6. Agentenbasierte Simulation von Fahrerlosen Transportsystemen

In diesem Kapitel werden die aus Kapitel 5 bekannten Fahrerlosen Transportsysteme benutzt, um ein Beispiel für Agentenbasierte Simulation darzustellen. Es enthält die Szenariobeschreibung in Abschnitt 6.1, die verwendeten Agenten-, Objekt- und Nachrichtentypen in Abschnitt 6.2 sowie das Externe Modell in Abschnitt 6.3, in dem auch die Wahrnehmungs- und Aktionstypen der Agenten vorgestellt werden. Die internen Modelle der verwendeten Agententypen befinden sich in den Abschnitten 6.4 - 6.7. Dabei enthält jedes Kapitel eine Kurzbeschreibung und den Zustand der Agententypen.

Anhang B enthält ferner die Reaktionsregeln, die das Verhalten der Umgebung und der Agenten beschreiben. Des Weiteren werden dort mögliche Erweiterungen, die das Szenario exakter und damit realistischer beschreiben, jeweils natürlichsprachlich angegeben.

6.1 Szenariobeschreibung

Das Szenario besteht aus mehreren FTF, einem Fahrkurs, bestehend aus Knoten und unidirektionalen Segmenten, aus mehreren Maschinen sowie aus verschiedenen Softwarekomponenten, die die Rolle von Transportauftragsmanagern (TA-Manager) und von Verkehrsreglern übernehmen. TA-Manager generieren TA und ordnen sie den sich bewerbenden FTF zu. Verkehrsregler dienen der Vermeidung von Kollisionen zwischen mehreren FTF auf Knoten.[63]

Die am System beteiligten FTF sind alle Single-Load-Carrier, können also zu jedem Zeitpunkt nur ein TS transportieren. Die Geschwindigkeit des i-ten FTF beträgt v_i. Um einen Knoten zu überfahren, brauchen alle FTF bei allen Knoten c_1 Zeiteinheiten, bis sie den Knoten wieder verlassen haben. Um in einen Knoten einzufahren, brauchen alle FTF bei allen Knoten c_2 Zeiteinheiten, bis sie in den Knoten eingefahren und stehengeblieben sind. Um aus einem Knoten, in dem sie stehen, auszufahren, brauchen alle FTF bei allen Knoten c_3 Zeiteinheiten, bis sie den Knoten wieder verlassen haben.[64]

Das System hat ein Eingangslager, in das noch unbearbeitete TS von außen abgelegt werden. Hierbei kann man davon ausgehen, dass von außen sichergestellt wird, dass immer genügend TS sich im Eingangslager befinden. Die TS durchlaufen eine Folge von Maschinen, die jeweils einen der n benötigten Arbeitsschritte ausführen. Für die Durchführung eines Arbeitsschritts braucht die i-te Maschine eine Zeit, die gleichverteilt im Intervall [a_i, b_i] liegt. Aus Gründen der Einfachheit wird hier davon ausgegangen, dass jede Maschine nur einen festen der n Arbeitsschritte ausführt und sich dieser im Laufe der Zeit auch nicht ändert. Nach Erledigung aller n Arbeitsschritte werden die TS in das Ausgangslager des Systems transportiert. Der Zustand eines TS wird durch eine natürliche Zahl kodiert, die ausdrückt, welcher Arbeitsschritt bereits erfolgt ist. TS kommen also im Zustand 0 ins Eingangslager und verlassen das System im Ausgangslager im Zustand n.

6.1.1 Erweiterungen

Eine denkbare Erweiterung des Szenarios wäre, dass es möglich ist, dass eine Maschine nicht nur auf einen Arbeitsschritt spezialisiert ist, sondern mehrere Arbeitsschritte für sie möglich ist. Für jede Maschine könnte es also eine beliebige nichtleere Teilmenge der Arbeitsschritte geben, die sie auszuführen imstande ist.

Eine weitere Erweiterung wäre, mehrere Eingangs- und Ausgangslager im System zuzulassen.

---

[63] Hierbei wird davon ausgegangen, dass nur auf Knoten Kollisionen durch Verkehrsregler verhindert werden müssen. Innerhalb eines Segments können FTF automatisch durch Sensoren verhindern, dass ein FTF auf ein anderes auffährt.

[64] Klarerweise ist $c_1 \leq c_2 + c_3$, denn wäre das nicht so, könnte ein FTF einen Knoten alleine dadurch schneller passieren, dass es in der Mitte der Kreuzung anhält.

Bei den FTF könnte man neben Single-Load-Carrier auch Multiple-Load-Carrier, die mehrere TS gleichzeitig transportieren können, zulassen.

Neben unidirektionalen Segmenten könnten auch bidirektionale Segmente zugelassen werden, bei denen dann ebenfalls Verkehrsregler eingesetzt werden, um Kollisionen zwischen entgegenkommenden FTF zu vermeiden.

6.2 Agenten-, Objekt- und Nachrichtentypen

Es gibt vier Agententypen: *Maschine*, *TA-Manager*, *FTF* und *Verkehrsregler*. Daneben gibt es drei Objekttypen: *Ort*, *TA* und *TS*, wobei es für *Ort* die Untertypen *Knoten* und *Segment* gibt.

Alle Nachrichtentypen werden auf dem Wege elektronischer Nachrichten übertragen, dem einzigen verwendeten Nachrichtenkanal.

Nachrichtenkanal	**elektronische Nachricht**
Übertragungsdauer	Δ
Zuverlässigkeit	1

Es gibt zwischen einem TA-Manager und einem FTF sechs verschiedene Nachrichtentypen, die allesamt die Auftragsvergabe betreffen. Der Nachrichtentyp *cfp* stellt die Ausschreibung eines TA dar, *propose* eine Bewerbung für einen TA, *accept_proposal* den Zuschlag für einen TA, *reject_proposal* die Ablehnung einer Bewerbung für einen TA, *accept_order* die Annahme des Zuschlags und *reject_order* die Ablehnung des Zuschlags.

Sender	Empfänger
TAManager	**FTF**
Nachricht	Nachrichtenkanal
cfp (TA, startort, zielort)	**elektronische Nachricht**
accept_proposal (TA, startort, zielort)	**elektronische Nachricht**
reject_proposal (TA)	**elektronische Nachricht**

Sender	Empfänger
FTF	**TAManager**
Nachricht	Nachrichtenkanal
propose (TA, t)	**elektronische Nachricht**
accept_order (TA)	**elektronische Nachricht**
reject_order (TA)	**elektronische Nachricht**

Zwischen einem TA-Manager und einer Maschine gibt es zehn verschiedene Nachrichtentypen. Der Nachrichtentyp *erteilenAA* stellt die Erteilung eines Abholauftrags und *erteilenBA* die Erteilung eines Beschaffungsauftrags dar. Die Nachricht *anfordernAnnahmeTS* ist die Anforderung des TA-Managers an die Maschine, ein TS aufzunehmen, *bestaetigenAnnahme* die Bestätigung dieser Anforderung und *ablehnenAnnahme* die Ablehnung dieser Anforderung durch die Maschine. Spiegelbildlich dazu ist *anfordernAbgabeTS* die Anforderung des TA-Managers an die Maschine, ein TS abzugeben, *bestaetigenAbgabe* die Bestätigung dieser Anforderung und *ablehnenAbgabe* die Ablehnung dieser Anforderung durch die Maschine. Storniert der TA-Manager seine Anforderung, so tut er das mittels einer Nachricht *stornierenAnnahme* bzw. *stornierenAbgabe*. Parameter bei den Nachrichten betreffend Annahme- und Abgabeanforderungen ist der AA bzw. BA, der durch diese Anforderung ergänzt werden soll.

Sender	Empfänger
TAManager	**Maschine**
Nachricht	Nachrichtenkanal
anfordernAnnahmeTS (AA)	**elektronische Nachricht**
anfordernAbgabeTS (BA)	**elektronische Nachricht**
stornierenAnnahme (AA)	**elektronische Nachricht**
stornierenAbgabe (BA)	**elektronische Nachricht**

Sender	Empfänger
Maschine	**TAManager**
Nachricht	Nachrichtenkanal
erteilenAA ( )	**elektronische Nachricht**
erteilenBA ( )	**elektronische Nachricht**
bestaetigenAnnahme (AA)	**elektronische Nachricht**
ablehnenAnnahme (AA)	**elektronische Nachricht**
bestaetigenAbgabe (BA)	**elektronische Nachricht**
ablehnenAbgabe (BA)	**elektronische Nachricht**

Zwischen einem FTF und einem Verkehrsregler gibt es drei verschiedene Nachrichtentypen, die allesamt der Kollisionsvermeidung beim Einfahren in Knoten dienen. Der Nachrichtentyp *Erlaubnisanfrage* stellt die Anfrage eines FTF an einen Verkehrsregler dar, ihm die Erlaubnis für das Einfahren in einen Knoten zu erteilen, während *Erlaubnis* die entsprechende Erlaubnis des Verkehrsreglers darstellt und *Knotenfreigabe* die Nachricht des FTF an den Verkehrsregler ist, dass der Knoten nun wieder verlassen wurde.

