

Zusammenfassung des Vortrags

# Formale Agententheorien

vorgetragen im Rahmen des Seminars: „Agentenorientierte Programmierung“

unter Leitung von Frau Dr. Lindemann

am 22.01.2002

von

Ingo Bendel

und

Conrad Flake

# 1. Einleitung

Formale Modelle ermöglichen es Annahmen klar darzulegen, was Vorhersagen über das Können und Nichtkönnen eines Modells erlaubt. Sie fördern das Verständnis von Systemen, die auf einer höheren Ebene als ihre Implementierung entwickelt wurden.

Allgemein sind Formalismen für Agentensysteme für zwei verschiedene Zwecke gefragt:

- als Interne Spezifikationssprache des Agenten (zum Agieren und Schlußfolgern des Agenten) und
- als Externe Metasprache für den Entwickler (zum Spezifizieren, Entwerfen und Überprüfen von Handlungsmöglichkeiten der Agenten in dynamischen Umgebungen)

Idealer Weise würde man beide Problem mit einer Sprache lösen, jedoch ist dies i. A. nicht möglich, da hier ein Trade-off zwischen Ausdrucksstärke und Berechenbarkeit besteht.

Echtzeitbestrebungen von Agenten in dynamischen Umgebungen erfordern eine schnell und effektiv zu berechnende Interne Sprache, während man sich zur Modellierung von möglichst verschiedenem, komplexem Verhalten in Systemen verteilter, autonomer Agenten eine sehr ausdrucksstarke Externe Sprache wünscht.

Vorteile von logik - basierten Techniken für Multiagentensysteme (MAS):

1. Durch Festlegung einer formalen, künstlichen Sprache als Gegenstück zu einer unstrukturierten, schlecht definierten, natürlichen Sprache ist es möglich, die Frage des „Was kann man ausdrücken“, auf einem festen, mathematischen Weg zu untersuchen.
2. Jede Form von Mehrdeutigkeit kann verhindert werden.
3. Die Theorie ist transparent; Beziehungen von Objekten und Zusammenhänge sind klar ersichtlich. (im Ggs. zu Programmcode, welcher Kontroll- und Implementationsaspekte erfordert, und dadurch oft recht konfus wird)

Weiterhin ist ein großer Bedarf von Tools zur Handhabung von komplexen Multiagentensystemen vorhanden, da die Entwicklung, Implementierung und das Austesten von MAS nicht einfach ist. Daraus resultieren kann als weitere Forderung an eine formale Theorie für MAS formuliert werden: Sie muss

- ein Tool ermöglichen, das Entwicklungen und Folgerungen in MASen erlaubt, sowie
- ein Tool zur Spezifikation und Verifikation von Eigenschaften des MAS bereitstellen.

Einzig nachteilig an Formalen Methoden ist, dass sie wegen ihrer Präzision die „ad hoc“ / „quick and dirty“ Vorgehen, welche kurzfristig oft effektiver sind, beschränken bzw. verhindern.

Logik-basierte Formalismen zur Systemmodellierung umfassen folgende drei Hauptbestandteile:

1. eine Menge wohl geformter Formeln (die Statements der Sprache) welche eine Klasse von syntaktischen Objekten bereitstellt (Syntax);
2. eine wohl definierte Modelltheorie, die jedem syntaktischen Objekt der Sprache eine formale Bedeutung zuordnet (Semantik);
3. und eine wohl definierte Beweistheorie (Axiome und Ableitungsregeln), das Schlußfolgern der Agenten ermöglicht. (Syntax)

## 2. Modale Logik und Mentale Einstellungen

Um einen Agenten und sein Verhalten genauer beschreiben zu können, kann man ihm mentale Einstellungen zuschreiben, welche zu jeder Zeit seinen mentalen Zustand repräsentieren. So wie informelle Einstellungen (z.B. Knowledge, Belief, Awareness ...) Wissen über die Welt beschreiben, so dienen motivierende oder soziale Einstellungen (z.B. Goal, Plan, Desire, Intention, Permission ...) der Selektion von Aktionen, die der Agent ausführen kann.

Um diese Einstellungen zu formalisieren wird eine höhere Ausdrucksstärke benötigt, als sie uns das Aussagenkalkül bieten kann. Man bedient sich hier der modalen Logik und definiert entsprechende Operatoren und Axiome.

Eine Standard Logik für Knowledge ist z.B. das S5 System mit folgenden Axiomen:

K: $[K\phi \wedge K(\phi \rightarrow \mu)] \rightarrow K\mu$	Abgeschlossen bzgl. Deduktion
D: $K\phi \rightarrow \neg K\neg\phi$	Konsistenz des Wissens
T: $K\phi \rightarrow \phi$	Knowledge Axiom (Was der Agent weiss ist wahr)
4: $K\phi \rightarrow KK\phi$	Positive Introspektion (Der Agent weiss was er weiss)
5: $\neg K\phi \rightarrow K\neg K\phi$	Negative Introspektion (Agent weiss was er nicht weiss)

Ein Problem dieser formalen Systeme ist folglich die logische Omniszenz der Agenten. Wenn ein Agent eine Menge von Fakten kennt, so kennt er auch alle daraus folgenden logischen Konsequenzen. Dies könnte zum Beispiel nebenläufige Effekte zur Folge haben.

Eine mögliche Lösung zu diesem Problem stammt von Levesque, Falgin und Hapern. Sie führten den Begriff Awareness ein und unterschieden so zwischen explizitem und implizitem Wissen. Ein Agent kann demnach nicht aus einer Formel Schlüsse ziehen, derer er sich nicht explizit bewußt ist.

# 3. Eine Theorie für berechenbare MAS

aus:

Michael J. Wooldridge  
Universität von Manchester  
Dissertation  
August 1992

## 3.1 Modell: Einführung

Diese Theorie ist dazu gedacht, die Hauptmerkmale des weiten Feldes von klassischen DAI, Agenten und Systemen in einer plausiblen Theorie zu modellieren.

Die Hauptmerkmale sind Agenten ausgestattet mit einer expliziten Menge von Beliefs aus denen sie Schlußfolgerungen ziehen können.

Beliefs (engl. Glaube) dienen der Repräsentation eines Modells der Umgebung, in der sich der Agent befindet. Sie werden in der internen, logischen Sprache  $L$  formuliert.

Für einen Agenten werden drei Arten von Aktionen unterschieden:

- kognitive Aktionen: entsprechen der Bearbeitung seiner eigenen Ressourcen, sie sind privat, d.h. sie können nicht von anderen Agenten beobachtet werden, und sind, da der Agent die ganze Kontrolle über sie hat, auch nicht fremd-beeinflußbar.
- kommunikative Aktionen: sind das Senden von Nachrichten an andere Agenten. Der Effekt den die Kommunikation verursacht, ist hier nicht unter vollständiger Kontrolle des Senders, er kann lediglich Vermutungen darüber anstellen, welchen Effekt sie bei dem Empfänger auslösen kann. Der aktuelle Effekt ist jedoch unter der Kontrolle des Empfängers.
- effektive Aktionen: werden in der realen Welt ausgeführt, und sind nicht störungssicher.

Es hat sich bewiesener Maßen als äußerst schwierig herausgestellt, realistische, handhabbare Formalismen für effektive Aktionen ausgeführt in MAS zu entwerfen. Deshalb wird in der folgenden Theorie von ihnen abstrahiert.

Agenten können also im weiteren kommunikative und kognitive Aktionen ausführen. Der Effekt von Nachrichten beim empfangenden Agenten wird mittels einer Interpretation modelliert, die jeder Agent individuell besitzt. Die epistemischen Inputs resultieren aus der Ausführung von Aktionen. Agenten können die ihrer Menge von Beliefs mittels der Belief-Revisions-Funktion, in dem die neuen epistemischen Inputs i. V. m. den vorhanden Beliefs betrachtet werden, warten.

## 3.2 Modell: Agenten

Agenten werden durch drei Attributen charakterisiert: die Beliefs, die Kommunikation sowie die kognitiven Aktionen des Agenten.

### 3.2.1 Belief

Die Beliefs werden mittels der interner Sprachen  $L$  ausgedrückt und wie folgt definiert:

$Belset = \text{powerset } Form(L)$

Symbol  $\Delta$

*Belset* ist also die Menge aller möglichen Mengen von Formeln der internen Sprache *L*, powerset steht hier für die Potenzmenge.

$\rho$  bezeichnet die Menge von Ableitungsregeln  
 $\vdash_\rho$  als Ableitungsrelation definiert für jede Regelmenge

Definition 1: Sei  $\Delta$  eine Menge von Formeln,  $\phi$  eine Formel und  $\rho$  eine Menge von Ableitungsregeln, alle aus der logischen Sprache *L*. Dann gilt  $\Delta \vdash_\rho \phi$  wenn es einen Beweis für  $\phi$  von  $\Delta$  aus gibt, der nur Regeln aus  $\rho$  benutzt.

Der Abschluß einer Menge von Beliefs unter Reduktionsregeln ist gegeben durch:

*close*:  $Belset \times powerset\ Drule \rightarrow Belset$   
 $close(\Delta, \rho) \hat{=} \{\phi \mid \Delta \vdash_\rho \phi\}$

Änderungen in der Beliefmenge werden verursacht durch epistemische (=erkenntnismäßige) Inputs, die aus einer Menge von Formeln der internen Sprache bestehen:

$Epin = powerset\ Form(L)$

Agenten können ihrer Beliefs durch gewonnene, neue Informationen mittels der Belief - Revisions-Funktion (BRF) ändern.

$Brf = Belset \times powerset\ Epin \rightarrow Belset$   
Symbol  $\beta$

Eine BRF ist konsistent, wenn sie nie zu logisch-inkonsistenten Beliefs führt.

### 3.2.2 Kommunikation

Kommunikation wird durch Austausch von Nachrichten welche wohlgeformte Formeln der internen Sprache *L* enthalten, point-to-point (=Standverbindung) modelliert. Ein Agent kann also höchstens eine Nachricht gleichzeitig verschicken.

*Agid* beliebige, abzählbare Menge von AgentenId's,  
Symbole  $i, j, k, l$

Eine Nachricht ist ein Trippel von Sender, Empfänger und Inhalt:

$Mess = Agid \times Agid \times Form(L)$   
Symbol  $\mu$

Selektorfunktionen *sender*, *recvr* geben den Sender bzw. Empfänger einer übergebenen Nachricht aus.

Forderung:

Agenten können sich nicht selbst Nachrichten schicken, und damit ihren kognitiven Zustand selbst beeinflussen:

$\forall \mu \in Mess \cdot (sender(\mu) \neq recvr(\mu))$

Nachrichten-Interpretations-Funktion:

Kommunikative Aktionen verursachen epistemische Inputs beim Empfänger der Nachricht.

$$Mess_{int} = Belset \times Mess \rightarrow E_{pin}$$

Symbol  $\iota$  (Jota)

$Mess_{nil}$  eine Menge von nil - Nachrichten, die jeder Agent jederzeit senden kann jedoch von keinem Agenten empfangen werden können. Sie dienen zur Modellierung des „nicht - senden“.

### 3.2.3. Aktion

$$Action = Belset \rightarrow E_{pin}$$

Symbol  $\alpha$

nil-Aktion werden zur Modellierung des „nichts - tun“ definiert.

Jeder Agent verfügt über kognitive und kommunikative Aktionen, jedoch sind nicht alle Aktionen in jedem kognitiven Zustand ausführbar. Deshalb wird jeder Aktion eine Bedingung zugeordnet.

Eine Bedingung ist eine Formel oder true:

$$Cond = Form(L) \cup \{true\}$$

Kognitive Aktionen werden im folgenden kurz als Aktion bezeichnet.

Aktionsregel:

$$Arule = Cond \times Action$$

Symbol  $ar$

Kommunikative Aktionen werden nur noch als Nachricht bezeichnet.

Nachrichtenregel:

$$Mrule = Cond \times Mess$$

Symbol  $mr$

Die Anwendbarkeit einer Aktionsregel, gegeben ein Belset:

$$ar\_applic: Arule \times Belset \rightarrow \{true, false\}$$

$$ar\_applic((\phi, \alpha), \Delta) \hat{=} \phi \in (\Delta \cup \{true\})$$

Eine Aktionsregel ist also anwendbar, wenn die Bedingung  $\phi$  geglaubt wird; ist sie true so ist die Regel immer anwendbar.

$mr\_applic$  analog

Menge von Aktionsregeln  $AR$

Menge von Nachrichtenregeln  $MR$

Die Aktions- und Nachrichtenregeln sind ‚schwach vollständig‘, wenn für jedes Beliefset eines Agenten immer mindestens eine Aktion ausführbar und eine Nachricht absendbar ist.

$$ar\_wk\_cmplt: powerset Arule \rightarrow \{true, false\}$$

$$ar\_wk\_cmplt(AR) \hat{=} \forall \Delta \in Belset \cdot \exists ar \in AR \cdot ar\_applic(ar, \Delta)$$

$mr\_wk\_cmplt$  analog

Eine Aktion ist durchführbar, wenn:

$ac\_legal: Action \times powerset Arule \times Belset \rightarrow \{true, false\}$

$ac\_legal(\alpha, AR, \Delta) \hat{=} \exists(\phi, \alpha') \in AR \cdot (\alpha = \alpha') \wedge ar\_applic((\phi, \alpha'), \Delta)$

$ms\_legal$  analog

Forderung:

Die Nachrichtenregelmenge soll ‚ehrlich‘ sein, d.h. Agenten ist es nicht möglich Nachrichten zu versenden, die über ihren Absender „lügen“.

$honest: Agid \times powerset Mrule \rightarrow \{true, false\}$

$honest(i, MR) \hat{=} \forall(\phi, \mu) \in MR \cdot (sender(\mu) = i)$

### 3.3 Modell: Agenten - Architektur

Agenten sind vom Typ *Agent*.

Definition 2: *Ein Agent ist eine Struktur:*

$(\Delta_0, \rho, \beta, \iota, MR, AR)$

- $\Delta_0 \in Belset$  ist eine initiale Menge von Beliefs;
- $\rho \subseteq Drule$  ist eine Menge von Ableitungsregeln für L;
- $\beta \in Brf$  ist eine Belief - Revisions - Funktion;
- $\iota \in Messint$  ist eine Nachrichten - Interpretations - Funktion;
- $MR \subseteq Mrule$  ist eine Menge von Nachrichten für die gilt:  $mr\_wk\_cmplt(MR)$ ;
- $AR \subseteq Arule$  ist eine Menge von Aktionsregeln, für die gilt:  $ar\_wk\_cmplt(AR)$ ;

Ausführungsschritte eines Agenten:

1. Interpretiere jede eintreffende Nachricht.
2. Update der Beliefs durch Aufbereitung der epistemischen Inputs resultierend aus
  - vergangenen Aktionen
  - Nachrichteninterpretationen
3. Leite den Abschluß der Ableitungsmenge des Beliefssets ab.
4. Leite die Menge der möglichen Nachrichten ab, wähle eine Nachricht aus und sende sie.
5. Leite die Menge der möglichen Aktionen ab, wähle eine Aktion und führe sie aus.
6. Goto (1).

### 3.4 Modell: Systeme

Eine Gruppe von bezeichneten Agenten nennt man ein System, deren Typ *system* ist.

Definition 3: *Ein System ist eine Struktur*

$(Ag, \Delta_0, \rho, \beta, \iota, MR, AR)$

wo

- $Ag \subseteq Agid$  ist eine abzählbare Menge von Agenten Id's;
- $\Delta_0 = Ag \xrightarrow{m} Belset$  ordnet jedem Element aus  $AR$  eine Menge von initialen Beliefs zu;
- $\rho = Ag \xrightarrow{m} powerset$  Drule ordnet jedem Element aus  $Ag$  eine Menge von Ableitungsregeln zu;
- $\beta = Ag \xrightarrow{m} Brf$  ordnet jedem Element aus  $Ag$  eine Belief - Revisions - Funktion zu;
- $\iota = Ag \xrightarrow{m} Messint$  ordnet jeder Nachrichten - Interpretations - Funktion zu;
- $MR = Ag \xrightarrow{m} powerset$  Mrule ordnet jedem Element aus  $Ag$  eine Menge von Nachrichtenregeln ein Element aus  $Ag$  zu;
- $AR = Ag \xrightarrow{m} powerset$  Arule ordnet jedem Element aus  $Ag$  eine Menge von Aktionsregeln zu;

(Symbole bezeichnen jetzt Abbildungen auf die Mengen welche die Symbole bisher bezeichnet haben!)

so daß

$\forall i \in Ag \cdot (\Delta_0(i), \rho(i), \beta(i), \iota(i), MR(i), AR(i)) \in Agent$

und

$\forall i \in Ag \cdot honest(i, MR(i)).$

Eine Funktion die bei Übergabe einer Agenten-Id und einem System  $sys$  den zugehörigen Agenten extrahiert wird wie folgt definiert:

$agent: Agid \times System \rightarrow Agent$

$agent(i, sys) \hat{=} let (Ag, \Delta_0, \rho, \beta, \iota, MR, AR) = sys$   
 $in (\Delta_0(i), \rho(i), \beta(i), \iota(i), MR(i), AR(i))$

### 3.5 Ausführungsmodelle

Um das Verhalten miteinander operierender und interagierender Agenten in einem MAS zu beschreiben, definieren wir ein Ausführungsmodell in dem der Zustand eines MAS durch Aktionen der Agenten geändert werden kann.

Zuerst wird das Agentenverhalten in einem einfachen synchronen Ausführungsmodell definiert, welches dann zu einem verschachtelten, komplexeren, aber auch realistischeren Modell, erweitert werden kann.

#### 3.5.1 Ein Synchrones Ausführungsmodell

Jeder Agent besitzt einen Startzustand. Dieser Startzustand definiert sich über das anfängliche Belief Set des Agenten. Der Startzustand eines Systems ist daher eine Menge von initialen Belief Sets, eines für jeden Agenten. Durch das Ausführen eines Zuges (Move) können Agenten ihren Zustand ändern. Ein Zug besteht aus einer kommunikativen und einer kognitiven Aktion.

Wenn *jeder* Agent einen (legalen) Zug macht, bilden diese eine Transition in unserem System, woraus ein neuer Systemzustand resultiert. Ein Zustand und eine Transition bestimmen genau einen Folgezustand (Abb.1).

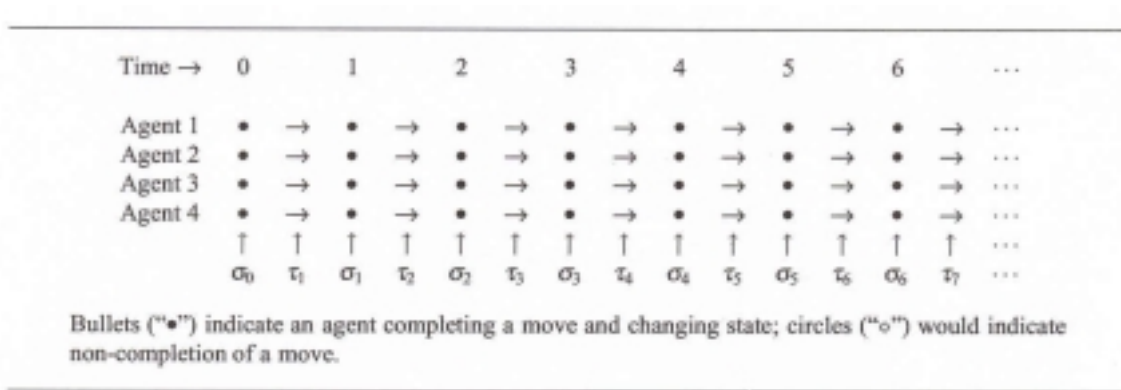


Abb. 1

Ein Systemzustand ist Mapping von Agent Ids auf Belief Sets:

$$State = Agid \xrightarrow{map} BelSet$$

Symbol:  $\sigma$

Der Startzustand eines Systems ist der Zustand, an dem jeder Agent sein initiales Beliefset besitzt und wird durch folgende Funktion definiert:

$$init\_state: System \rightarrow State$$

$$init\_state(sys) \hat{=} \{i \rightarrow close(\Delta_0(i), \rho(i)) \mid i \in Ag\}$$

Ein Move ist ein Tupel bestehend aus einer Aktion und einer Nachricht:

$$Move = Action \times Mess$$

Symbol:  $m$

Selektorfunktionen:  $action, mess$

Ein leerer Zug ist ein Tupel aus einer leeren Aktion und einer leeren Nachricht:

$$nil-move = nil-action \times nil-mess$$

Der Agent hat demnach einen *Move* getan, wenn er eine (kognitive) Aktion ausgeführt und eine Nachricht verschickt hat.

Es muss nun entschieden werden, wann ein *Move* legal ist und wann nicht. Eine Aktion ist legal, wenn die Bedingung für diese Aktion Bestandteil des Belief Sets ist:

$$ac\_legal: Action \times AR \times \Delta \rightarrow \{true, false\}$$

Eine legale Nachricht ist analog definiert:

$$ms\_legal: Mess \times MR \times \Delta \rightarrow \{true, false\}$$

Jetzt können wir entscheiden, ob ein Agent einen legalen Zug getan hat:

$$mv\_legal: Move \times Agent \times \Delta \rightarrow \{true, false\}$$

$$mv\_legal(m, ag, \Delta) \hat{=} ac\_legal(action(m), AR, \Delta) \wedge ms\_legal(mess(m), MR, \Delta)$$

Eine Transition ist definiert als ein Mapping von Agent Ids auf Moves:

$$Trans = Agid \xrightarrow{map} Move$$

Symbol:  $\tau$

Eine Transition ist genau dann legal, wenn jeder Agent einen legalen Zug ausgeführt hat:

$$trans\_legal: Trans \times System \times State \rightarrow \{true, false\}$$

$$trans\_legal(\tau, sys, \sigma) \hat{=} mv\_legal(\tau(i), agent(i, sys), \sigma(i)) \text{ für alle } i \in \text{dom } \tau$$

Eine nil-Transition ist eine Transition, in der jeder Agent einen nil-Move getan hat.

Die Funktion *sent* bestimmt die Menge an Nachrichten, die während einer Transition gesendet wurden:

$$sent: Trans \rightarrow \text{powerset } Mess$$

Die Funktion *recvd* filtert die Nachrichten heraus, die während einer Transition zu einem bestimmten Agenten gesendet wurden:

$$recvd: Agid \times Trans \rightarrow \text{powerset } Mess$$

$$recvd(i, \tau) \hat{=} \{\mu \mid \mu \in sent(\tau) \wedge recvr(\mu)=i\}$$

Wir geben jetzt eine Funktion an, die für einen Agenten, der Aktionen ausgeführt und Nachrichten erhalten hat, ein neues Belief Set ermittelt:

$$next\_bel: Agent \times BelSet \times \text{powerset } Mess \times Action \rightarrow BelSet$$

$$next\_bel(ag, \Delta, ms, \alpha) \hat{=} close(\beta(\Delta, \{\alpha(\Delta)\} \cup \{t(\Delta, \mu) \mid \mu \in ms\}), \rho) \text{ mit } ag = (\Delta_0, \rho, \beta, t, MR, AR)$$

Letztendlich kann nun eine Funktion definiert werden, die zu einem System, einem Zustand und einer Transition, den Folgezustand berechnet:

$$next\_state: System \times State \times Trans \rightarrow State$$

$$next\_state(sys, \sigma, \tau) \hat{=} \{i \rightarrow next\_bel(agent(i, sys), \sigma(i), recvd(i, \tau), action(\tau(i))) \mid i \in \text{dom } \sigma\}$$

Um von einzelnen Zuständen und Transitionen weiter zu abstrahieren, definieren wir nun Ausführungssequenzen (Runs). Ein Run ist eine Sequenz von Welten. Eine Welt besteht aus einem Zustand und der Transition, die zu diesem Zustand geführt hat:

$$World = State \times Trans$$

Symbol:  $w$

Selektorfunktionen: *state*, *trans*

Die Startwelt ist ein Tupel aus initialem Systemzustand und einer nil-Transition:

$$init\_world: System \rightarrow World$$

$$init\_world(sys) \hat{=} (init\_state(sys), nil\_trans(sys))$$

Wir definieren nun eine Funktion, die entscheidet, ob eine Welt  $w'$  Nachfolger einer Welt  $w$  ist:

$$next\_world: World \times World \times System \rightarrow \{true, false\}$$

$$next\_world(w_1, w_2, sys) \hat{=} (trans\_legal(trans(w_2), sys, state(w_1)) \wedge state(w_2) = next\_state(sys, state(w_1), trans(w_2)))$$

Eine Weltsequenz (Run) ist wie folgt definiert:

$$\text{Worldseq} = \text{World}^*$$

Die Funktion *run\_of* entscheidet, ob eine Weltsequenz in einem System gelaufen sein kann:

$$\text{run\_of}: \text{Worldseq} \times \text{System} \rightarrow \{\text{true}, \text{false}\}$$

$$\text{run\_of}(W, \text{sys}) \hat{=} (W(0) = \text{init\_world}(\text{sys})) \wedge \text{next\_world}(W(u-1), W(u), \text{sys}) \text{ für alle } u \in \mathbb{N}$$

### 3.5.2 Ein Verschachteltes Ausführungsmodell

Synchrone Ausführungsmodelle sind einfach zu beschreiben, jedoch führt die Annahme eines globalen „Schrittmachers“ oft zu einem unrealistischen Bild eines MAS. Deshalb soll das bisherige Modell erweitert werden. Jeder Agent kann zu jeder Zeit handeln! Ein Zustandsübergang erfolgt, wenn *genau ein* Agent einen Zug ausführt (Abb.2).

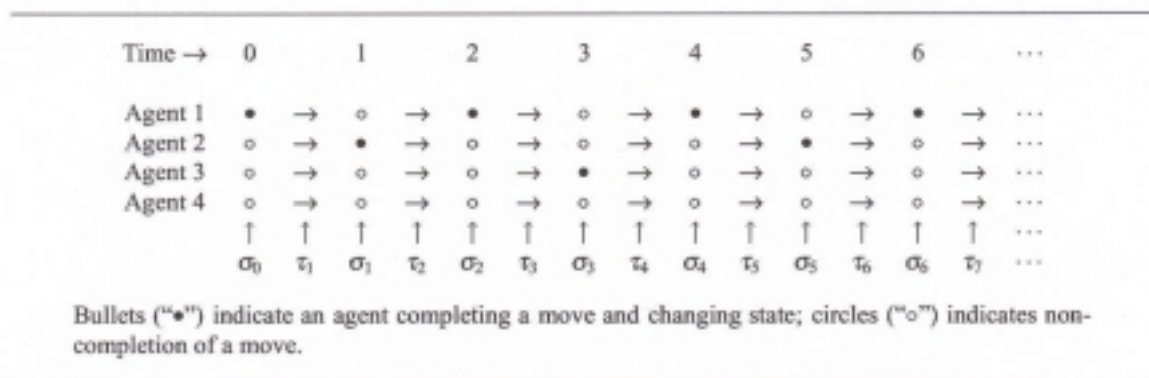


Abb. 2

In unserem Modell erhält ein Agent seine Nachrichten nach Abschluß eines legalen Zuges. Bei einer Transition werden nun nicht mehr alle Nachrichten an alle Agenten verschickt. Dieses Problem ist durch einen Nachrichten-Pool zu lösen:

$$\text{Pool} = \text{powerset Mess}$$

Die Definition eines Zustandes muss nun erweitert werden:

$$\text{state}' = (\text{Agid} \xrightarrow{\text{map}} \text{BelSet}) \times \text{Pool}$$

Selektorfunktionen: *pool, bel*

Der Startzustand hat einen leeren Pool:

$$\text{init\_state}': \text{System} \rightarrow \text{State}'$$

$$\text{init\_state}'(\text{sys}) \hat{=} (\text{init\_state}(\text{sys}), \{\})$$

Wenn ein Agent einen Zyklus ausführt, empfängt er alle für ihn im Pool enthaltenen Nachrichten:

$$\text{recvd}': \text{Agid} \times \text{State}' \rightarrow \text{powerset Mess}$$

$$\text{recvd}'(i, \sigma) \hat{=} \{\mu / \mu \in \text{pool}(\sigma) \wedge \text{recvr}(\mu) = i\}$$

In einer Transition führt immer genau ein Agent einen Move aus. Agenten die nichts tun, werden auf „nil gemappt“:

$$Trans' = Agid \xrightarrow{map} (Move \cup \{nil\})$$

Es gilt:  $\forall \tau \in Trans \cdot \exists ! i \in \text{dom } \tau \cdot \tau(i) \neq \text{nil}$

$$trans\_legal': Trans' \times System \times State' \rightarrow \{true, false\}$$

$$trans\_legal'(\tau, sys, \sigma) \hat{=} i \in \text{dom } \tau \cdot \tau(i) \neq \text{nil} \rightarrow mv\_legal(\tau(i), agent(i, sys), bel(\sigma)(i))$$

