Deductive Reasoning Agents

Multiagentensysteme SS 09
Veranstalter: Alexa Breuing, Ipke Wachsmuth
Vortrag vom 30.04.2009
Vortragende: Sebastian Schuhmacher, Ruth Engel

1. Einleitung

Bau eines künstlichen intelligenten Systems:

- Intelligentes Verhalten wird in einem System generiert:
  - Symbolische Repräsentation der Umgebung und des Verhaltens wird erstellt
  - Repräsentation kann manipuliert werden
- Höchste Entwicklung dieser Tradition:
  - Symbolische Repräsentationen sind logische Formeln
  - Manipulation entspricht logischer Deduktion oder Theorembeweisen

Deduktion/Deduktives Schließen: Gegeben, dass gilt: (if p q) und p, inferiere dass gilt: q
Gegeben, dass gilt: (forall (-vars-) p), inferiere, dass gilt: p mit allen Vorkommen von –vars-

Beispiel Ralph:

- autonomer Roboter-Agent, der in einer echten-Welt-Umgebung von Blöcken und Fluren agiert
- Input über eine Videokamera
- Untersystem übersetzt Video in interne Repräsentation, basierend auf Logik 1.Ordnung
- Agenteninformationen über die Welt sind enthalten in einer Datenstruktur, „Knowledge Base“ (Wissensbasis)

Zwei Hauptprobleme, um Ralph zu bauen:

1.Transduction Problem:

- die reale Welt zeitnah in eine adäquate, akkurate symbolische Repräsentation zu übersetzen

=> An Vision, Sprachverständnis, Lernen etc. arbeiten

2.Representation Problem

- Informationen symbolisch so repräsentieren, dass die Agenten sie zeitnah manipulieren / auf ihnen schlussfolgern können

=> An Wissensbasis, Automatisches Schlussfolgern, Automatisches Planen, etc. arbeiten

2. Agenten als „Theorem Provers“

Deliberative Agenten: Benötigen interne Repräsentation und denken vor dem Handeln nach
Theorembeweise: Eine Anfrage wird als zu beweisendes Theorem betrachtet (durch Anwendung von Schlussregeln oder Widerlegung des Gegenteils)

Einfaches Modell eines deliberativen Agenten:

- Interner Zustand als Datenbasis von Formeln der PL-1
  z.B. *Open (valve221)*
- Formeln sind recht einfach zu benutzen, um die Umgebung zu repräsentieren
- Datenbasis eines Agenten spielt analoge Rolle zum Glauben eines Menschen
- Datenbasis ist die Information, die der Agent über seine Umgebung hat
- Fakten in der Datenbasis des Agenten können fehlerhaft sein
  Mögliche Gründe:
  - Sensoren des Agenten können fehlerhaft sein
  - Sein Verständnis könnte fehlerhaft sein
  - Interpretation der Formel „Open“ könnte anders sein, als vom Agentendesigner gedacht

- L : Menge von Sätzen Logik 1.Ordnung
- D = $\wp$(L) : Menge von Datenbanken (L)/ Menge von Mengen von PL-1 Formeln
- Interner Zustand eines Agenten = Element von D
- Δ, $\Delta 1$, $\Delta 2$, ... : Mitglieder von D
- ρ : Menge von Deduktionsregeln
- $\Delta \vdash_\rho \varphi$: Formel φ kann durch Anwendung von ρ aus Δ bewiesen werden

Funktion *see:* *S => Per*
Funktion *next:* *next: D x Per => D*
Funktion zur Aktionsauswahl: *action: D => Ac*

```
Function: Action Selection as Theorem Proving
1.  function action(Δ : D) returns an action Ac
2.  begin
3.      for each α ∈ Ac do
4.          if Δ ⊢ρ Do(α) then
5.              return α
6.          end-if
7.      end-for
8.      for each α ∈ Ac do
9.          if Δ ⊬ρ ¬Do(α) then
10.             return α
11.         end-if
12.     end-for
13.     return null
14. end function action
```

3-7 die Formel Do (α) von seiner Datenbasis, indem er die Deduktionsregeln ρ anwendet
8-12 War die Überprüfung von Do (α) nicht erfolgreich, wird versucht, eine Action α ϵ *Ac*, sodass ¬ Do (α) nicht aus der Datenbasis durch benutzen der Deduktionsregeln gewonnen werden kann.
13-14 Wird keine Aktion gefunden und ausgewählt, wird *Null* ausgeworfen

VACUUM_WORLD (Beispiel)
Idee: Roboter-Agent, der ein Haus reinigt
Ziel: Den Raum kontinuierlich nach Dreck durchsuchen und diesen ggf. beseitigen
- Ausstattung: Sensor, der Dreck erkennt, Vakuum-Reiniger, der Dreck entfernen kann
- Roboter hat definierte Orientierung (Nord, Ost, Süd, West)
- Roboter kann einen Schritt vorwärts machen oder sich um 90° nach rechts drehen
- hier: Roboter bewegt sich in einem 3x3 Gitter
- Roboter startet im Quadrat (0,0) Richtung Norden
- 3 mögliche Aktionen: *forward (Vorwärts), suck (Saugen), turn (Drehen)*
- 3 Hauptprädikate:
    In (x,y) Agent ist auf Position *(x,y)*
    Dirt (x,y) Dreck auf Position *(x,y)*
    Facing(d) Agent schaut in Richtung *(d)*

-*next*-Funktion:
- muss die wahrgenommene Information (*dirt* oder *null*), die sie aus der Umgebung erhalten hat, betrachten und eine neue Datenbasis mit dieser neuen Information generieren
- muss alte oder irrelevante Informationen entfernen
- muss die neue Position und Richtung des Agenten herausfinden

-Funktion zur Bescheribung alter Informationen:
    $old(\Delta) = \{ P(t1,...tn)$ I $P \varepsilon \{In, Dirt, Facing\}$ and $P(t1,....tn) \varepsilon \Delta\}$
- Funktion *new*:
    $new: D \times Per \Rightarrow D$
- Funktion *next*:
    $next(\Delta,p) = (\Delta \setminus old(\Delta)) \cup new(\Delta,p)$
- Deduktionsregeln: φ(…) => ψ(…)
- In (x,y) Λ Dirt (x,y) => Do (suck)
- Basic Navigation Algorithm
- Roboter bewegt sich von (0,0) zu (0,1) zu (0,2) zu (1,2) zu (1,1)...Ist er bei (2,2) muss er zurück zu (0,0)

=> weitere simple Regeln generieren zusammen mit Next-Function das erforderte Verhalten unseres Agenten

Aber: Völlig unpraktisch, auf diese „naive“ Weise Agenten zu bauen:
- Agentenregelnset ρ designed für beliebige Datenbasis Δ.
  Können wir Do (α) beweisen können, dann ist α eine optimale Aktion
  ACHTUNG!!! Wenn sich die Umgebung zwischenzeitlich geändert hat, gibt es keine Garantie dafür, dass α optimal ist!
- Da Schlussfolgern von logikbasierenden Agenten alles andere als unmittelbar ist (dann könnte man dieses Problem vernachlässigen), kann die Zeitspanne sehr groß, und damit α weit weg von einer optimalen Lösung sein.

FAZIT:
- Agent besitzt berechnende Rationalität, wenn er zu Beginn des Entscheidungsprozesses eine optimale Lösung vorschlägt
-Wechselt die Umgebung schneller als der Agent Entscheidungen treffen kann
=> Berechnende Rationalität nicht akzeptabel

Abwendung von logikbasierender Näherung => Verlust einer einfachen, eleganten, logischen Semantik

Weitere Probleme:
1. *see*-Funktion (Wahrnehmungskomponente)
=>In vielen Umgebungen nicht offensichtlich, wie die Ausarbeitung von der Umgebung zu einer symbolischen Wahrnehmung realisiert werden könnte

2. Das Repräsentieren von Eigenschaften einer dynamischen, echten-Welt-Umgebung ist extrem schwer.
=>Gerade ziemlich simples „prozedurales“ Wissen in traditionelle Logik umzuwandeln kann ziemlich unintuitiv und unhandlich sein
Schlußfolgerung: Die Probleme, die sich auf Repräsentieren und Schlussfolgern über komplexe, dynamische, möglicherweise physikalischen Umgebungen beziehen, sind im Wesentlichen ungelöst.

3. Agent-Oriented Programming (AgentO, von Yoav Shoham 1993)

Einführung mentaler Begriffe: Annahme (believe), Ziel (desire), Plan (intention)
Idee dahinter: Verhalten des Menschen funktioniert ähnlich
erste Implementierung dieser Art: AgentO

Ein AgentO-Agent ist spezifiziert durch:
- Fähigkeiten (was der Agent tun kann)
- initiale Annahmen
- initiale Verpflichtungen (commitments)
- Verpflichtungsregeln (was der Agent tut)

Jede dieser Verpflichtungsregeln beinhaltet:
- Nachrichten-Zustand: zum Vergleich mit Nachrichten, die den Agenten erreichen. (requests & unrequests → Änderung von Verpflichtungen, inform → Annahmen)
- Mental-Zustand: zum Vergleich mit Annahmen.
- Aktion: Treffen die Zustände zu, startet die Aktion. (private/communicative)

Beispiel einer Verpflichtungsregel in Worte gefasst:
Wenn ich eine Nachricht eines Agenten erhalte, die bei mir eine Aktion zu einem Zeitpunkt anfragt (request) und ich annehme (believe), dass
- der Agent aktuell ein Freund ist;
- ich die Aktion durchführen kann;
- ich zu diesem Zeitpunkt zu keiner anderen Aktion verpflichtet bin,

dann verpflichte, die Aktion zu dem Zeitpunkt durchzuführen.

Der Ablauf eines operierenden Agenten:
(1) Lese aktuelle Nachrichten, aktualisiere Annahmen (und daraus folgernd Verpflichtungen)
(2) Führe alle Verpflichtungen aus für den aktuellen Durchlauf, bei denen der Fähigkeiten-Zustand auf die Aktion zutrifft.
(3) Goto (1).

4. Concurrent MetateM (von Michael Fisher 1994)

Concurrent MetateM zeichnet sich aus durch:
- direkte Ausführung logischer Formeln (deduktiver Theorembeweiser)
- simultan arbeitende Agenten, kommunizieren mit anderen Agenten per Rundmeldungen
- das Verhalten der Agenten wird mit einer „zeitlogischen Beschreibung“ programmiert

Jeder Concurrent MetateM Agent besitzt zwei Hauptkomponenten:
- Interface: Wie findet die Interaktion mit der Umgebung statt (z.B. mit anderen Agenten)
- Computational Engine: Berechnungsvorschrift wie der Agent handelt

Dabei besteht das Interface aus:
- einer einzigartigen Agenten ID („agent identifier“)
- einer Menge von Symbolen, die definieren welche eingehende Nachrichten vom Agenten akzeptiert werden („environment propositions“)
- einer Menge von Symbolen, die definieren welche Nachrichten der Agent senden kann („component propositions“)

Beispiel des Interfaces eines Stapel-Agenten: stack(pop,push)[popped,full]

Die „Computational Engine“:

- basiert auf direkt ausführbarer zeitlogischer Beschreibung
- eine Menge von zeitlogischen Formeln (Regeln) der Form:
  Antezedent über die Vergangenheit => Konsequent über Gegenwart und Zukunft

Betrachten dazu die „Propositional MetateM Logic“ (PML, von Barringer et al. 1989):