Sender	Empfänger
FTF	**Verkehrsregler**
Nachricht	Nachrichtenkanal
Erlaubnisanfrage ( )	**elektronische Nachricht**
Knotenfreigabe ( )	**elektronische Nachricht**

Sender	Empfänger
Verkehrsregler	**FTF**
Nachricht	Nachrichtenkanal
Erlaubnis ( )	**elektronische Nachricht**

6.3 Externes Modell (Umgebungssimulator)

Für den Umgebungssimulator sind die externen Zustände der Agenten sowie die Zustände der Objekte relevant.

Es gibt vier Agententypen: *Maschine*, *TA-Manager*, *FTF* und *Verkehrsregler*. Daneben gibt es drei Objekttypen: *Ort*, *TA* und *TS*, wobei es für *Ort* die Untertypen *Knoten* und *Segment* gibt, die sich anhand der Variable *typ* abfragen lassen.

Orte mit Untertyp *Knoten* haben einen Verkehrsregler in der Variable *verkehrsregler* und optional eine Maschine in der Variable *maschine*. Orte mit Untertyp *Segment* haben einen Startknoten und einen Zielknoten in den Variablen *startknoten* bzw. *zielknoten*.

Da der *TA-Manager* und der *Verkehrsregler* reine Softwarekomponenten sind, haben sie keinen externen (physikalischen) Zustand und können außer eingehenden Nachrichten auch nichts wahrnehmen und außer dem Versand von Nachrichten auch keine Aktionen durchführen.

Es gibt vier Typen von Umgebungsereignissen: *ProduktionFertig*, *KnotenPassiert*, *KnotenErreicht* und *SegmentendeErreicht*. Dabei stellt *ProduktionFertig* dar, dass bei einer Maschine die Produktion des gerade bearbeiteten TS fertig ist. *KnotenPassiert* stellt dabei dar, dass ein FTF nun den befahrenen Knoten passiert und somit das als nächstes anzufahrende Segment erreicht hat. *KnotenErreicht* stellt dar, dass ein FTF einen Knoten, auf dem es stehen bleiben möchte, erreicht hat. *SegmentendeErreicht* stellt dar, dass ein FTF das Ende des Segments erreicht hat.

6.3.1 Maschine

Der externe Zustand einer Maschine beinhaltet jeweils die Kapazität und der aktuellen Anzahl der TS des Eingangspuffers und des Ausgangspuffers, KAP_EIN, ANZ_EIN, KAP_AUS und ANZ_AUS. Dabei ändern sich KAP_EIN und KAP_AUS im Laufe der Zeit nicht. Ferner gehört zum Zustand der Maschine das Attribut *schritt*, das ausdrückt, welchen Arbeitsschritt die Maschine ausführt sowie das Attribut *ort*, den ort, an dem sich die Maschine befindet. Die Attribute *schritt* und *ort* ändern sich ebenfalls im Laufe der Zeit nicht.

Die Aktionen, die eine Maschine durchführen kann sind *aufnehmenTS* und *ablegenTS*. Dabei stellt *aufnehmenTS* das Aufnehmen eines TS aus dem Eingangspuffer mit anschließendem Beginn der Verarbeitung dieses TS dar. *ablegenTS* stellt das Ablegen eines fertig produzierten TS in den Ausgangspuffer dar.

Die Wahrnehmungen, die eine Maschine haben kann sind *ProduktionFertig*, *TSkommtAn* und *TSwirdAbgeholt*. Dabei stellt *ProduktionFertig* dar, dass die Maschine wahrnimmt, dass die Produktion des gerade bearbeiteten TS fertig ist, *TSkommtAn*, dass ein TS durch ein FTF in den Eingangspuffer abgelegt worden ist und *TSwirdAbgeholt*, dass ein TS durch ein FTF aus dem Ausgangspuffer entnommen worden ist.

6.3.2 FTF

Der externe Zustand eines FTF beinhaltet seinen Bewegungszustand *bewZust* (kann die Werte *stehend* oder *fahrend* annehmen) und seinen aktuellen Ort *ort* (das kann ein Segment oder ein Knoten sein).

Ein FTF kann also auf einem Segment stehen. Das ist dann der Fall, wenn es das Segment komplett zu Ende gefahren ist und nun vor einem Knoten steht und darauf wartet, in ihn einfahren zu können.

Ein FTF kann auf einem Segment fahren. Ein FTF befährt ein Segment, in das es eingefahren ist, immer bis zu seinem Ende.

