Zusammenfassung - Multi-Agenten-Systeme

Andreas Ruscheinski,

13. Dezember 2015

Korrektheit und Vollständigkeit der Informationen sind nicht gewährleistet. Macht euch eigene Notizen oder ergänzt/korrigiert meine Ausführungen!

Inhaltsverzeichnis

1 Einführung

1.1 Definition

- Ein Agent ist ein Computer System welches **selbstständig** Aktionen im Interesse des Benutzers ausführen kann.
- Ein Agent **befindet** sich in einer **dynamischen Umgebung** befindet mit welcher er interagiert.
- Ein Multi-Agenten-System besteht aus **meheren Agenten**, welcher **miteinander agieren**.
- In einem Multi-Agenten-System ist es notwending für **erfolgreiche Interaktion** dass Agenten miteinander **kooperieren**, **sich abstimmen** und miteinander **verhandeln** können.

1.2 Eigenschaften

- Jeder Agent hat **keine vollständigen Informationen** über die Umgebung
- