4. Agentenorientierte Erweiterungen der Objektbasierten Simulation

In Kapitel 3 wurde das Basismodell der Diskreten-Ereignis-Simulation um drei nicht-agentenorientierte Erweiterungen angereichert, die unter dem Begriff der *Objektbasierten Simulation* subsummiert wurden.

Ein Nachteil in der Objektbasierten Simulation ist, dass es keine Unterscheidung zwischen aktiven und passiven Entitäten gibt. Ein weiterer Nachteil ist, dass die Simulation mehrerer aktiver Entitäten, die nebenläufig zueinander Aktionen ausführen, mit Hilfe der Objektbasierten Simulation schwierig ist und, dass Nachrichtenaustausch zwischen Objekten nicht direkt unterstützt wird. Solange es sich nicht nur um Entitäten mit reaktivem Verhalten handelt, die auf stattfindende Ereignisse reagieren, sondern um Entitäten mit eigenem Kontrollfluss, die von sich aus Aktionen durchführen (proaktives Verhalten), müsste zu jedem Zeitpunkt, an dem eine solche Aktion möglich ist, ein Pseudoereignis eingefügt werden, das verhindert, dass die Zeit über den Zeitpunkt der Aktion hinaus weiterläuft und somit die Aktion unmöglich macht. Dies wäre eine zugleich uneffiziente und unnatürliche Modellierungsweise.

Im Folgenden werden zwei agentenorientierte Erweiterungen der Objektbasierten Simulation vorgestellt, die diese Nachteile ausgleichen sollen. Sie werden unter dem Begriff der Agentenbasierten Simulation zusammengefasst. In den agentenorientierten Erweiterungen wird zwischen Agenten (aktiven Entitäten) und Objekten (passiven Entitäten) unterschieden.

Dabei wird in Abschnitt 4.1 als eine mögliche agentenorientierte Erweiterung der Austausch von Nachrichten vorgestellt. Abschnitt 4.2 enthält eine weitere agentenorientierte Erweiterung, die Trennung zwischen externem und internem Zustand eines Agenten, verbunden mit der Einführung eines Umgebungssimulators. In Abschnitt 4.3 wird dann das Prinzip der auf der Objektbasierten Simulation aufbauenden *Agentenbasierten Simulation* vorgestellt, das diese beiden Erweiterungen in sich vereinigt.

4.1 Nachrichten

Nachrichtenaustausch zwischen Objekten wird in der Objektbasierten Simulation nicht direkt unterstützt. Wenn man modellieren will, dass ein Objekt einem anderen eine Nachricht versendet, so kann man dies nur dadurch lösen, dass in einer Reaktionsregel als Folgeereignis der Empfang der Nachricht durch das Empfängerobjekt erzeugt wird. Dies ist eine unnatürliche Modellierung, da ja der Versand und nicht der Empfang der Nachricht eine Reaktion auf das Ereignis darstellt.

In dieser Erweiterung können sich aktive Entitäten (Agenten) Nachrichten senden. Bei jedem Nachrichtentyp kann man den (Agenten-)Typ von Sender und Empfänger, eine (ggf. vom Zufall abhängige) Zeit angeben, wie lange es dauert, bis die Nachricht beim Empfänger ankommt (*Übertragungsdauer*) sowie die *Zuverlässigkeit*, das ist die Wahrscheinlichkeit dafür, dass die Nachricht auch beim Empfänger ankommt. Anders ausgedrückt ergibt sich für die Zuverlässigkeit der Wert *1 - Verlustwahrscheinlichkeit*. Wenn es sich um Nachrichten handelt, bei denen sichergestellt ist, dass sie auch beim Empfänger ankommen, ist die Zuverlässigkeit 1. Ein Nachrichtentyp kann in der nachfolgenden tabellarischen Form spezifiziert werden.

Nachricht	
Sender	
Empfänger	
Übertragungsdauer	
Zuverlässigkeit	

Übertragungsdauer und Zuverlässigkeit hängen dabei vom verwendeten Nachrichtenkanal ab. Der Nachrichtenkanal ist das Medium, durch das die Nachricht transportiert wird. Mögliche Nachrichtenkanäle sind z.B. Sprache, E-Mails, Briefe, Telefon, Telefax, SMS sowie (im Falle von Softwareagenten) Socket-Kommunikation über TCP/IP.

Anstelle der direkten Angabe von Übertragungsdauer und Zuverlässigkeit reicht es, für einen Nachrichtentyp den Nachrichtenkanal anzugeben, durch den dieser übertragen wird. Ein Nachrichtentyp kann dann auch in der nachfolgenden tabellarischen Form spezifiziert werden.

Nachricht	
Sender	
Empfänger	
Nachrichtenkanal	

Für den Nachrichtenkanal muss dann die Übertragungsdauer und die Zuverlässigkeit angegeben werden, was in der nachfolgenden tabellarischen Form geschehen kann.

Nachrichtenkanal	
Übertragungsdauer	
Zuverlässigkeit	

Nachrichten haben eine Menge P = { p_1, p_2, .. , p_k } von Parametern, wobei Anzahl und Typ der Parameter vom Nachrichtentyp abhängig sind.

Das Versenden einer Nachricht vom Typ n mit Parametern p_1, p_2, .. , p_k an einen Empfängeragenten a wird mit $n\ (p_1, p_2, .., p_k) \rightarrow a$ bezeichnet. Den Empfang einer Nachricht vom Typ n mit Parametern p_1, p_2, .. , p_k von einem Senderagenten a zum Zeitpunkt t wird mit *Empfang* $(n\ (p_1, p_2, .., p_k) \leftarrow a) @ t$ bezeichnet. Hat ein Nachrichtentyp keine Parameter, also ist k = 0, so sind die die Parameter umschließenden Klammern optional.

4.2 Externer/interner Agentenzustand und Umgebungssimulator

Eine weitere Erweiterung der Objektbasierten Simulation ist die Trennung zwischen externem und internem Zustand eines Agenten sowie die Einführung eines Umgebungssimulators.

Dabei haben die Agenten sowohl einen internen Zustand, der ihren Zustand aus ihrer subjektiven Sicht beschreibt, sowie einen externen (physikalischen) Zustand, der den tatsächlichen objektiven Zustand des Agenten beschreibt.

Die externen Zustände der Agenten und die Zustände der (passiven) Objekte werden von einem *Umgebungssimulator* verwaltet, der eine globale Sicht auf das System hat. Der Umgebungssimulator in der Simulation ist kein Äquivalent einer Instanz in der Realität. Er stellt ein Hilfsmittel dar, um ein System zu simulieren, bei dem (meistens) in der Realität keine Instanz vorhanden ist, die eine globale Sicht hat. Die internen Zustände der Agenten werden von einem *Agentensimulator* verwaltet, wobei es für jeden Agenten einen Agentensimulator gibt.

In der Simulation verwaltet der Umgebungssimulator die Umgebung, also die passiven Objekte und die externen Zustände der Agenten. Er ermittelt für jeden einzelnen Agenten seine Wahrnehmungen, teilt sie dem zugehörigen Agentensimulator mit und erfährt von ihm, welche Aktionen der Agent durchführt, woraufhin der Umgebungssimulator wieder ermittelt, welche Auswirkungen diese Aktionen auf die Umgebung haben und welche Wahrnehmungen für die Agenten sich daraus ergeben.

Jeder Agentensimulator verwaltet den internen Zustand eines Agenten. Er erfährt vom Umgebungssimulator die Wahrnehmungen des von ihm verwalteten Agenten und teilt ihm mit, welche Aktionen der Agent (als Reaktion auf diese Wahrnehmungen) durchführt. Damit lassen sich vollständig reaktive Agenten beschreiben. Um die Spezifikation von Agenten mit proaktivem Verhalten zu ermöglichen, kann ein Agentensimulator zusätzlich sogenannte interne Zeitereignisse verwalten, wobei zwischen einmaligen und periodischen internen Zeitereignissen unterschieden wird. Einmalige Zeitereignisse (Timeouts) haben den Zweck, dass der Agent auf das Verstreichen einer Zeitspanne reagieren kann. Periodische Zeitereignisse hingegen ermöglichen es, regelmäßige

Aktionen oder Überprüfungen des internen Zustands vorzunehmen sowie Pläne zu erstellen und auszuführen. Ein internes periodisches Zeitereignis hat ebenso wie ein exogenes Ereignis für den Umgebungssimulator einen oder mehrere Stränge mit einer Periodizität und einer Stopbedingung. Ein Strang für ein internes periodisches Zeitereignis kann in der nachfolgenden tabellarischen Form dargestellt werden.

internes Zeitereignis	
Periodizität	
Stopbedingung	

Die am System beteiligten Agenten haben eine lokale Sicht auf den Zustand des Systems. Da es auch denkbar ist, dass ihre Informationen über den globalen Zustand falsch sein kann, spricht man statt von ihren *Informationen* oder von ihrem *Wissen* auch von ihren *Überzeugungen* (engl.: beliefs).

Wie in der Objektbasierten Simulation wird der Zustand der Agenten bzw. Objekte gekapselt, allerdings mit dem Unterschied, dass in der Agentenbasierten Simulation es sich bei den internen Zuständen um einen mentalen Zustand handelt, der die Überzeugung des Agenten über den tatsächlichen Zustand darstellt und bei den externen Zuständen um den tatsächlichen (physikalischen) Zustand, während man bei der Objektbasierten Simulation nur den tatsächlichen Zustand betrachtet.

4.3 Agentenbasierte Simulation

In der Agentenbasierten Simulation wird das Modell der Objektbasierten Simulation um die beiden vorgestellten Erweiterungen, die Modellierung von Nachrichtenaustausch und die Trennung zwischen externem und internem Agentenzustand verbunden mit der Einführung eines Umgebungssimulators, erweitert.

Beide Erweiterungen sind logisch unabhängig voneinander, so dass man jede der beiden Erweiterungen verwenden könnte, um die Objektbasierte Simulation zu erweitern.

Die folgende informelle Grafik (Abbildung 8) veranschaulicht den Weg vom Basismodell der Diskreten-Ereignis-Simulation zur Agentenbasierten Simulation grafisch.

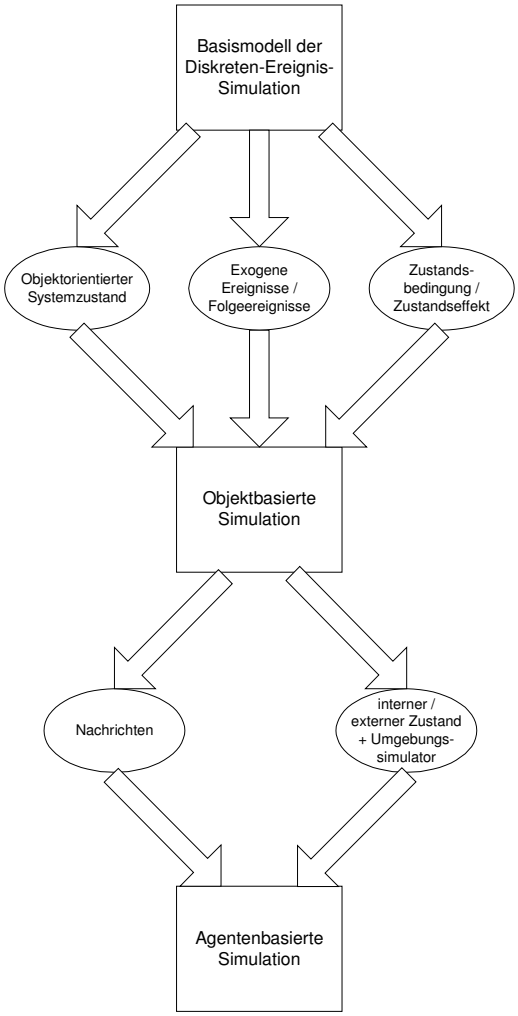

Abbildung 8: Der Weg vom Basismodell der Diskreten-Ereignis-Simulation zur Agentenbasierten Simulation

In Abschnitt 4.3.1 wird das Konzept der Agentenbasierten Simulation präzisiert und mathematisch formalisiert. Anschließend wird in Abschnitt 4.3.2 dargestellt, wie man Reaktionsregeln zur Spezifikation eines agentenbasierten Simulationsmodells verwenden kann. Um zu zeigen, dass die Agentenbasierte Simulation tatsächlich eine Verfeinerung der Objektbasierten Simulation und damit auch der Diskreten-Ereignis-Simulation ist, wird in Abschnitt 4.3.3 gezeigt, wie man ein mit Hilfe der Agentenbasierten Simulation spezifiziertes Modell in ein mit Hilfe der Objektbasierten Simulation spezifiziertes Modell transformiert. In Abschnitt 4.3.4 wird auch das bereits aus den Abschnitten 2.3 und 3.4.4 bekannte Fahrstuhl-Beispiel diesmal mit Hilfe der Agentenbasierten Simulation spezifiziert. Das aus den Abschnitten 2.2 und 3.4.3 bekannte Beispiel einer Warteschlange in einem Supermarkt ist nicht so komplex, dass eine Spezifikation mit Hilfe der Agentenbasierten Simulation eine Vereinfachung gegenüber der Objektbasierten Simulation bringen würde. Deshalb wird auf eine Spezifikation dieses Beispiels mit Hilfe der Agentenbasierten Simulation verzichtet.

4.3.1 Formalisierung

Das Simulationssystem in der Agentenbasierten Simulation besteht aus einem Umgebungssimulator und aus je einem Agentensimulator für jeden am System beteiligten Agenten. Der Systemzustand besteht aus dem Umgebungszustand, der durch den Umgebungssimulator verwaltet wird und den von den Agentensimulatoren verwalteten internen Zuständen der einzelnen Agenten. Der Umgebungszustand besteht dabei aus den externen Zuständen der Agenten sowie aus den Zuständen der (passiven) Objekte. Für die Struktur des internen Zustands eines Agenten müssen in der Formalisierung der Agentenbasierten Simulation keine Einschränkungen gemacht werden. Häufig ist jedoch der interne Zustand eines Agenten sein mentaler Zustand, der aus den Überzeugungen des Agenten, zu denen man auch Komponenten wie Erinnerungen des Agenten und Verpflichtungen, die ein Agent eingegangen ist, zählen kann.

Die Agentenbasierte Simulation läuft in Zyklen, die Interaktionen zwischen dem Umgebungssimulator und den Agentensimulatoren beinhalten, ab.

In Abschnitt 4.3.1.1 wird zunächst der Ablauf eines Zyklus in der Agentenbasierten Simulation informell, also natürlich-sprachlich beschrieben. In Abschnitt 4.3.1.2 wird dann dieser Ablauf mathematisch formal dargestellt. In Abschnitt 4.3.1.3 wird auf Zeitaspekte in der Agentenbasierten Simulation eingegangen, darunter der Zusammenhang zwischen Echtzeit und Simulationszeit, die Wahl der Zyklenlänge und die Verzögerung zwischen Handlungsentscheidung, -ausführung und -wahrnehmung.

4.3.1.1 Informelle Beschreibung des Ablaufs

Der Ablauf eines Zyklus in der Agentenbasierten Simulation sieht informell beschrieben wie folgt aus.

Schritt 1	Der Umgebungssimulator ermittelt alle in diesem Zyklus stattfindenden Ereignisse (die *aktuellen Ereignisse*). Dabei handelt es sich um: a) die Aktionen, die die Agenten am Ende des vergangenen Zyklus ausgeführt haben b) Ereignisse aus der Ereignismenge des Umgebungssimulators, die in diesen Zyklus fallen c) exogene Ereignisse, die in diesen Zyklus fallen.
Schritt 2	Der Umgebungssimulator berechnet aus dem aktuellen Umgebungszustand und den aktuellen Ereignissen einen neuen Umgebungszustand, eine Menge von Folgeereignissen sowie für jeden Agenten seine Wahrnehmungen für den folgenden Zyklus. Zu den Wahrnehmungen gehört auch der Empfang von Nachrichten, die andere Agenten versendet haben.
Schritt 3	Der Umgebungssimulator sendet jedem Agentensimulator die (ggf. leere) Menge von Wahrnehmungen des durch ihn simulierten Agenten in diesem Zyklus (die er im vergangenen Zyklus ermittelt hat).
Schritt 4	Jeder Agentensimulator (auch die, die eine leere Menge von Wahrnehmungen empfangen haben) bestimmt zunächst die Menge der in diesem Zyklus stattfindenden internen Ereignisse (die *aktuellen internen Ereignisse*). Dabei handelt es sich um: a) die empfangenen Wahrnehmungen b) interne Zeitereignisse aus der Zeitereignismenge des Agentensimulators, die in diesen Zyklus fallen c) periodische interne Zeitereignisse, die in diesen Zyklus fallen. Anschließend berechnet der Agentensimulator aus dem aktuellem internen Agentenzustand und aus den aktuellen internen Ereignissen einen neuen internen Agentenzustand, eine Menge von internen Folge-Zeitereignissen sowie eine Menge von Aktionen, die der durch ihn simulierte Agent durchführt. Zu den Aktionen gehören auch an andere Agenten versendete Nachrichten. Die (ggf. leere) Menge von Aktionen sendet er an den Umgebungssimulator.
Schritt 5	Der Umgebungssimulator wartet ab, bis er von allen Agenten jeweils die Menge ausgeführter Aktionen empfangen hat, und setzt die Zeit weiter. Die Menge der ausgeführten Aktionen wird im darauf folgenden Zyklus berücksichtigt.

Abbildung 9: Ablauf eines Zyklus in der Agentenbasierten Simulation

4.3.1.2 Formale Beschreibung des Ablaufs

Dieser Abschnitt enthält eine formale Beschreibung des Zustands des Umgebungssimulators und der Agentensimulatoren sowie des Ablaufs eines Zyklus in der Agentenbasierten Simulation.

Der Zustand Z des Umgebungssimulators in der Agentenbasierten Simulation besteht aus:

- dem Umgebungszustand S
- der aktuellen Simulationszeit c

- der Ereignismenge *E*, wobei hier nur alle zukünftigen, also noch nicht ausgeführten Ereignisse enthalten sind.

Z lässt sich somit als Tupel *(S, c, E)* auffassen. Der Umgebungszustand *S* besteht aus den externen Zuständen der Agenten und aus den Zuständen der Objekte. Sei *n* die Anzahl der Agenten und *m* die Anzahl der Objekte, so lässt sich der Umgebungszustand beschreiben als S = { Ext_1, Ext_2, .. , Ext_n, O_1, O_2, .. , O_m }, wobei Ext_i der externe Zustand des i-ten Agenten und O_i der Zustand des i-ten Objekts ist.

Der Zustand Z_i des Agentensimulators des i-ten Agenten in der Agentenbasierten Simulation besteht aus:

- dem internen Zustand Int_i des i-ten Agenten
- der aktuellen internen Simulationszeit c_i
- der Zeitereignismenge ZE_i, wobei hier nur alle zukünftigen internen Zeitereignisse enthalten sind.

Z_i lässt sich somit als Tupel *(* Int_i, c_i, ZE_i *)* auffassen. Die Menge der Wahrnehmungen des i-ten Agenten wird mit W_i bezeichnet und die Menge der Aktionen, die der i-te Agent ausführt mit Akt_i.

Die Agentenbasierte Simulation läuft in Zyklen mit fester Länge Δc ab. Die Problematik der Wahl einer geeigneten Zyklenlänge wird in Abschnitt 4.3.1.3 näher erläutert.

In **Schritt 1** ermittelt der Umgebungssimulator die Menge der aktuellen Ereignisse E_{akt}, indem er die drei folgenden Ereignismengen bestimmt:

a) die Menge *Akt* von Aktionen, die die Agenten am Ende des vergangenen Zyklus ausgeführt haben

b) (Folge-)Ereignisse *Folg* aus der Ereignismenge *E* des Umgebungssimulators, die in diesen Zyklus fallen

c) exogene Ereignisse *Exog*, die in diesen Zyklus fallen.

Für die Bestimmung von *Folg* und *Exog* werden alle Ereignisse verwendet, deren Zeitpunkt im Intervall *[c ; c +* Δc*)* liegen. Bei der Bestimmung von *Exog* ist es möglich, dass mehrere exogene Ereignisse aus einem Strang enthalten sind, wenn die Periodizität des Stranges kleiner als die Zyklenlänge ist.

Der Umgebungssimulator bestimmt die Menge der aktuellen Ereignisse E_{akt} als die Vereinigung der drei ermittelten Ereignismengen.

$E_{akt} \leftarrow Akt \cup Folg \cup Exog.$

In **Schritt 2** wertet der Umgebungssimulator die Funktion *u* aus, die aus dem aktuellen Umgebungszustand *S* und den aktuellen Ereignissen E_{akt} einen neuen Umgebungszustand *S'*, eine Menge von Folgeereignissen E_{neu} sowie für jeden Agenten seine Wahrnehmungen W_i für den nachfolgenden Zyklus ermittelt. Dabei ist $u (S, E_{akt}) = (S', E_{neu}, W_1, W_2, .. , W_n)$. Hierbei bezeichnen *S* und *S'* konkrete Umgebungszustände, E_{akt} und E_{neu} konkrete Ereignismengen sowie $W_1, W_2, .. , W_n$ konkrete Wahrnehmungsmengen.

Der Umgebungssimulator aktualisiert den Umgebungszustand *S* und die Ereignismenge *E*.

$S \leftarrow S'$

$E \leftarrow E \setminus Folg \cup E_{neu}$

In **Schritt 3** sendet der Umgebungssimulator jedem Agentensimulator die aktuelle Simulationszeit *c* sowie die (ggf. leere) Menge von Wahrnehmungen des durch ihn simulierten Agenten, also an den

Agentensimulator des i-ten Agenten die Menge W_i. Dabei sendet er die Wahrnehmungen des Agenten für den aktuellen Zyklus. Das sind die Wahrnehmungen, die er im vergangenen Zyklus ermittelt hat.

Schritt 4 führt jeder Agentensimulator aus. Als erstes aktualisiert er seine interne Simulationszeit, indem er sie auf die vom Umgebungssimulator empfangene Zeit setzt.

$c_i \leftarrow c$

Anschließend ermittelt der Agentensimulator des i-ten Agenten die Menge der aktuellen internen Ereignisse E_i, indem er die drei folgenden Ereignismengen bestimmt:

a) die Menge der empfangenen Wahrnehmungen W_i

b) (Folge-)Zeitereignisse FZE_i aus der Zeitereignismenge ZE_i des Agentensimulators, die in diesen Zyklus fallen

c) periodische interne Zeitereignisse $Perio_i$, die in diesen Zyklus fallen

Für die Bestimmung von FZE_i und $Perio_i$ werden alle Zeitereignisse verwendet, deren Zeitpunkt im Intervall $[\ c_i\ ;\ c_i\ +\ \Delta c)$ liegen.

Der Agentensimulator des i-ten Agenten bestimmt die Menge der aktuellen Ereignisse E_i als die Vereinigung der drei ermittelten Ereignismengen.

$E_i \leftarrow W_i \cup FZE_i \cup Perio_i$.

Danach wertet der Agentensimulator des i-ten Agenten die Funktion a_i aus, die aus seinem aktuellen internen Zustand Int_i und den aktuellen internen Ereignissen E_i seinen neuen internen Zustand Int_i', interne Folge-Zeitereignisse $ZEneu_i$ sowie die Menge von Aktionen Akt_i, die der i-te Agent durchführt, berechnet. Dabei ist $a_i\ (\ Int_i,\ E_i) = (\ Int_i',\ ZEneu_i,\ Akt_i\ )$, wobei es sich bei Int_i und Int_i' um konkrete interne Zustände, bei E_i um eine konkrete Menge interner Ereignisse, bei $ZEneu_i$ um eine konkrete Menge interner Folge-Zeitereignisse und bei Akt_i um eine konkrete Menge von Aktionen handelt.

Der Agentensimulator des i-ten Agenten aktualisiert seinen internen Zustand Int_i sowie seine Zeitereignismenge ZE_i.

$Int_i \leftarrow Int_i'$

$ZE_i \leftarrow ZE_i \setminus FZE_i \cup ZEneu_i$

Er sendet seine Aktionen Akt_i an den Umgebungssimulator.

In **Schritt 5** wartet der Umgebungssimulator ab, bis er von allen Agenten jeweils die Menge Akt_i der ausgeführten Aktionen empfangen hat. Die Menge *Akt* der ausgeführten Aktionen stellt die Vereinigung der Aktionen aller Agenten dar, wobei die Auswirkungen der Aktionen erst im darauf folgenden Zyklus berücksichtigt werden.

$Akt \leftarrow Akt_1 \cup Akt_2 \cup .. \cup Akt_n$.

Am Ende des Zyklus setzt der Umgebungssimulator die Zeit weiter.

$c \leftarrow c + \Delta c$

Was genau in den Schritten 3, 4 und 5 mit *Senden* und *Empfangen* gemeint ist, hängt von der Implementierung des Simulationssystems, insbesondere von der Realisierung der Aufteilung in Umgebungssimulator und Agentensimulatoren ab. Denkbar wäre, dass ein zentrales Programm abläuft, in dem sequentiell alle Agentensimulatoren per Methodenaufruf ihre Wahrnehmungsmenge

mitgeteilt bekommen, Schritt 4 ausführen und als Ergebnis des Methodenaufrufs ihre Aktionenmenge zurückgeben. Möglich ist aber auch, dass die einzelnen Agentensimulatoren Threads sind, die nebenläufig zueinander ihren Schritt 4 ausführen. Eine weitere Möglichkeit wäre, dass es sich bei den einzelnen Agentensimulatoren um eigene Prozesse handelt, die möglicherweise auf mehreren Prozessoren oder Rechnern verteilt ablaufen. In allen Fällen ist eine Überlappung der Schritte 3 und 4 möglich.[44] Da der Umgebungssimulator in Schritt 5 wartet, bis alle Aktionen aller Agentensimulatoren vorliegen, hängt die Semantik des Simulationssystems nicht von der Realisierung der Aufteilung in Umgebungssimulator und Agentensimulatoren ab. Die Ausführungszeit der Simulation und damit ihre Effizienz kann jedoch sehr wohl davon abhängen. Man kann das Senden und Empfangen zwischen Umgebungssimulator und den Agentensimulatoren auch als Nachrichten auffassen, wobei es sich beim Ablauf der Simulation dabei um reale Nachrichten handelt, während die Nachrichten zwischen den Agenten simulierte Nachrichten sind.

Das Versenden einer Nachricht durch einen Agenten ist eine spezielle Aktion, so dass sich Nachrichten mit dem bisher eingeführten Formalismus darstellen lassen. Nachrichten, die ein Agent an einen anderen senden möchte, gehören zur Menge Akt_i, die in Schritt 4 versendet wird. Der Umgebungssimulator ermittelt dann in Schritt 2 des darauf folgenden Zyklus anhand der spezifizierten Übertragungsdauer und Zuverlässigkeit, ob und wann die Nachricht beim Empfängeragenten ankommt. Dem Empfängeragenten wird dann (im richtigen Zyklus) eine entsprechende Wahrnehmung, die den Empfang der Nachricht ausdrückt, gesendet.

4.3.1.3 Zeitaspekte in der Agentenbasierten Simulation

In diesem Abschnitt wird auf Zeitaspekte in der Agentenbasierten Simulation eingegangen, darunter der Zusammenhang zwischen Echtzeit und Simulationszeit, die Wahl der Zyklenlänge und die Verzögerung zwischen Handlungsentscheidung, Handlungsausführung und Handlungswahrnehmung.

Echtzeit vs. Simulationszeit

Festzuhalten ist zunächst, dass es sich bei der hier formalisierten Simulation um keine Echtzeitsimulation handelt. Deshalb ist mit Zeit im Folgenden immer Simulationszeit gemeint, deren Fortschritt in Abhängigkeit der Anzahl der Ereignisse, der Zyklenlänge und der Leistung des Rechners schneller oder langsamer als der Fortschritt der realen Zeit sein kann. Die Möglichkeit der Erweiterung der Formalisierung um Echtzeitsimulation wird im Ausblick in Abschnitt 9.3 diskutiert.

Die modellierte Domäne spielt dabei indirekt eine Rolle. So wählt man z.B. bei der Simulation einer Bevölkerungsentwicklung oder der Entwicklung des Arbeitsmarkts große Zyklenlängen (etwa 1 Monat oder 1 Jahr), während man sich z.B. bei der Simulation elektronischer Schaltungen oder Effekten der Kernphysik eine Zyklenlänge im Millisekunden oder gar Mikrosekundenbereich vorstellen kann. Die Anzahl der Ereignisse hängt nicht nur von der modellierten Domäne, sondern auch von konkreten Simulationsparametern ab. So dauert z.B. die Simulation eines Fahrerlosen Transportsystems mit 1000 Fahrzeugen erheblich länger als dies bei einem mit 10 Fahrzeugen der Fall wäre. Unmittelbar einsichtig ist die Tatsache, dass auf einem leistungsstarken Rechner eine Simulation schneller abläuft als auf einem nicht so leistungsstarken.

Wahl der Zyklenlänge / Zeitprogression

Verbunden mit der Aufteilung des Zeitablaufs in Zyklen ergibt sich die Frage, wie man eine geeignete Zyklenlänge zu wählen hat.

Wählt man diese zu lang, so besteht die Gefahr, dass Ereignisse, die eigentlich hintereinander stattfinden und sich gegenseitig beeinflussen können, innerhalb eines Zyklus stattfinden und somit als gleichzeitig angesehen werden.

Wählt man sie zu kurz, so gibt es sehr viele Zyklen, in denen alle oder fast alle Agentensimulatoren in Schritt 3 eine leere Menge von Wahrnehmungen mitgeteilt bekommen und alle oder fast alle Agentensimulatoren in Schritt 4 eine leere Menge von Aktionen an den Umgebungssimulator senden. Damit wäre die Effizienz der Simulationsausführung stark beeinträchtigt. Der Vorteil der Diskreten-

[44] In Abschnitt 9.2 wird diese Thematik ausführlich behandelt.

Ereignis-Simulation, die Zeit, in der sich der Systemzustand nicht ändert, überspringen zu können, wäre nicht mehr gegeben.

Eine Lösung für dieses Problem ist, kurze Zyklen zu verwenden und das Ausführungsmodell um den im Folgenden erläuterten zusätzlichen Mechanismus zu erweitern, der lediglich einen effizienzsteigernden Effekt hat.

Wenn der Agentensimulator weiß, dass der durch ihn simulierte Agent keine Aktionen mehr ausführen wird, solange er keine Wahrnehmungen empfängt, kann er sich bis zur nächsten Wahrnehmung des Agenten schlafen legen[45]. Dies ist der Fall, wenn es keine internen periodischen Zeitereignisse für diesen Agenten gibt und seine Zeitereignismenge ZE_i gerade leer ist. Weiß er zumindest, dass der Agent bis zu einem bestimmten Zeitpunkt t keine Aktionen mehr ausführen wird, solange er keine Wahrnehmungen empfängt, so kann er sich bis zu diesem Zeitpunkt schlafen legen. Das ist der Fall, wenn das nächste interne Zeitereignis (aus der Zeitereignismenge ZE_i oder ein periodisches Zeitereignis) den Zeitpunkt t hat. Der erste Fall ist also ein Spezialfall des zweiten Falls mit $t = \infty$. Der Umgebungssimulator sendet einem Agentensimulator, der sich schlafen gelegt hat, bis zum Zeitpunkt t keine leeren Wahrnehmungsmengen in Schritt 3 mehr, sondern nimmt eine leere Aktionsmenge als Antwort aus Schritt 4 an. Nichtleere Wahrnehmungsmengen in Schritt 3 werden immer versendet und sie haben den Effekt, dass der Agentensimulator nun nicht mehr schlafend ist. Er kann sich aber bei seiner Antwort in Schritt 4 sofort wieder schlafen legen. Wenn ein Agentensimulator dem Umgebungssimulator mitteilt, dass er sich bis zum Zeitpunkt t schlafen legt, so versendet er die entsprechende Benachrichtigung immer in Schritt 4 gemeinsam mit der Aktionenmenge.

Eine andere Möglichkeit der Handhabung der Zyklenlänge wäre, sie variabel zu gestalten. Der Umgebungssimulator könnte dann je nach Anzahl der auftretenden Ereignisse die Länge des Zyklus dynamisch verkürzen oder verlängern.

Verzögerung zwischen Wahrnehmung und Aktion

Es ergibt sich die Fragestellung, in wie weit es eine Verzögerung zwischen Wahrnehmungen und Aktionen gibt. Der schnellste Fall eines Wechselspiels von Wahrnehmungen und Aktionen sieht wie im Folgenden beschrieben aus. Ein Agent führt eine Aktion aus. Im darauf folgenden Zyklus führt der Umgebungssimulator das Handlungsereignis aus und ermittelt Wahrnehmungen für dieses Ereignis, die der Agent, sofern es sich um eine Aktion handelt, dessen Auswirkung auf die Umwelt er selbst wahrnimmt, im nächsten Zyklus empfängt. Er kann dann in diesem Zyklus auf die Wahrnehmung reagieren und eine neuerliche Aktion ausführen. Sowohl zwischen Aktion und Wahrnehmung der Auswirkung der Aktion als auch zwischen Wahrnehmung und Entscheidung zu einer neuen Aktion und der Ausführung dieser Aktion liegt also ein Zyklus.

Es ergibt sich folgendes Bild:

Zyklus n	Entscheidung für eine Handlung (= subjektive Handlungsausführung beim Agentensimulator)
Zyklus n+1	Handlungsereignis (= objektive Handlungsausführung beim Umgebungssimulator)
Zyklus n+2	Wahrnehmung des Handlungsereignisses und seiner Effekte und neue Handlungsentscheidungen (= subjektive Handlungsausführung beim Agentensimulator)

Abbildung 10: Verzögerungen zwischen Handlungsentscheidung, -ausführung und -wahrnehmung

Bei biologischen Agenten sollte eine Handlungsentscheidung im gleichen Zyklus wie die Wahrnehmung nur bei unverzügliche Entscheidungen/ Willensakte vorkommen, deren Dauer in der

[45] Dieser Begriff klingt zwar sehr technisch und nicht auf das Konzept agentenorientierter Modellierung bezogen, aber da es hier um eine simulationstechnische Konstruktion geht, die das Ergebnis des Simulationsauflaufs nicht beeinflusst, sondern nur seine Effizienz, ist ein solcher Begriff durchaus angemessen.