Ein FTF kann auf einem Knoten stehen. Das ist nur dann der Fall, wenn das FTF gerade ein TS an einer Maschine ablegt oder aufnimmt oder, wenn es gerade ein TS abgelegt und seinen TA damit erfüllt hat und nun noch darauf wartet, einen neuen TA zu übernehmen.

Ein FTF kann auf einem Knoten fahren. Ein FTF fährt in diesem Fall direkt in das Segment, das als nächstes angefahren werden soll.

Zum externen Zustand eines FTF gehört auch sein Beladungszustand *belZust*, der die Werte *beladen* oder *unbeladen* annehmen kann. Da hier nur Single-Load-Carrier betrachtet werden, reicht die Art der Zustandscodierung aus.

Die Wahrnehmungen, die ein FTF erhalten kann, sind *KnotenPassiert*, *KnotenErreicht* und *SegmentendeErreicht*. *KnotenPassiert* stellt dabei dar, dass das FTF wahrnimmt, dass es nun den befahrenen Knoten passiert und somit das als nächstes anzufahrende Segment erreicht hat. *KnotenErreicht* stellt dar, dass das FTF einen Knoten, auf dem es stehen bleiben möchte, erreicht hat. *SegmentendeErreicht* stellt dar, dass das FTF wahrnimmt, dass es das Ende des Segments erreicht hat.

Die Aktionen, die ein FTF durchführen kann sind *aufnehmenTS*, *ablegenTS*, *abfahrenKnotenUndSegment*, *befahrenKnoten* und *abfahrenSegment*. Dabei stellt *aufnehmenTS* das Aufnehmen eines TS aus dem Ausgangspuffer einer Maschine oder dem Eingangslager des Systems und *ablegenTS* das Ablegen eines TS in den Eingangspuffer einer Maschine oder das Ausgangslager des Systems dar.

Die Aktion *abfahrenKnotenUndSegment*[65] stellt das Überfahren eines Knotens mit anschließendem kompletten Abfahren eines Segments bis zum Ende dar. Dabei wird davon abstrahiert, dass das FTF auf seinem Weg vor Kurven abbremsen, Kurven fahren, nach Kurven wieder beschleunigen und am Ende des Segments anhalten muss. In Abbildung 29 ist dabei der Weg, den das FTF im Laufe dieser Aktion ausführt, in Form eines Pfeils dargestellt. Die Stellen, an denen das FTF eine Wahrnehmung erhält sind mit einem mit dem Namen der Wahrnehmung beschrifteten Pfeil versehen.

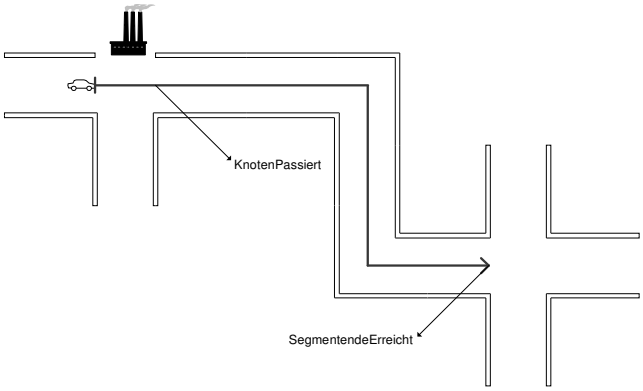

Abbildung 29: Aktion abfahrenKnotenUndSegment

Die Aktion *befahrenKnoten* stellt das Einfahren in einen Knoten und anschließende Stehenbleiben im Knoten dar. In Abbildung 30 ist diese Aktion dargestellt.

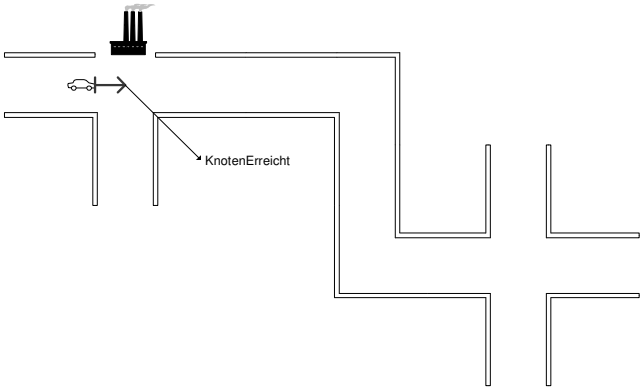

Abbildung 30: Aktion befahrenKnoten

Die Aktion *abfahrenSegment* stellt das komplette Abfahren eines Segments bis zum Ende dar, nachdem ein FTF auf einem Knoten gestanden hat. In Abbildung 31 ist diese Aktion dargestellt.

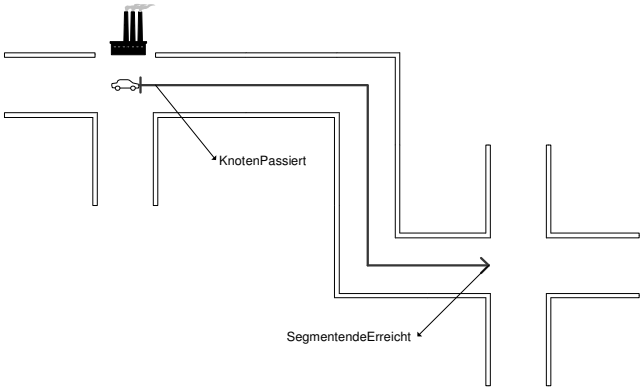

Abbildung 31: Aktion abfahrenSegment

[65] Dieser Name suggeriert, dass der komplette Prozess des Abfahrens des Knotens und des Segmentes als Aktion angesehen wird. Tatsächlich ist aber nur das Anfahren als Aktion zu modellieren. Ein genauerer Name für die Aktion wäre demnach z.B. *anfahrenZumAbfahrenVonKnotenUndSegment*.

6.4 Internes Modell Maschine

Jede Maschine hat einen fest zugeordneten TA-Manager, dem sie Beschaffungsaufträge (BA) und Abholaufträge (AA) erteilen kann. Sie hat einen Eingangspuffer mit Kapazität *KAP_EIN*. Überschreitet der Inhalt des Eingangspuffers die Schwelle *C_EIN* nicht (im einfachsten Falle ist C_EIN = 0), so erteilt die Maschine dem TA-Manager einen BA. Sie hat ferner einen Ausgangspuffer mit Kapazität *KAP_AUS*. Unterschreitet der Inhalt des Ausgangspuffers die Schwelle *C_AUS* nicht (im einfachsten Falle ist C_AUS = KAP_AUS), so erteilt die Maschine dem TA-Manager einen AA.

Der interne Zustand einer Maschine beinhaltet wie schon der externe Zustand jeweils die Kapazität und die aktuelle Anzahl der TS des Eingangspuffers und des Ausgangspuffers, KAP_EIN, ANZ_EIN, KAP_AUS und ANZ_AUS, wobei die Informationen der Maschine bezüglich dieser Attribute stets korrekt sind, also mit ihrem tatsächlichen Zustand, dem externen Zustand übereinstimmen.

Der interne Zustand einer Maschine enthält außerdem noch die Schwellen C_EIN und C_AUS sowie die Werte *bestellteTS* und *abzuholendeTS*, das ist die Anzahl der TS, deren Ankunft bzw. deren Abholung bereits durch Vereinbarung mit dem *TAManager* feststeht, aber noch nicht erfolgt ist. Die Vereinbarung kann durch ein *erteilenBA* bzw. *erteilenAA* seitens der Maschine initiiert worden sein, aber auch durch ein *anfordernAnnahmeTS* bzw. *anfordernAbgabeTS* seitens des Managers. Bei der Überprüfung, ob ein BA erteilt wird, muss der aktuelle Bestand des Eingangspuffers um die bestellten TS korrigiert werden, wenn man ihn mit dem Schwellenwert C_EIN vergleicht, da sonst in der Zeit, in der das bestellte TS noch nicht eingetroffen ist, immer wieder ein neuer BA erteilt wird. Spiegelbildlich muss bei der Überprüfung, ob ein AA erteilt wird, der aktuelle Bestand des Ausgangspuffers um die abzuholenden TS korrigiert werden, wenn man ihn mit dem Schwellenwert C_AUS vergleicht. Die Erteilung eines BA wird immer dann überprüft, wenn die Maschine ein TS aus dem Eingangspuffer nimmt, denn nur dann kann der Schwellenwert erreicht werden. Die Erteilung eines AA wird immer überprüft, wenn die Maschine ein TS in den Ausgangspuffer legt.

Eine weitere Komponente des internen Zustands ist der Produktionszustand *prodZust*. Als mögliche Produktionszustände kommen *leerlauf*, *produzierend* und *blockiert* in Frage. Eine Maschine befindet sich im Produktionszustand *leerlauf*, wenn sie nicht produzieren kann, weil sie kein zu bearbeitendes TS mehr vorrätig hat. Wenn sie gerade ein TS produziert, ist sie im Produktionszustand *produzierend*. Hat sie ein TS fertig produziert und kann es nicht ablegen, weil der Ausgangspuffer voll ist, so ist sie im Produktionszustand *blockiert*. Initial ist die Maschine im Zustand *leerlauf*.

