEEE Internet of Things Journal* 1.1, S. 22–32. DOI: 10.1109/JIOT.2014.2306328.

Zimmerman, Scott D., Richard J. McCormick, Ratna K. Vadlamudi und D. Paul Thomas (1993). „Age and training alter collagen characteristics in fast- and slow-twitch rat limb muscle“. In: *Journal of Applied Physiology* 75.4, S. 1670–1674. DOI: 10.1152/jappl.1993.75.4.1670.

Zimmermann, Hans-Georg, Christoph Tietz, Ralph Grothmann und Thomas Runkler (2012). „Recurrent Neural Networks for Industrial Procurement

Decisions“. In: *Künstliche Intelligenz* 26, S. 403–406. DOI: 10.1007/s13218-012-0194-3.