Größenordnung des mentalistischen Simulationstakts (ca. 100 ms) liegt. Möchte man einen bewusste Entscheidungsfindung simulieren, so kann eine Entscheidung innerhalb des gleichen Zyklus abhängig von der Wahl der Zyklenlänge unrealistisch sein. Deshalb bietet es sich dort an, für den Vorgang der Wahrnehmung und Entscheidungsfindung eine gewisse Zeitspanne zu veranschlagen, die z.B. bei der Simulation eines menschlichen Schachspielers unter Umständen sehr lang sein kann. Realisieren kann man die Simulation von „Nachdenken“ durch ein internes Folge-Zeitereignis in der Zukunft, das die Beendigung des Entscheidungsfindungsprozesses darstellt. Als Reaktion auf dieses Folge-Zeitereignis wird dann die ermittelte Aktion ausgeführt. Genauso verhält es sich bei komplexen Berechnungen, die ein Softwareagent durchführt. Dort ist der Zeitpunkt der Beendigung der Berechnung als internes Folge-Zeitereignis zu modellieren.

4.3.2 Reaktionsregeln

Ein konkretes agentenbasiertes Simulationssystem lässt sich mit Hilfe der Funktion u für den Umgebungssimulator und der Funktionen a_i für die Agentensimulatoren beschreiben. Allerdings ist es schwierig, diese Funktionen explizit anzugeben, so dass man sie aus Reaktionsregeln zusammensetzt. Bereits im Basismodell der Diskreten-Ereignis-Simulation wurde die Transitionsfunktion f aus mehreren Regeln und in der Objektbasierten Simulation aus mehreren Reaktionsregeln zusammengesetzt. Auch in der Agentenbasierten Simulation werden die Funktionen u und a_i (i = 1 .. n) aus Reaktionsregeln zusammengesetzt, wobei die Reaktionsregeln für die Funktion u auch als Reaktionsregeln für den Umgebungssimulator oder kurz *U-Regeln* und die Reaktionsregeln für die Funktion a_i auch als Reaktionsregel für den Agenten i oder kurz *A-Regeln* bezeichnet werden.

In Abschnitt 4.3.2.1 werden Reaktionsregeln für Agenten (A-Regeln) und in Abschnitt 4.3.2.2 Reaktionsregeln für den Umgebungssimulator (U-Regeln) vorgestellt.

4.3.2.1 Reaktionsregeln für Agenten (A-Regeln)

Die Reaktionsregeln für einen Agenten in der Agentenbasierten Simulation unterscheiden sich von den Reaktionsregeln für Objekte in der Objektbasierten Simulation zum einen dadurch, dass sie keine Folgeereignisse festlegen, sondern Aktionen. Da es sich bei den Reaktionsregeln um dem Agenten zugeordnete Regeln handelt, entfällt zum anderen die Angabe des Reaktors. Auch wenn Nachrichten spezielle Aktionen sind, werden sie aus Gründen der Übersichtlichkeit in der tabellarischen Darstellung von ihnen getrennt, da die Reaktion des Umgebungssimulators auf Nachrichten nicht explizit mittels Reaktionsregeln ausgedrückt werden muss, sondern sich aus der Nachrichtensemantik ergibt.

Eine Reaktionsregel für einen Agenten kann in der nachfolgenden tabellarischen Form dargestellt werden.

Ereignis	
Ereignisbedingung	
Zustandsbedingung	
Zustandseffekt	
Folge-Zeitereignisse	
Nachrichten	
Aktionen	

Enthält die Menge der aktuellen internen Ereignisse des Agenten genau ein internes Ereignis (also E_i = { E }), so ergibt sich der Wert der Funktion a_i aus der Anwendung der Reaktionsregel für dieses interne Ereignis. Sei R_j mit

Ereignis	E
Ereignisbedingung	EB
Zustandsbedingung	ZB
Zustandseffekt	ZE
Folge-Zeitereignisse	FZ
Nachrichten	N
Aktionen	A

die ausgewählte Reaktionsregel[46], dann ist

a_i (Int_i, { E }) = (Upd (Int_i, ZE), FZ, N $\cup$ A).

Es wird also der Zustand gemäß der Reaktionsregel geändert, die Folge-Zeitereignisse werden verwendet und als Aktionsmenge werden die gesendeten Nachrichten und die (sonstigen) ausgeführten Aktionen gesendet.

Um die Funktion auf mehrere interne Ereignisse ausweiten zu können, wird eine Hilfsfunktion a_i' eingeführt, die der Verarbeitung eines einzelnen internen Ereignisses durch eine Reaktionsregel entspricht. Sei Z der interne Zustand des Agenten, E ein internes Ereignis, ZE der Zustandseffekt der durch Z und E bestimmten Reaktionsregel[47] und seien FZ die Folge-Zeitereignisse, N die Nachrichten und A die Aktionen dieser Reaktionsregel, so ist

a_i' (Z , E) = (Upd (Z , ZE) , FZ, N $\cup$ A).

Im Falle einer einelementigen Menge interner Ereignisse (also E_i = { E }) gilt also

a_i (Int_i, { E }) = a_i' (Int_i, E).

Enthält die Menge interner Ereignisse E_i mehrere Ereignisse, so können spezielle Regeln, die das Auftreten gleichzeitiger interner Ereignisse behandeln, angewendet werden. Diese Regeln werden am Ende dieses Abschnitts vorgestellt.

Sind solche Regeln für mehrere interne Ereignisse nicht vorhanden, so müssen die internen Ereignisse nacheinander und zur Vermeidung unerwünschter nichtdeterministischer Effekte bzw. zur Sicherstellung der Reproduzierbarkeit der Ergebnisse (siehe Abschnitt 1.2) in einer definierten Reihenfolge verarbeitet werden. Die Reihenfolge der Verarbeitung der internen Ereignisse ist entscheidend, da die Verarbeitung eines internen Ereignisses den internen Zustand des Agenten dahingehend verändern kann, dass nun eine andere Reaktionsregel ausgewählt wird. Aber auch wenn die Wahl der Reaktionsregel sich nicht ändert, so ist es trotzdem möglich, dass die Anwendung der Zustandstransformation auf den internen Agentenzustand nicht kommutativ ist. Den Wert der Funktion a_i erhält man, indem man die internen Ereignisse nacheinander verarbeitet. Dabei verwendet man den neuen internen Zustand des Agenten nach einer Verarbeitung eines internen Ereignisses als alten Zustand für die Verarbeitung des nächsten internen Ereignisses. Der neue interne Zustand des Agenten nach Verarbeitung der Menge interner Ereignisse ergibt sich dann als der neue interne Zustand des Agenten nach der Verarbeitung des letzten internen Ereignisses. Die Menge der erzeugten Folge-Zeitereignisse ergibt sich aus der Vereinigung der erzeugten Folge-Zeitereignisse bei den Verarbeitungen der einzelnen internen Ereignisse und die Menge der ausgeführten Aktionen aus der Vereinigung der durchgeführten Aktionen bei den Verarbeitungen der einzelnen internen Ereignisse.

Für eine allgemeine n-elementige Menge interner Ereignisse E_i ergibt sich

[46] Das setzt voraus, dass EB und ZB gelten.

[47] Das ist die Reaktionsregel, die ausgewählt wird, wenn der Agent im Zustand Z das interne Ereignis E verarbeitet.

a_i (Int_i, { E_{i1} , E_{i2}, .. , E_{in} }) = (Z_n, $FZ_1 \cup FZ_2 \cup .. \cup FZ_n$, $Akt_1 \cup Akt_2 \cup .. \cup Akt_n$) ,

wobei

a_i' (Z_{k-1} , E_{ik}) = (Z_k, FZ_k, Akt_k)

für k = 1 .. n mit $Z_0 = Int_i$ gilt.

Dabei ergeben sich die zweiten Indizes der internen Ereignisse aus der festgelegten Reihenfolge ihrer Ausführung.

Für eine zweielementige Menge interner Ereignisse E_i = { E_{i1} , E_{i2} } ergibt sich im speziellen

a_i (Int_i, { E_{i1} , E_{i2} }) = (Z_2, $FZ_1 \cup FZ_2$, $Akt_1 \cup Akt_2$) ,

wobei

a_i' (Int_i , E_{i1}) = (Z_1 , FZ_1, Akt_1) und

a_i' (Z_1 , E_{i2}) = (Z_2 , FZ_2, Akt_2) gelten.

Für die leere Menge interner Ereignisse $E_i = \varnothing$ ergibt sich im speziellen

a_i (Int_i, $\varnothing$) = (Int_i , $\varnothing$, $\varnothing$),

also bleibt der interne Zustand unverändert, es werden keine Folge-Zeitereignisse erzeugt und keine Aktionen durchgeführt.

Verarbeitet man mehrelementige Mengen interner Ereignisse auf die genannte Art und Weise, so ignoriert man die Gleichzeitigkeit der internen Ereignisse und legt sie künstlich hintereinander, um sie dann einzeln auszuführen. Da es keine plausible allgemeingültige Verfahrensweise gibt, wie ein Agent auf gleichzeitige interne Ereignisse reagiert, ist diese Vorgehensweise durchaus legitim. Allerdings kann der Modellierer eines konkreten Simulationssystems durchaus wissen, wie der Agent auf mehrere gleichzeitige interner Ereignisse reagieren soll. Deshalb kann er Reaktionsregeln mit mehreren internen Ereignissen angeben. Eine solche Reaktionsregel wird dann ausgewählt, wenn alle angegebenen internen Ereignisse gleichzeitig auftreten. Das Schema einer solchen Reaktionsregel exemplarisch für zwei gleichzeitige interne Ereignisse sieht wie folgt aus.

Ereignis 1	E_1
Ereignisbedingung1	EB_1
Ereignis 2	E_2
Ereignisbedingung2	EB_2
Zustandsbedingung	ZB
Zustandseffekt	ZE
Folge-Zeitereignisse	FZ
Nachrichten	N
Aktionen	A

Eine Regel hat gegenüber einer anderen den Vorrang, wenn die Menge der internen Ereignisse, die sie beinhaltet, eine echte Obermenge der internen Ereignisse der anderen Regel darstellt.

Die Reaktionsregel aus der Tabelle kann also ausgewählt werden, wenn sowohl E_1 als auch E_2 in der Ereignismenge enthalten sind und EB_1, EB_2 und ZB gelten. In diesem Falle wären sowohl E_1 als

auch E_2 abgearbeitet. Die Regel hat Vorrang vor allen Regeln, die nur E_1 oder nur E_2 beinhalten. Wäre dagegen noch ein internes Ereignis E_3 in der Ereignismenge und gäbe es eine Reaktionsregel, die E_1, E_2 und E_3 beinhaltet, so hätte diese Regel den Vorrang.

Da durch die Mengeninklusion keine totale Ordnung auf den Reaktionsregeln existiert, muss zur Vermeidung unerwünschter nichtdeterministischer Effekte eine definierte Reihenfolge unter den Reaktionsregeln vorhanden sein, die als nachrangiges Kriterium zur Regelauswahl herangezogen werden kann.