Operator	Meaning
$\bigcirc\varphi$	φ is true 'tomorrow'
$\mathbf{O}\varphi$	φ was true 'yesterday'
$\Diamond\varphi$	at some time in the future, φ
$\Box\varphi$	always in the future, φ
$\blacklozenge\varphi$	at some time in the past, φ
$\blacksquare\varphi$	always in the past, φ
$\varphi\,\mathcal{U}\,\psi$	φ will be true until ψ
$\varphi\,S\,\psi$	φ has been true since ψ
$\varphi\,\mathcal{W}\,\psi$	φ is true unless ψ
$\varphi\,\mathcal{Z}\,\psi$	φ is true zince ψ

Tabelle:
Zusammenfassung temporaler Operatoren

(zince: „weak since“, ähnlich wie since, aber erlaubt die Möglichkeit, dass das zweite Argument nie zutraf)

Beispiele:
$\Box$important(agents), $\Diamond$ important(Janine),

(¬friends(us)) $\mathcal{U}$apologize (you), $\bigcirc$ apologize(you)

Anwendung eines Concurrent MetateM Agenten:
Der Agent durchläuft seine Regeln und prüft deren Antezedenten auf Übereinstimmung mit seiner Historie. Bei Treffern wird der Konsequent der Regel ausgeführt. Also folgend kann der Ablauf aussehen:

(1) Die Historie des Agenten aktualisieren durch Erhalt von Nachrichten (environment propositions)
(2) „Feuernde“ Regeln finden (Vergleich zwischen Historie und vergangenen Antezedenten jeder Regel)
(3) Füge zuerst neue Konsequenten und alte Verpflichtungen zum current constraint zusammen. Erzeuge unter diesen Bedingungen den nächsten Zustand.
(4) Goto (1).

Wird dabei eine der component proposition aus dem Interface wahr, so wird sie vom Agenten versendet.

Abbildungen: Links ein Beispiel für ein einfaches Concurrent MetateM System. Rechts ein Verlauf dazu:

$$
\begin{array}{l}
rp(ask1, ask2)[give1, give2]: \\
\qquad \mathbf{O}ask1 \Rightarrow \Diamond give1; \\
\qquad \mathbf{O}ask2 \Rightarrow \Diamond give2; \\
\qquad \text{start} \Rightarrow \Box\neg(give1 \wedge give2). \\
\\
rc1(give1)[ask1]: \\
\qquad \text{start} \Rightarrow ask1; \\
\qquad \mathbf{O}ask1 \Rightarrow ask1. \\
\\
rc2(ask1, give2)[ask2]: \\
\qquad \mathbf{O}(ask1 \wedge \neg ask2) \Rightarrow ask2.
\end{array}
$$

Time	Agent		
	rp	$rc1$	$rc2$
0.		$ask1$	
1.	$ask1$	$ask1$	$ask2$
2.	$ask1, ask2, give1$	$ask1$	
3.	$ask1, give2$	$ask1, give1$	$ask2$
4.	$ask1, ask2, give1$	$ask1$	$give2$
5.	...	...	...

5. Fragen:

- Was versteht man unter dem Transduction Problem?
- Was bedeutet deduktives Schließen?
- Welche Idee steht hinter Agenten als Theoremprover?
- Welche Vor- und Nachteile hat die logikbasierende Näherung?
- Warum darf sich die Umwelt beim logikbasierenden Agenten nicht schneller ändern als er Entscheidungen treffen kann?
- Was ist die Grundidee der agentenorientierten Programmierung?
- Nenne die drei Zustände der Verpflichtungsregeln eines AgentO-Agenten.
- Denke dir ein Beispiel für ein Interface eines Concurrent-MetateM-Agenten aus. (z.B. Staubsauger mit Staubbeutel-Anzeige)
- Was ist die „Computational Engine“ eines Concurrent-MetateM-Agenten?