Die Maschine kennt ihren TAManager, ausgedrückt in der Zustandsvariable *manager*.

Eine Maschine hat zwei Typen von internen Zeitereignissen: *ueberpruefenAA* und *ueberpruefenBA*. Bei ihrem Auftreten überprüft die Maschine, ob ein AA bzw. ein BA erteilt werden muss. Initial hat eine Maschine ein Ereignis vom Typ *ueberpruefenBA*.

6.5 Internes Modell TA-Manager

Der TA-Manager ist dafür verantwortlich, aus einkommenden AA oder BA der Maschinen TA zu generieren, sie auszuschreiben und den sich bewerbenden FTF zu erteilen.

Wenn der TA-Manager einen AA oder einen BA erhält, so versucht er zunächst, ob er aus einem zusammenpassenden Paar von einem AA und einem BA einen TA generieren kann. Ein solches Paar passt zusammen, wenn eine Maschine mit Arbeitsschritt *i* einen AA und eine Maschine mit Arbeitsschritt *i+1* einen BA erteilt hat. Aus einem BA einer Maschine mit Arbeitsschritt 1 und einem AA einer Maschine mit Arbeitsschritt *n* kann direkt ein TA generiert werden, wobei der Startort des TA das Eingangslager bzw. der Zielort das Ausgangslager ist. Trifft das nicht zu und kann der TA-Manager auch kein Paar von AA und BA finden, so muss er versuchen, einen geeigneten Zielort für den AA bzw. Startort für den BA zu finden. Im Falle eines AA einer Maschine mit Arbeitsschritt *i* fragt er der Reihe nach alle Maschinen mit Arbeitsschritt *i+1*, ob sie denn noch ein weiteres TS in ihren Eingangspuffer aufnehmen können, solange, bis er eine entsprechende Maschine gefunden hat. Im Falle eines BA einer Maschine mit Arbeitsschritt *i* fragt er der Reihe nach alle Maschinen mit Arbeitsschritt *i-1*, ob sie denn noch ein weiteres TS in ihrem Ausgangspuffer vorrätig haben, solange, bis er eine entsprechende Maschine gefunden hat. Gibt es keine entsprechende Maschine, so versucht der TA-Manager nach c_{aa_neu} bzw. c_{ba_neu} Zeiteinheiten, den AA bzw. BA erneut zu bearbeiten.

Hat der TA-Manager einen TA generiert, so schreibt er ihn wie schon bereits in Fall 3 aus Kapitel 5 ausführlich beschrieben an alle bei ihm angemeldeten FTF aus. Nach Ablauf der Bewerbungsfrist nach c_{bewerb} Zeiteinheiten vergleicht er die Bewerbungen der FTF, wobei als Kriterium die früheste Zeit, zu der das FTF am Startort des TA sein kann, zugrundegelegt wird. Dem FTF mit der besten Bewerbung wird der Zuschlag erteilt. Wird der Zuschlag abgelehnt, erteilt der TA-Manager solange dem FTF mit der nächstbesten Bewerbung den Zuschlag, bis ein FTF ihn annimmt. Hat kein FTF den Zuschlag angenommen oder sich von vorneherein kein FTF beworben, so schreibt der TA-Manager den Auftrag neu aus.

Der Zustand des TA-Managers beinhaltet eine Menge *offeneTA* von offenen, d.h. noch nicht an ein FTF fest vergebenen, TA. Jeder TA hat eine ID *id*, einen Startort *startort* und einen Zielort *zielort*. Ferner hat jeder TA hat einen Zustand *zust*, der die Werte *ausgeschrieben* und *zuschlagErteilt* haben kann. Der Wert *ausgeschrieben* bedeutet dabei, dass die Bewerbungsfrist für den TA noch läuft. Der Wert *zuschlagErteilt* bedeutet hingegen, dass die Bewerbungsfrist abgelaufen und der Zuschlag für den TA erteilt ist, aber noch nicht angenommen wurde. Weitere denkbare Zustände für einen TA wären *unerledigt* und *erledigt*. Ein TA wäre *unerledigt*, wenn der Zuschlag für ihn angenommen worden ist, aber er noch nicht ausgeführt worden ist. Nachdem das TS am Zielort abgelegt worden ist, würde der TA dann in den Zustand *erledigt* übergehen. Aus Gründen der Einfachheit beobachtet der TA-Manager die TA, für die der Zuschlag angenommen wurde, jedoch nicht mehr. Jeder TA hat neben seinem Zustand *zust* noch eine Menge von Bewerbungen in der Bewerbungstabelle *bewerbTab*, bei der es sich um eine Prioritätswarteschlange handelt.[66] Initial ist ein TA im Zustand *ausgeschrieben*.