Allgemeiner als hier dargestellt könnte man in Reaktionsregeln beliebige logische Ausdrücke von Ereignissen für die Regeln zulassen, also Konjunktion, Disjunktion und Negation von Ereignissen. Der hier vorgestellte Formalismus erlaubt auf den ersten Blick nur Konjunktionen. Allerdings lassen sich beliebige Formeln von Konjunktionen und Disjunktionen von Ereignissen darstellen, da sich eine Formel mit Konjunktionen und Disjunktionen in eine äquivalente Formel transformieren lässt, die die Form einer Disjunktion von Konjunktionen hat (Disjunktive Form). Für jede Konjunktion wird dann der Regelrumpf (Zustandsbedingung, Zustandseffekt, Folge-Zeitereignisse, Nachrichten, Aktionen) dupliziert. Negationen von Ereignissen lassen sich darstellen, wenn es eine weitere Regel gibt, die das negierte Ereignis in nicht negierter Form und ansonsten die gleichen Ereignisse enthält. In dem Fall kann das negierte Ereignis weggelassen werden, da aufgrund des Vorrangs der anderen Regel dieses Ereignis implizit schon ausgeschlossen ist und damit die Regel nur dann ausgeführt wird, wenn die Negation des Ereignisses zutrifft.

4.3.2.2 Reaktionsregeln für den Umgebungssimulator (U-Regeln)

Die Reaktionsregeln für den Umgebungssimulator unterscheiden sich zum einen von den Reaktionsregeln aus der Objektbasierten Simulation dadurch, dass kein Reaktor angegeben wird, sondern Zustandsbedingung und Zustandseffekt aus der globalen Sicht des Umgebungszustandes beschrieben werden. Zum anderen werden zusätzlich zu den Folgeereignissen Wahrnehmungen für die Agenten aufgeführt, wobei hier für jeden Agenten anzugeben ist, welche Wahrnehmungen er empfängt. Dabei wird erst der Agent genannt, dann folgt ein Doppelpunkt und dann die Menge der Wahrnehmungen für diesen Agenten. Die Aufzählung von Agenten, die keine Wahrnehmung empfangen ist dabei optional. Handelt es sich bei dem Ereignis um eine Aktion eines Agenten, so wird bei *Ereignis* erst der Agent genannt, dann folgt ein Doppelpunkt und dann die Aktion des Agenten. Eine Reaktionsregel für den Umgebungssimulator kann in der nachfolgenden tabellarischen Form spezifiziert werden.

Ereignis	
Ereignisbedingung	
Zustandsbedingung	
Zustandseffekt	
Folgeereignisse	
Wahrnehmungen	

Enthält die Menge der aktuellen Ereignisse genau ein Element (also E_{akt} = { E }), so ergibt sich der Wert der Funktion *u* aus der Anwendung der Reaktionsregel für dieses Ereignis. Sei R_j mit

Ereignis	E
Ereignisbedingung	EB
Zustandsbedingung	ZB
Zustandseffekt	ZE
Folgeereignisse	FE
Wahrnehmungen	$\{ A_1 : W_1,\ A_2 : W_2, .., A_n : W_n \}$

die ausgewählte Reaktionsregel[48], dann ist

u (S, { E }) = (Upd (S, ZE), FE, W_1, W_2, .. , W_n).

Es wird also der Zustand gemäß der Reaktionsregel geändert, die Folgeereignisse werden verwendet und die Wahrnehmungen, die aus dieser Regel entspringen, werden an die Agenten gesendet.

Um die Funktion auf mehrere Ereignisse ausweiten zu können, wird eine Hilfsfunktion u' eingeführt, die der Verarbeitung eines einzelnen Ereignisses durch eine Reaktionsregel entspricht. Sei S der Umgebungszustand, E ein Ereignis und seien ZE der Zustandseffekt, FE die Folgeereignisse, und W_1, W_2, .. , W_n die Wahrnehmungen der durch S und E bestimmten Reaktionsregel[49] , so ist

u' (S, E) = (Upd (S, ZE), FE, W_1, W_2, .. , W_n)

Im Falle einer einelementigen Menge aktueller Ereignisse (also E_{akt} = { E }) gilt also

u (S, { E }) = u' (S, E).

Bei mehrelementigen Mengen aktueller Ereignisse erhält man den Wert der Funktion u, indem man die Ereignisse nacheinander verarbeitet. Dabei verwendet man den neuen Umgebungszustand nach einer Verarbeitung eines Ereignisses als alten Umgebungszustand für die Verarbeitung des nächsten Ereignisses. Der neue Umgebungszustand nach Verarbeitung der Menge der aktuellen Ereignisse ergibt sich dann als der Umgebungszustand nach der Verarbeitung des letzten Ereignisses. Die Menge der erzeugten Folgeereignisse ergibt sich aus der Vereinigung der erzeugten Folgeereignisse bei den Verarbeitungen der einzelnen Ereignisse und die Menge der Wahrnehmungen für jeden Agenten aus der Vereinigung der Mengen der Wahrnehmungen für jeden Agenten bei den Verarbeitungen der einzelnen Ereignisse.

Für eine allgemeine m-elementige Menge E_{akt} von aktuellen Ereignissen ergibt sich

u (S, { E_1 , E_2, .. , E_m }) = (S_m, $FE_1 \cup FE_2 \cup .. \cup FE_m$, $W_{11} \cup W_{12} \cup .. \cup W_{1m}$, $W_{21} \cup W_{22} \cup .. \cup W_{2m}$, .., $W_{n1} \cup W_{n2} \cup .. \cup W_{nm}$),

wobei

u' (S_{k-1} , E_k) = (S_k, FE_k, W_{1k}, W_{2k}, .. , W_{nk})

für k = 1 .. m mit S_0 = S gilt.

Dabei ergeben sich die Indizes der Ereignisse aus der festgelegten Reihenfolge ihrer Ausführung. Die Wahrnehmungsmenge W_{ij} stellt die Wahrnehmungen des i-ten Agenten aufgrund des j-ten Ereignisses dar.

Für die leere Menge aktueller Ereignisse $E_{akt} = \varnothing$ ergibt sich im speziellen

u (S, $\varnothing$) = (S , $\varnothing$, $\varnothing$, .., $\varnothing$),

---

[48] Das setzt voraus, dass EB und ZB gelten.

[49] Das ist die Reaktionsregel, die ausgewählt wird, wenn der Umgebungssimulator beim Umgebungszustand S das Ereignis E verarbeitet.

also bleibt der Umgebungszustand unverändert, es werden keine Folgeereignisse erzeugt und keine Wahrnehmungen versendet.

Wie schon bei den Reaktionsregeln für Agenten ist es auch bei den Reaktionsregeln für den Umgebungssimulator möglich, Reaktionsregeln mit mehreren Ereignissen anzugeben. Das Schema einer solchen Reaktionsregel exemplarisch für zwei gleichzeitige Ereignisse sieht wie folgt aus.

Ereignis 1	E_1
Ereignisbedingung1	EB_1
Ereignis 2	E_2
Ereignisbedingung2	EB_2
Zustandsbedingung	ZB
Zustandseffekt	ZE
Folgeereignisse	FZ
Wahrnehmungen	$\{ A_1 : W_1,\ A_2 : W_2, .., A_n : W_n \}$

Analog zu den Reaktionsregeln für Agenten mit mehreren internen Ereignissen kann eine Reaktionsregel für den Umgebungssimulator mit mehreren Ereignissen dann ausgewählt werden, wenn alle angegebenen Ereignisse gleichzeitig auftreten. Eine Regel hat auch hier gegenüber einer anderen den Vorrang, wenn die Menge der Ereignisse, die sie beinhaltet, eine echte Obermenge der Ereignisse der anderen Regel darstellt.

4.3.3 Abbildung auf die Objektbasierte Simulation

Um zu zeigen, dass die Agentenbasierte Simulation tatsächlich eine Verfeinerung der Objektbasierten Simulation und damit auch der Diskreten-Ereignis-Simulation ist, wird in diesem Abschnitt gezeigt, wie man ein mit Hilfe der Agentenbasierten Simulation spezifiziertes Modell in ein mit Hilfe der Objektbasierten Simulation spezifiziertes Modell transformiert.

Zunächst wird ein Agentenbasiertes Simulationsmodell in ein äquivalentes Agentenbasiertes Simulationsmodell ohne Nachrichten transformiert. Nachrichten werden dabei als normale Aktionen aufgefasst. Die Reaktionsregel für den Umgebungssimulator auf eine Nachricht sieht so aus, dass er als Folgeereignis der Empfang der Nachricht durch das Empfängerobjekt erzeugt. Es wird kein Folgeereignis erzeugt, wenn eine Zuverlässigkeit kleiner Eins angegeben wurde und mittels Zufallszahlen bestimmt wurde, dass diese Nachricht verloren gehen soll. Auf dieses erzeugte Folgeereignis reagiert der Umgebungssimulator bei seinem Auftreten damit, dass er dem Empfängeragenten den Empfang der Nachricht als Wahrnehmung sendet. Somit lässt sich ein Agentenbasiertes Simulationsmodell in ein äquivalentes Agentenbasiertes Simulationsmodell ohne Nachrichten transformieren.

Das diesem Agentenbasierten Simulationsmodell entsprechende Objektbasierte Simulationsmodell enthält lediglich zwei Objekte. Ein Objekt U entspricht dabei dem Umgebungssimulator, das andere Objekt A der Menge aller Agentensimulatoren. Der Zustand des Objekts U beinhaltet den Umgebungszustand sowie eine Menge von Ereignissen E_{akt}. Der Zustand des Objekts A beinhaltet die internen Agentenzustände sowie für jeden Agenten eine Menge von Ereignissen E_i. Neben den normalen Ereignistypen aus dem Agentenbasierten Simulationsmodell gibt es zwei Hilfsereignisse *Berechnung_U* und *Berechnung_A*, die ausdrücken sollen, dass das Objekt U bzw. das Objekt A an der Reihe mit seinen Berechnungen ist. Interne Folge-Zeitereignisse der Agenten werden als normale Ereignisse, interne periodische Zeitereignisse der Agenten als normale exogene Ereignisse angesehen.

Auf die normalen Ereignistypen aus dem Agentenbasierten Simulationsmodell (exogene Ereignisse und vom Umgebungssimulator erzeugte Folgeereignisse) reagiert Objekt U damit, dass es sie in die Menge E_{akt} in seinem Zustand aufnimmt.

Das Ereignis *Berechnung_U* enthält als Parameter die Aktionen der Agenten. Auf das Ereignis *Berechnung_U* zum Zeitpunkt *c* reagiert Objekt U damit, dass es die Aktionen der Agenten zur Menge E_{akt} in seinem Zustand hinzufügt. Es wendet seine Funktion *u* aus der Agentenbasierten Simulation an, berücksichtigt die Zustandsänderung und erzeugt die Folgeereignisse gemäß *u* sowie ein weiteres Folgeereignis *Berechnung_A* zum Zeitpunkt *c*, das als Parameter die im vorherigen Zyklus ermittelten Wahrnehmungen der Agenten enthält, während es die aktuell ermittelten Wahrnehmungen in der Menge *W* in seinem Zustand sich merkt.

Auf die internen Zeitereignisse (Folge-Zeitereignisse und periodische Zeitereignisse) reagiert Objekt A damit, dass es sie in die entsprechende Menge E_i in seinem Zustand aufnimmt.

Das Ereignis *Berechnung_A* enthält als Parameter die Wahrnehmungen der Agenten. Auf das Ereignis *Berechnung_A* zum Zeitpunkt *c* reagiert Objekt A damit, dass es die Wahrnehmungen der Agenten jeweils zur entsprechenden Menge E_i in seinem Zustand hinzufügt. Es wendet die Funktionen a_i aus der Agentenbasierten Simulation an, berücksichtigt die Zustandsänderungen und erzeugt die Folge-Zeitereignisse gemäß den Funktionen a_i sowie ein weiteres Folgeereignis *Berechnung_U* zum Zeitpunkt $c + \Delta c$, das als Parameter die Aktionen der Agenten enthält.