Bevor wir mit dem Modell abschließen können, werden noch zwei zusätzliche Funktionen benötigt, die wo hier nur kurz anführen wollen:

- $next\_bel\_state: System \times State' \times Trans' \rightarrow (Agid \xrightarrow{map} Belset)$
- $next\_pool\_state: State' \times Trans' \rightarrow Pool$

Nun sind alle Voraussetzungen gegeben und es kann wieder eine Funktion zur Berechnung eines Folgezustandes definiert werden:

$$next\_state': System \times State' \times Trans' \rightarrow State'$$

$$next\_state'(sys, \sigma, \tau) \hat{=} (next\_bel\_state(sys, \sigma, \tau), next\_pool\_state(\sigma, \tau))$$

## 3.6 Einige Erweiterungen zum Basis Modell

Zum Abschluß stellen wir noch einige Erweiterungen vor, die im realen Einsatz von MAS von Nutzen sein können.

### 3.6.1 Alternative Definitionen von Epin

Bisher wurde ein epistemischer Input (Epin) als unstrukturierte Formelmenge definiert. Es ist oft wünschenswert mehr Informationen über den Input zu haben. Eine Alternative wäre, den epistemischen Input mit Wahrscheinlichkeiten zu verknüpfen, um z.B. Aussagen über die Sicherheit zu treffen, mit der diese Information wahr ist:

$$Epin' = \mathbb{R} \times \text{powerset } Form(L)$$

Als weitere Möglichkeit kann man den Input in Beziehung zur Quelle (Agent Id) setzen:

$$Epin'' = Agid \times \text{powerset } Form(L)$$

Und schließlich:

$$Epin''' = Agid \times \text{powerset } Form(L) \times \mathbb{R}$$

### 3.6.2 Broadcast Messages

Neben der einfachen Annahme, das ein Agent zu einem Zeitpunkt immer nur eine Nachricht an einen Empfänger senden kann, gibt es auch hier alternative Ansätze. *Broadcast Messaging* bedeutet für einen Agenten die Möglichkeit, Nachrichten an eine Gruppe von anderen Agenten zu senden.

Um dies mit der bisher vorgestellten Theorie in Einklang zu bringen, sind wieder Änderungen an einigen Definitionen nötig:

$$Mrule' = Cond \times \text{powerset } Mess$$

$$Move' = Action \times \text{powerset } Mess$$

$$sent': Trans \rightarrow \text{powerset } Mess$$

$$sent'(\tau) \hat{=} \bigcup \{mess(m) \mid m \in \text{rng } \tau\} - Mess_{nil}$$

### 3.6.3. Mehrfache Vorbedingungen für Aktionen und Nachrichten

Diese Erweiterung ist nützlich, um an eine Aktion oder Nachricht mehrere Bedingungen zu knüpfen. Auch hierfür sind Definitionsänderungen nötig:

$$Arule' = \text{powerset } Cond \times Action$$

$$Mrule' = \text{powerset } Cond \times Mess$$

$$ar\_applic': Arule' \times Belset \rightarrow \{true, false\}$$

$$ar\_applic'((\Gamma, \alpha), \Delta) \hat{=} \forall \varphi \in \Gamma. \varphi \in (\Delta \cup \{true\})$$

$$mr\_applic': Mrule' \times Belset \rightarrow \{true, false\}$$

$$mr\_applic'((\Gamma, \alpha), \Delta) \hat{=} \forall \varphi \in \Gamma. \varphi \in (\Delta \cup \{true\})$$

### Literatur:

- Michael J. Wooldridge, University of Manchester: *Dissertation*, August 1992
- Gerhard Weiss: *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence* Kapitel 8: Formal Methods in DAI: Logic-Based Representation and Reasoning (331-376)
- A. Rao, M. Georgeff: *Formal Models and Decision Procedures for Multi-Agent Systems*, Juni 1995
- M. Wooldridge, M. Fisher: *A First-Order Branching Time Logic of Multi-Agent Systems*, 1992
- G. Lakemeyer, B. Nebel: *Foundations of Knowledge Representation and Reasoning*, 1994
- J. Halpern, Y. Moses: *Knowledge and Common Knowledge in a Distributed Environment*, 1990
- diverse Aufsätze unter: <http://citeseer.nj.nec.com/>