Ein weiterer Bestandteil des Zustands des TA-Managers ist eine Menge von erhaltenen AA und BA, die noch offen sind, für die also noch kein entsprechender TA generiert wurde. Sie haben den Namen *offeneAA* bzw. *offeneBA*. Der Zustand eines AA bzw. BA beinhaltet die ID *id*, die Maschine *maschine*, die den Auftrag erteilt hat sowie den Arbeitsschritt *schritt* dieser Maschine. Ferner kann ein AA bzw. BA noch eine Menge *moeglPart* möglicher Partner haben. Das sind Maschinen die Zielort eines AA bzw. Startort eines BA sein können. Auch bei *moeglPart* handelt es sich um eine Prioritätswarteschlange.[67]

Außerdem gehört zum Zustand des TAManagers die Menge *angemeldeteFTF* der FTF, die bei dem Manager angemeldet sind. An diese FTF werden neue Ausschreibungen für TA gesendet.

Drei weitere Attribute dienen der Sicherstellung der Eindeutigkeit von IDs von neuen AA, BA und TA: *nextAA*, *nextBA* und *nextTA*. Wenn immer ein neuer AA, BA oder TA generiert wird, bekommt er die entsprechende ID und das entsprechende Attribut wird hochgezählt.

Ein TA-Manager hat drei Typen von internen Zeitereignissen: *bearbeitenAA*, *bearbeitenBA* und *timeoutTA*. Beim Auftreten von *bearbeitenAA* und *bearbeitenBA* versucht der TA-Manager, ob er zu einem AA bzw. zu einem BA einen TA generieren kann. Beim Auftreten von *timeoutTA* ist die Bewerbungsfrist für einen TA abgelaufen.

6.6 Internes Modell FTF

Ein FTF bewirbt sich um einen ausgeschriebenen TA genau dann, wenn es gerade frei ist, also keinen zugeordneten TA hat. Wenn ein FTF den Zuschlag für einen TA erhält, so nimmt es ihn genau dann an, wenn es gerade frei ist.

Hat ein FTF einen TA übernommen, so plant es zunächst seinen Weg zu Startort des TA, wobei es den statisch kürzesten Weg nimmt. Den geplanten Weg (eine Folge von Segmenten) fährt es ab, wobei es vor jedem Knoten anhält und den Verkehrsregler für diesen Knoten um Erlaubnis für die Einfahrt in diesen Knoten fragt. Ist die Erlaubnis erteilt, so fährt das FTF in den Knoten ein. Sobald das nächste Segment erreicht und der Knoten somit wieder verlassen ist, wird durch eine Nachricht vom FTF an

---

[66] In eine Prioritätswarteschlange lassen sich mittels INSERT Elemente einfügen, mittels MINIMUM wird das minimale Element zurückgegeben und mittels EXTRACT-MIN das minimale Element zurückgegeben und entfernt (siehe [CLR90]). Hier wird noch eine zusätzliche Operation EXTRACT_ALL benötigt, die alle Elemente zurückgibt.

[67] Aus Gründen der Einfachheit haben in den ausformulierten Regeln die Maschinen alle die gleiche Priorität.

den Verkehrsregler der Knoten wieder freigegeben. Ist der Startknoten des TA erreicht, nimmt das FTF das TS auf, plant seinen Weg zum Zielknoten und fährt den geplanten Weg ab. Ist der Zielknoten des TA erreicht, legt das FTF das TS ab und ist ab sofort wieder frei.

Der interne Zustand eines FTF beinhaltet wie schon der externe Zustand seinen Bewegungszustand *bewZust* (kann die Werte *stehend* oder *fahrend* annehmen), seinen aktuellen Ort *ort* (das kann ein Segment oder ein Knoten sein) und seinen Beladungszustand *belZust*, der die Werte *beladen* oder *unbeladen* annehmen kann, wobei die Informationen des FTF bezüglich dieser Attribute stets korrekt sind, also mit seinem tatsächlichen Zustand, dem externen Zustand übereinstimmen.