Somit lässt sich ein Agentenbasiertes Simulationsmodell auf ein äquivalentes Objektbasiertes Simulationsmodell zurückführen.

Eine natürlichere Abbildung des Agentenbasierten Simulationsmodells ohne Nachrichten in ein äquivalentes Objektbasiertes Simulationsmodell würde sich ergeben, wenn man jedes Objekt, jeden externen Agentenzustand und jeden internen Agentenzustand als Objekt im Objektbasierten Simulationsmodell auffassen würde. Die Regeln für den Umgebungssimulator könnten bestehen bleiben, wobei die Wahrnehmungen für die Agenten ebenfalls als Folgeereignisse modelliert würden. Die Regeln für die Agentensimulatoren müssten lediglich so abgeändert werden, dass das Objekt, dass den internen Agentenzustand repräsentiert, als Reaktor eingesetzt wird. Ausgeführte Aktionen würden als Folgeereignis modelliert. Nachteil dieser natürlicheren Abbildung ist jedoch, dass die Aufteilung der Simulation in Zyklen verloren geht. Somit sind auch keine Reaktionsregeln für mehrere gleichzeitige Ereignisse mehr möglich. Gleichzeitige Aktionen von Agenten, die gegenseitig die Auswirkungen ihrer Aktionen wahrnehmen, wären nicht mehr möglich. Somit könnte man nur noch von einem Simulationsmodell mit ähnlicher Semantik, aber nicht mehr von einem äquivalenten Simulationssystem sprechen.

4.3.4 Beispiel: Fahrstühle

Das bereits in den Abschnitten 2.3 und 3.4.4 beschriebene Beispiel der Fahrstühle wird nun mit Hilfe der Agentenbasierten Simulation spezifiziert.

Um das Lesen zu erleichtern, werden auch hier die Erläuterungen zu den Regeln aus den Abschnitten 2.3 und 3.4.4 bei den Erläuterungen in diesem Abschnitt wiederholt. Außerdem wird die Nummer der äquivalenten Regel aus Abschnitt 3.4.4 angegeben. Die Benennung der Regeln entspricht der dortigen Benennung. Dabei entsprechen in der Nummerierung die Regeln für die Wahrnehmungen *Kommt_an* und *Wird_angefordert* sowie für das Zeitereignis *Will_fahren* den Regeln für die gleichnamigen Ereignisse aus der Objektbasierten Simulation. Die Regeln für die Nachrichten *Erlaubnisanfrage* und *Erlaubnis* entsprechen in der Nummerierung den Regeln für die Ereignisse *Erhält_Erlaubnisanfrage* und *Erhält_Erlaubnis* aus der Objektbasierten Simulation.

Bei den Zustandsbedingungen und den Zustandeffekten sowie bei den Zusicherungen wird die Sprache OCL benutzt.

4.3.4.1 Ereignistypen und Systemzustand

Die Ereignistypen *Kommt_an* und *Wird_angefordert* kommen auch in der Agentenbasierten Simulation vor. Sie werden vom Umgebungssimulator verarbeitet. Für beide Ereignistypen gibt es einen gleichnamigen Wahrnehmungstyp, der der Wahrnehmung der Ereignisse durch die Fahrstuhl-Agenten entspricht. Die Ereignisse werden vollständig und unverfälscht wahrgenommen.

Die einzige Aktion, die ein Fahrstuhl-Agent durchführen kann, ist *Fährt_an*.

In den Agentensimulatoren gibt es das interne Zeitereignis *Will_fahren*, das als Folge-Zeitereignis erzeugt werden kann. Interne periodische Zeitereignisse gibt es keine.

Es gibt zwei Nachrichtentypen, *Erlaubnisanfrage* und *Erlaubnis*.

Nachricht	**Erlaubnisanfrage ( )**
Sender	**Fahrstuhl**
Empfänger	**Fahrstuhl**
Übertragungsdauer	Δ
Zuverlässigkeit	**1**

Nachricht	**Erlaubnis ( )**
Sender	**Fahrstuhl**
Empfänger	**Fahrstuhl**
Übertragungsdauer	Δ
Zuverlässigkeit	**1**

Da es unter Umständen sehr viele Nachrichtentypen zwischen Agenten geben kann, bietet sich die nachfolgende kompaktere Darstellung an, bei der alle Nachrichten, die den gleichen Sender- und Empfängertyp haben, in einer Tabelle zusammengefasst sind.

Sender	Empfänger	
Fahrstuhl	**Fahrstuhl**	
Nachricht	Übertragungsdauer	Zuverlässigkeit
Erlaubnisanfrage ( )	Δ	**1**
Erlaubnis ( )	Δ	**1**

Der Umgebungszustand besteht aus den externen Zuständen der beiden einzigen Agenten, der beiden Fahrstühle. Objekte gibt es keine.

Der externe Zustand eines Fahrstuhls beinhaltet die Information, ob er steht oder fährt und an welchem Ort er sich befindet (wenn er steht) bzw. zuletzt befand (wenn er fährt).

Das Attribut *bewZust* bestimmt den aktuellen Bewegungszustand und kann die Werte *fahrend* und *stehend* annehmen.

Das Attribut *etage* besagt, dass sich der Fahrstuhl in Etage *etage* (bei Bewegungszustand *stehend*) oder auf dem Weg von Etage *etage* weg (bei Bewegungszustand *fahrend*) befindet.

Der interne Zustand des Fahrstuhls beinhaltet ebenfalls diese beiden Attribute, wobei die Informationen des Fahrstuhls bezüglich dieser Attribute stets korrekt sind, also mit seinem tatsächlichen Zustand, dem externen Zustand übereinstimmen[50]. Außerdem beinhaltet der interne Zustand eines Fahrstuhls, ob er der Bevorrechtigte ist, ob er sich vorgemerkt hat, dass der andere Fahrstuhl auf eine Erlaubnis von ihm wartet, ob er selbst auf eine Erlaubnis des anderen Fahrstuhls wartet, welche Anforderungen er erhalten hat und ob diese Anforderungen von außen kamen.

Das Attribut *bevorrechtigt* drückt aus, ob der Fahrstuhl sich selbst als bevorrechtigt ansieht, wobei *bevorrechtigt* die Werte *true* und *false* annehmen kann.

[50] In der Simulation stimmt dies nicht ganz, da im internen Zustand bei Anfahren sofort in den Bewegungszustand *fahrend* gewechselt wird, im externen Zustand allerdings erst beim Verarbeiten der Aktion im darauf folgenden Zyklus. Für die Dauer eines Zyklus weichen interner und externer Zustand hier also jeweils voneinander ab.

Das Attribut *vorgemerkt* drückt aus, ob der Fahrstuhl sich vorgemerkt hat, dass der andere Fahrstuhl auf eine Erlaubnis von ihm wartet, wobei *vorgemerkt* die Werte *true* und *false* annehmen kann.

Das Attribut *wartend* drückt aus, ob der Fahrstuhl darauf wartet, dass der andere Fahrstuhl ihm eine Erlaubnis erteilt, wobei *wartend* die Werte *true* und *false* annehmen kann.

Ein Fahrstuhl kann bis zu zwei Anforderungen erhalten haben, die jeweils als Informationen die Etage enthalten, zu der der Fahrstuhl angefordert wurde sowie das Attribut *außen*, das die Werte *true* und *false* annehmen kann. Dabei drückt *außen* aus, ob die Anforderung von außen gekommen ist. Mehrere Anforderungen eines Fahrstuhls zu einer Etage werden grundsätzlich zu einer zusammengefasst, wobei eine zusammengesetzte Anforderung genau dann als von außen erfolgt angesehen wird, wenn mindestens eine ihrer Teilanforderungen von außen erfolgte.

Der interne Zustand eines Fahrstuhls entspricht damit genau dem Zustand eines Fahrstuhls in der Objektbasierten Simulation.

Abbildung 11 stellt den internen Zustand des Fahrstuhls als UML-Klassendiagramm dar.

Abbildung 11: UML-Klassendiagramm für den internen Zustand eines Fahrstuhls

Für die internen Startzustände ergibt sich

Fahrstuhl [1] . S_0 = { bevorrechtigt = true, vorgemerkt = false, wartend = false, etage = 1, bewZust = stehend, Anforderung = Ø }

und

Fahrstuhl [2] . S_0 = { bevorrechtigt = false, vorgemerkt = false, wartend = false, etage = 3, bewZust = stehend, Anforderung = Ø }.

Für die externen Startzustände ergibt sich

Fahrstuhl [1] . S_0 = { etage = 1, bewZust = stehend }

und

Fahrstuhl [2] . S_0 = { etage = 3, bewZust = stehend }.

Am Anfang ist Fahrstuhl 1 bevorrechtigt und steht in Etage 1, während Fahrstuhl 2 in Etage 3 steht, Anforderungen gibt es noch keine.

Gestartet wird zum Zeitpunkt $c_0 = 0$. Es gibt vier Stränge von exogenen Ereignissen, die jeweiligen Anforderungen an die Fahrstühle. Alle Stränge gehören zum Typ *Wird_angefordert*.

exogenes Ereignis	`Wird_angefordert ( 1, 1 ) @ t`
Periodizität	`Expo_ZV(` $\mu_{1,1}$ `)`
Stopbedingung	-

exogenes Ereignis	**Wird_angefordert (1, 2) @ t**
Periodizität	**Expo_ZV(** $\mu_{1,2}$ **)**
Stopbedingung	–

exogenes Ereignis	**Wird_angefordert (2, 2) @ t**
Periodizität	**Expo_ZV(** $\mu_{2,2}$ **)**
Stopbedingung	–

exogenes Ereignis	**Wird_angefordert (2, 3) @ t**
Periodizität	**Expo_ZV(** $\mu_{2,3}$ **)**
Stopbedingung	–

Ein explizites Simulationsende wird hier nicht erfasst, die Simulation läuft unendlich lange.

4.3.4.2 Ereignis Kommt_an

Das Ereignis *Kommt_an* stellt die Ankunft eines Fahrstuhls in einer Etage dar. Parameter sind die Nummer des Fahrstuhls und die Nummer der Etage, in der der Fahrstuhl ankommt.

Als Zusicherung kann gegeben werden, dass der Fahrstuhl sich vorher auf dem Weg zu dieser Etage befunden haben muss.

Ereignis	**Kommt_an (i , j) @ t**
Zusicherung	**context Fahrstuhl [i]** **etage <> j and bewZust = fahrend**

Zusicherung Ereignis_Kommt_an

Der Ort und der Bewegungszustand des Fahrstuhls werden aktualisiert. Der Fahrstuhl nimmt wahr, dass er in der Etage angekommen ist.

Ereignis	**Kommt_an (i, j) @ t**
Ereignisbedingung	–
Zustandsbedingung	–
Zustandseffekt	**context Fahrstuhl [i]** **etage = j and bewZust = stehend**
Folgeereignisse	ø
Wahrnehmungen	**{ Fahrstuhl [i] : { Kommt_an (j) } }**

Regel 50: Ereignis_Kommt_an.1

4.3.4.3 Wahrnehmung Kommt_an

Die Wahrnehmung *Kommt_an* stellt die Wahrnehmung eines Fahrstuhls seiner Ankunft in einer Etage dar. Parameter ist die Nummer der Etage, in der der Fahrstuhl ankommt.

Als Zusicherung kann gegeben werden, dass der Fahrstuhl sich vorher auf dem Weg zu dieser Etage befunden haben muss.

Ereignis	**Kommt_an (j)**
Zusicherung	**etage <> j and bewZust = fahrend**

Zusicherung Wahrnehmung_Kommt_an

In allen Fällen gilt, dass, wenn der Fahrstuhl in einer Etage ankommt, alle Anforderungen an diese Etage erfüllt sind. Ferner muss in allen Fällen im internen Zustand des Fahrstuhls der Ort und der Bewegungszustand des Fahrstuhls aktualisiert werden.

Kommt der Fahrstuhl in einer Etage an, zu der er von außen angefordert wurde, so will der Fahrstuhl demnächst (in c_1 Zeiteinheiten) in die andere Etage fahren, um die eingestiegenen Personen dorthin zu befördern. Die Anforderung von innen zur anderen Etage wird ergänzt. Existiert schon eine Anforderung zur anderen Etage, so will der Fahrstuhl ebenfalls demnächst dorthin fahren. In beiden Fällen wird der Wert von *außen* beibehalten, um keine evtl. bereits existierende Anforderung zu löschen. Regel *Wahrnehmung_Kommt_an.1a* (Äquivalent: Regel 30) deckt dabei den Fall ab, dass es bereits eine Anforderung von außen zur anderen Etage gab, während Regel *Wahrnehmung_Kommt_an.1b* (Äquivalent: Regel 31) den Fall abdeckt, dass es noch keine Anforderung von außen zur anderen Etage gab.

Ereignis	**Kommt_an (j)**
Ereignisbedingung	**–**
Zustandsbedingung	**let k: Integer = self.etage** **Anforderung -> exists (außen and etage = k)**
Zustandseffekt	**etage = j and bewZust = stehend and** **not (Anforderung -> exists (etage = j))**
Folge-Zeitereignisse	**{ Will_fahren (j) @ (c +** c_1 **) }**
Nachrichten	**Ø**
Aktionen	**Ø**

Regel 51: Wahrnehmung_Kommt_an.1a

Ereignis	**Kommt_an (j)**
Ereignisbedingung	**–**
Zustandsbedingung	**let k: Integer = self.etage** **(Anforderung -> exists ((außen and etage = j) or (etage = k)))**
Zustandseffekt	**let k: Integer = self.etage@pre** **etage = j and bewZust = stehend and** **not (Anforderung -> exists (etage = j)) and** **(Anforderung -> exists (not außen and etage = k))**
Folge-Zeitereignisse	**{ Will_fahren (j) @ (c +** c_1 **) }**
Nachrichten	**Ø**
Aktionen	**Ø**

Regel 52: Wahrnehmung_Kommt_an.1b

Kommt der Fahrstuhl in Etage 2 an und hat sich vorgemerkt, dass der andere Fahrstuhl auf eine Erlaubnis wartet, in Etage 2 fahren zu dürfen, so will der Fahrstuhl (spätestens) nach c_2 Zeiteinheiten anfahren, um Etage 2 für den anderen Fahrstuhl freizumachen. (Äquivalent: Regel 32)

Ereignis	**Kommt_an (j)**
Ereignisbedingung	**j = 2**
Zustandsbedingung	**vorgemerkt**
Zustandseffekt	**etage = 2 and bewZust = stehend and** **not (Anforderung -> exists (etage = 2))**
Folge-Zeitereignisse	**{ Will_fahren (2) @ (c +** c_2 **) }**
Nachrichten	**Ø**
Aktionen	**Ø**

Regel 53: Wahrnehmung_Kommt_an.3

Ansonsten werden nur die ggf. vorhandenen Anforderungen[51] als erfüllt markiert und der Ort und der Bewegungszustand werden aktualisiert. (Äquivalent: Regel 33)

[51] Dass ein Fahrstuhl in einer Etage ankommt, ohne dass eine Anforderung in diese Etage vorlag, ist möglich, wenn ein Fahrstuhl nur zurück in seine Heimatetage gefahren ist, um Etage 2 für den anderen Fahrstuhl frei zu machen.

Ereignis	**Kommt_an (j)**
Ereignisbedingung	–
Zustandsbedingung	–
Zustandseffekt	**etage = j and bewZust = stehend and not (Anforderung -> exists (etage = j))**
Folge-Zeitereignisse	Ø
Nachrichten	Ø
Aktionen	Ø

Regel 54: Wahrnehmung_Kommt_an.4

4.3.4.4 Ereignis Wird_angefordert

Das Ereignis *Wird_angefordert* stellt die Anforderung eines Fahrstuhls von außen dar. Parameter sind die Nummer des Fahrstuhls und die Nummer der Etage, von der aus der Fahrstuhl angefordert wird.

Der Fahrstuhl nimmt wahr, dass er von dieser Etage aus angefordert worden ist.

Ereignis	**Wird_angefordert (i, j) @ t**
Ereignisbedingung	–
Zustandsbedingung	–
Zustandseffekt	–
Folgeereignisse	Ø
Wahrnehmungen	**{ Fahrstuhl [i] : { Wird_angefordert (j) } }**

Regel 55: Ereignis_Wird_angefordert.1

4.3.4.5 Wahrnehmung Wird_angefordert

Das Ereignis *Wird_angefordert* stellt die Wahrnehmung eines Fahrstuhls seiner Anforderung von außen zu einer Etage dar. Parameter ist die Nummer der Etage, zu der der Fahrstuhl angefordert wird.

Als Zusicherung kann gegeben werden, dass, wenn der Fahrstuhl in einer Etage steht, dass dann zu dieser Etage keine Anforderungen vorhanden sein können.

Ereignis	**Wird_angefordert (j)**
Zusicherung	**bewZust = stehend implies not (Anforderung -> exists (etage = self.etage))**

Zusicherung Wahrnehmung_Wird_angefordert

Steht der Fahrstuhl gerade in der angeforderten Etage (dann kann gemäß Zusicherung vorher noch keine Anforderung zu dieser Etage vorhanden gewesen sein), so ist der erste Teil der Anforderung sofort erfüllt. Deshalb will der Fahrstuhl demnächst (in c_1 Zeiteinheiten) in die andere Etage fahren, um die eingestiegenen Personen dorthin zu befördern. Die Anforderung von innen zur anderen Etage wird ergänzt. Regel *Wahrnehmung_Wird_angefordert.1a* (Äquivalent: Regel 34) deckt dabei den Fall ab, dass es bereits eine Anforderung von außen zur anderen Etage gab, während Regel *Wahrnehmung_Wird_angefordert.1b* (Äquivalent: Regel 35) den Fall abdeckt, dass es noch keine Anforderung von außen zur anderen Etage gab.

Ereignis	**Wird_angefordert (j)**
Ereignisbedingung	–
Zustandsbedingung	**let k: Integer = andereEtage (j)** **etage = j and bewZust = stehend and** **Anforderung -> exists (außen and etage = k)**
Zustandseffekt	–
Folge-Zeitereignisse	**{ Will_fahren (j) @ (c +** c_1 **) }**
Nachrichten	Ø
Aktionen	Ø

Regel 56: Wahrnehmung_Wird_angefordert.1a

Ereignis	**Wird_angefordert (j)**
Ereignisbedingung	–
Zustandsbedingung	**etage = j and bewZust = stehend**
Zustandseffekt	**let k: Integer = andereEtage (j)** **Anforderung -> exists (not außen and etage = k))**
Folge-Zeitereignisse	**{ Will_fahren (j) @ (c +** c_1 **) }**
Nachrichten	Ø
Aktionen	Ø

Regel 57: Wahrnehmung_Wird_angefordert.1b

Sollte vorher noch keine Anforderung nach Etage j vorhanden sein und steht der Fahrstuhl gerade in der anderen Etage, so wird die Anforderung in den Zustand aufgenommen und der Fahrstuhl will außerdem demnächst (in c_1 Zeiteinheiten) losfahren. (Äquivalent: Regel 36)

Ereignis	**Wird_angefordert (j)**
Ereignisbedingung	–
Zustandsbedingung	**(Anforderung -> isEmpty()) and** **etage <> j and bewZust = stehend**
Zustandseffekt	**Anforderung -> exists (außen and etage = j)**
Folge-Zeitereignisse	**{ Will_fahren (etage) @ (c +** c_1 **) }**
Nachrichten	Ø
Aktionen	Ø

Regel 58: Wahrnehmung_Wird_angefordert.2

Ansonsten wird nur die Anforderung in den Zustand aufgenommen (sofern nicht bereits vorhanden). (Äquivalent: Regel 37)

Ereignis	**Wird_angefordert (j)**
Ereignisbedingung	–
Zustandsbedingung	–
Zustandseffekt	**(Anforderung -> exists (außen and etage = j)**
Folge-Zeitereignisse	Ø
Nachrichten	Ø
Aktionen	Ø

Regel 59: Wahrnehmung_Wird_angefordert.3

4.3.4.6 Zeitereignis Will_fahren

Das Zeitereignis *Will_fahren* stellt das Entstehen eines Anfahrwunsches eines Fahrstuhls dar. Parameter ist die Nummer der Etage, von der aus der Fahrstuhl anfahren will.

Als Zusicherung kann gegeben werden, dass der Fahrstuhl zu diesem Zeitpunkt noch nicht auf eine Erlaubnis wartet.

Ereignis	**Will_fahren (j)**
Zusicherung	**not wartend**

Zusicherung Zeitereignis_Will_fahren

Befindet sich der Fahrstuhl in Etage 2 und will von dort wegfahren, so kann er unmittelbar losfahren. Hat er sich vorgemerkt, dass der andere Fahrstuhl auf eine Erlaubnis für das Fahren in Etage 2 wartet, so erteilt er ihm diese in diesem Moment. Er braucht sich nun nicht mehr vorzumerken, dass der andere Fahrstuhl auf eine Erlaubnis von ihm wartet. Es wird in den Bewegungszustand *fahrend* gewechselt und als Aktion das Anfahren durchgeführt.

Ereignis	**Will_fahren (j)**
Ereignisbedingung	**j = 2**
Zustandsbedingung	**etage = 2 and bewZust = stehend and vorgemerkt**
Zustandseffekt	**not vorgemerkt and bewZust = fahrend**
Folge-Zeitereignisse	Ø
Nachrichten	**{ Erlaubnis ( ) → Fahrstuhl [andererFahrstuhl( )] }**
Aktionen	**{ Fährt_an ( ) }**

Regel 60: Zeitereignis_Will_fahren.1a

Befindet er sich in Etage 2 und hat sich nicht vorgemerkt, dass der andere Fahrstuhl auf eine Erlaubnis für das Fahren in Etage 2 wartet, so wird nur in den Bewegungszustand *fahrend* gewechselt und als Aktion das Anfahren durchgeführt.

Ereignis	**Will_fahren (j)**
Ereignisbedingung	**j = 2**
Zustandsbedingung	**etage = 2 and bewZust = stehend**
Zustandseffekt	**bewZust = fahrend**
Folge-Zeitereignisse	**Ø**
Nachrichten	**Ø**
Aktionen	**{ Fährt_an ( ) }**

Regel 61: Zeitereignis_Will_fahren.1b

Die Regeln *Zeitereignis_Will_fahren.1a* und *Zeitereignis_Will_fahren.1b* sind gemeinsam äquivalent zur Regel *Will_fahren.1* (Regel 38) aus Abschnitt 3.4.4.4, wobei Regel *Zeitereignis_Will_fahren.1a* den Fall abdeckt, dass sich der Fahrstuhl vorgemerkt hat, dass der andere Fahrstuhl auf eine Erlaubnis wartet und Regel *Zeitereignis_Will_fahren.1b* den Fall, dass er sich dies nicht vorgemerkt hat. Die Fallunterscheidung ist notwendig geworden, da das Verhalten aus den Reaktionsregeln zum Ereignis *Fährt_an* aus der Objektbasierten Simulation in der Agentenbasierten Simulation in die Regeln vorgezogen werden muss, in denen die Aktion *Fährt_an* durchgeführt wird.