Zum internen Zustand eines FTF gehört das Attribut *aktuellerTA*, das ist der aktuelle TA, den das FTF gerade ausführt, wobei *aktuellerTA* nicht immer einen Wert haben muss. Der TA hat eine ID *id*, einen Startort *startort* und einen Zielort *zielort*. Ferner hat jeder TA hat einen Zustand *zust*, der die Werte *startortAnvisiert* und *zielortAnvisiert* haben kann, je nachdem, ob das FTF sich gerade auf dem Weg zum Startort befindet oder das TS bereits aufgenommen hat und sich auf dem Weg zum Zielort befindet. Weitere denkbare Zustände für einen TA wären *beworben* und *erledigt*. Ein TA wäre im Zustand *beworben* wenn das FTF sich für den TA beworben hat, aber noch keinen Zuschlag für ihn bekommen hat. Da sich das FTF aus Gründen der Einfachheit nicht merkt, für welche TA es sich beworben hat, wird dieser Zustand hier nicht benötigt. Nachdem das TS am Zielort abgelegt worden ist, würde der TA in den Zustand *erledigt* übergehen. Da der TA für das FTF damit aber nicht mehr betrachtet werden muss, braucht es sich keinen TA im Zustand *erledigt* zu speichern.[68] Initial ist ein TA stets im Zustand *startortAnvisiert*. Teil des internen Zustands des FTF ist auch der geplante Weg *weg*, das ist eine Folge[69] von Segmenten.

Ein FTF hat einen Typ von internen Zeitereignissen, den Typ *abfahrenWeg*. Beim Auftreten von *abfahrenWeg* beginnt das FTF, den in der Zustandsvariable *weg* geplanten Weg abzufahren.

Zwei interne Funktionen des FTF werden hier nur nicht formal beschrieben. Zum einen ist das die Funktion *Wegberechnung*, die als Argumente einen Startort und einen Zielort bekommt und als Ergebnis den kürzesten Weg vom Startort zum Zielort zurückgibt.[70] Zum anderen ist das die Funktion *Zeitberechnung*, die als Argument ebenfalls einen Startort und einen Zielort bekommt und als Ergebnis die Zeit zurückgibt, die das FTF zum Abfahren des kürzesten Weges vom Startort zum Zielort zurückgibt.[71]

6.7 Internes Modell Verkehrsregler

Um zu verhindern, dass es auf den Knoten zu Kollisionen zwischen FTF kommen kann, dürfen FTF einen Knoten immer erst befahren, wenn sie vom für den Knoten verantwortlichen Verkehrsregler auf Anfrage eine Erlaubnis für das Einfahren in den Knoten erhalten haben. Ist der Knoten frei, so erteilt der Verkehrsregler dem anfragenden FTF die Erlaubnis, den Knoten zu durchfahren und markiert ihn als besetzt. Ist der Knoten besetzt, so reiht der Verkehrsregler das anfragende FTF in eine Warteschlange ein. Hat das FTF den Knoten verlassen, so gibt es ihn durch eine Nachricht an den Verkehrsregler wieder frei. Der Verkehrsregler erteilt dem ersten FTF aus der Warteschlange nun die Erlaubnis, in den Knoten einzufahren. Wenn die Warteschlange leer ist, wird der Knoten wieder als frei markiert.

---

[68] Wie man sieht, hat der TA beim FTF eine andere interne Repräsentation als beim TA-Manager.

[69] Aus einer Folge lässt sich mittels HEAD das erste Element und mittels TAIL der Rest der Folge ohne das erste Element abfragen. Außerdem lässt sich mittels LENGTH die Länge der Folge bestimmen.

[70] Den kürzesten Weg in einem gerichteten Graphen kann man mit Hilfe des Algorithmus von Dijkstra bestimmen (siehe z.B. [CLR90]).

[71] Hier muss zunächst *Wegberechnung* durchgeführt werden und danach müssen einfach die Längen der Segmente des Ergebnisses aufsummiert und durch die Geschwindigkeit des FTF geteilt werden.

Der Zustand des Verkehrsreglers beinhaltet den Zustand des verwalteten Knotens *zust*, der die Werte *frei* oder *belegt* annehmen kann. Ferner beinhaltet der Zustand eine Warteschlange[72] *wartendeFTF*, in der sich alle FTF befinden, die auf die Erlaubnis zur Einfahrt in den Knoten warten.

[72] In eine Warteschlange kann man mittels ENQUEUE ein Element hinten in die Schlange einreihen und mittels DEQUEUE das vorderste Element entnehmen (FIFO-Warteschlange). Außerdem lässt sich mittels LENGTH die Länge der Warteschlange bestimmen.