Will der Fahrstuhl in Etage 2 fahren, so muss er den anderen Fahrstuhl um Erlaubnis fragen. Somit erhält der andere Fahrstuhl eine Erlaubnisanfrage. Der Fahrstuhl wartet nun auf die Erlaubnis. (Äquivalent: Regel 39)

Ereignis	**Will_fahren (j)**
Ereignisbedingung	**j <> 2**
Zustandsbedingung	**etage = j and bewZust = stehend**
Zustandseffekt	**wartend**
Folge-Zeitereignisse	**Ø**
Nachrichten	**{ Erlaubnisanfrage ( ) → Fahrstuhl [andererFahrstuhl( )] }**
Aktionen	**Ø**

Regel 62: Zeitereignis_Will_fahren.2

Befindet sich der Fahrstuhl nicht mehr in der Etage, von der aus er losfahren will, so ist dieses Ereignis irrelevant. Der Zustand ändert sich nicht, es werden keine Folge-Zeitereignisse erzeugt und keine Aktionen durchgeführt. Dieser Fall kann eintreten, wenn sowohl gemäß einer der Regeln *Wahrnehmung_Kommt_an.1a* (Regel 51), *Wahrnehmung_Kommt_an.1b* (Regel 52) , *Wahrnehmung_Wird_angefordert.1a* (Regel 56) , *Wahrnehmung_Wird_angefordert.1b* (Regel 57) oder *Wahrnehmung_Wird_angefordert.2* (Regel 58) ein Folge-Zeitereignis *Will_fahren* erzeugt wurde, um Personen in die andere Etage zu befördern als auch gemäß einer der Regeln *Wahrnehmung_Kommt_an.3* (Regel 53) oder *Nachricht_Erlaubnisanfrage.2* (Regel 66) ein Folge-Zeitereignis *Will_fahren*, um Etage 2 für den anderen Fahrstuhl freizumachen. (Äquivalent: Regel 40)

Ereignis	`Will_fahren ( j )`
Ereignisbedingung	–
Zustandsbedingung	–
Zustandseffekt	–
Folge-Zeitereignisse	ø
Nachrichten	ø
Aktionen	ø

Regel 63: Zeitereignis_Will_fahren.3

4.3.4.7 Aktion Fährt_an

Die Aktion *Fährt_an* stellt das Anfahren eines Fahrstuhls dar.

Als Zusicherung kann gegeben werden, dass der Fahrstuhl steht.

Ereignis	`Fahrstuhl [ i ] : Fährt_an ( ) @ t`

Zusicherung	`context Fahrstuhl [ i ]`
	
	`bewZust = stehend`

Zusicherung Aktion_Fährt_an

Der Fahrstuhl wechselt in den Bewegungszustand *fahrend* und es wird ein Folgeereignis erzeugt, die Ankunft des Fahrstuhls auf der anderen Etage. (Äquivalent: Regel 41 und Regel 42)

Ereignis	`Fahrstuhl [ i ] : Fährt_an ( ) @ t`
Ereignisbedingung	–
Zustandsbedingung	–

Zustandseffekt	`context Fahrstuhl [ i ]`
	
	`bewZust = fahrend`

Folgeereignisse	`{ Kommt_an ( i, andereEtage ( i, Fahrstuhl [ i ].etage ) )` `@ ( t + Gleich_ZV ( a, b ) ) }`
Wahrnehmungen	ø

Regel 64: Aktion_Fährt_an.1

Anders als in den Reaktionsregeln zum Ereignis *Fährt_an* aus der Objektbasierten Simulation kann in der Agentenbasierten Simulation in den Reaktionsregeln zur Aktion *Fährt_an* kein Verhalten der Fahrstühle spezifiziert werden, da es sich um Reaktionsregeln für den Umgebungssimulator handelt. Das Verhalten der Agenten muss deshalb in die Reaktionsregeln vorgezogen werden, in denen die Aktion *Fährt_an* durchgeführt wird. Dies sind die Regeln Zeitereignis_Will_fahren.1a (Regel 60), Zeitereignis_Will_fahren.1b (Regel 61) und Nachricht_Erlaubnis.1 (Regel 71).

4.3.4.8 Nachricht Erlaubnisanfrage

Die Nachricht *Erlaubnisanfrage* stellt dar, dass ein Fahrstuhl dem anderen Fahrstuhl eine Erlaubnisanfrage für das Fahren in Etage 2 erteilt.

Als Zusicherung für den Empfang der Nachricht kann gegeben werden, dass der Empfänger der Nachricht sich vorher noch nicht vorgemerkt hat, dass der andere Fahrstuhl auf eine Erlaubnis von ihm wartet.

Ereignis	**Empfang(Erlaubnisanfrage ( ) ← Fahrstuhl [k])**
Zusicherung	**not vorgemerkt**

Zusicherung Nachricht_Erlaubnisanfrage

Befindet sich der Fahrstuhl, der die Anfrage erhält, auf dem Weg von Etage 2 weg, so kann er bedenkenlos eine Erlaubnis erteilen. (Äquivalent: Regel 43)

Ereignis	**Empfang(Erlaubnisanfrage ( ) ← Fahrstuhl [k])**
Ereignisbedingung	**-**
Zustandsbedingung	**etage = 2 and bewZust = fahrend**
Zustandseffekt	**-**
Folge-Zeitereignisse	**Ø**
Nachrichten	**{ Erlaubnis ( ) → Fahrstuhl [k] }**
Aktionen	**Ø**

Regel 65: Nachricht_Erlaubnisanfrage.1

Befindet sich der Fahrstuhl, der die Anfrage erhält, in Etage 2, so kann er dem anderen Fahrstuhl im Moment keine Erlaubnis erteilen, merkt sich aber vor, dass er ihm eine Erlaubnis erteilt, sobald er Etage 2 verlässt. Außerdem wird er (spätestens) in c_2 Zeiteinheiten anfahren, um dem anderen Fahrstuhl dann eine Fahrt in Etage 2 zu ermöglichen. (Äquivalent: Regel 44)

Ereignis	**Empfang(Erlaubnisanfrage ( ) ← Fahrstuhl [k])**
Ereignisbedingung	**-**
Zustandsbedingung	**etage = 2**
Zustandseffekt	**vorgemerkt**
Folge-Zeitereignisse	**Will_fahren (2) @ (c + c_2)**
Nachrichten	**Ø**
Aktionen	**Ø**

Regel 66: Nachricht_Erlaubnisanfrage.2

Befindet sich der Fahrstuhl, der die Anfrage erhält, gerade auf dem Weg zu Etage 2 hin, so kann er dem anderen Fahrstuhl im Moment keine Erlaubnis erteilen, merkt sich aber vor, dass er ihm eine Erlaubnis erteilt, sobald er Etage 2 wieder verlässt. (Äquivalent: Regel 45)

Ereignis	**Empfang(Erlaubnisanfrage ( ) ← Fahrstuhl [k])**
Ereignisbedingung	**-**
Zustandsbedingung	**fahrend**
Zustandseffekt	**vorgemerkt**
Folge-Zeitereignisse	**Ø**
Nachrichten	**Ø**
Aktionen	**Ø**

Regel 67: Nachricht_Erlaubnisanfrage.3

Befindet er sich in seiner Heimatetage und hat selbst noch keine Erlaubnisanfrage gestellt (wartet also nicht seinerseits auf eine Erlaubnis des anderen Fahrstuhls), so erteilt er dem anderen die Erlaubnis. (Äquivalent: Regel 46)

Ereignis	**Empfang(Erlaubnisanfrage ( ) ← Fahrstuhl [k])**
Ereignisbedingung	**-**
Zustandsbedingung	**not wartend**
Zustandseffekt	**-**
Folge-Zeitereignisse	**Ø**
Nachrichten	**{ Erlaubnis ( ) → Fahrstuhl [k] }**
Aktionen	**Ø**

Regel 68: Nachricht_Erlaubnisanfrage.4

Befindet er sich in seiner Heimatetage und hat ebenfalls schon eine Erlaubnisanfrage gestellt, so haben sich also die Erlaubnisanfragen der beiden Fahrstühle gekreuzt. Demnach ist es entscheidend, welcher Fahrstuhl der Bevorrechtigte ist. Ist der Empfänger der Nachricht der Bevorrechtigte, so erteilt er dem anderen Fahrstuhl keine Erlaubnis, merkt sich aber vor, dass er die Erlaubnis erteilt, sobald er dann später irgendwann Etage 2 erreicht und auch wieder verlassen hat. Dadurch wird der andere Fahrstuhl der Bevorrechtigte. (Äquivalent: Regel 47)

Ereignis	**Empfang(Erlaubnisanfrage ( ) ← Fahrstuhl [k])**
Ereignisbedingung	**-**
Zustandsbedingung	**bevorrechtigt**
Zustandseffekt	**not bevorrechtigt and vorgemerkt**
Folge-Zeitereignisse	**Ø**
Nachrichten	**Ø**
Aktionen	**Ø**

Regel 69: Nachricht_Erlaubnisanfrage.5

Ist hingegen der Empfänger der Nachricht nicht der Bevorrechtigte, so erteilt er dem anderen Fahrstuhl die Erlaubnis. Dadurch wird er selbst der Bevorrechtigte. (Äquivalent: Regel 48)

Ereignis	**Empfang(Erlaubnisanfrage ( ) ← Fahrstuhl [k])**
Ereignisbedingung	-
Zustandsbedingung	-
Zustandseffekt	**bevorrechtigt**
Folge-Zeitereignisse	ø
Nachrichten	**{ Erlaubnis ( ) → Fahrstuhl [k] }**
Aktionen	ø

Regel 70: Nachricht_Erlaubnisanfrage.6

4.3.4.9 Nachricht Erlaubnis

Die Nachricht *Erlaubnis* stellt dar, dass ein Fahrstuhl dem anderen Fahrstuhl eine Erlaubnis für das Fahren in Etage 2 erteilt.

Als Zusicherung für den Empfang der Nachricht kann gegeben werden, dass der Empfänger der Nachricht in seiner Heimatetage steht und auf eine Erlaubnis wartet.

Ereignis	**Empfang(Erlaubnis ( ) ← Fahrstuhl [k])**
Zusicherung	**etage <> 2 and bewZust = stehend and wartend**

Zusicherung Nachricht_Erlaubnis

Der Empfänger der Nachricht ändert seinen Zustand dahingehend, dass er jetzt nicht mehr auf die Erlaubnis wartet. Es wird in den Bewegungszustand *fahrend* gewechselt und als Aktion das Anfahren durchgeführt. (Äquivalent: Regel 49)

Ereignis	**Empfang(Erlaubnis ( ) ← Fahrstuhl [k])**
Ereignisbedingung	-
Zustandsbedingung	-
Zustandseffekt	**not wartend and bewZust = fahrend**
Folge-Zeitereignisse	ø
Nachrichten	ø
Aktionen	**{ Fährt_an ( ) }**

Regel 71: Nachricht_Erlaubnis.1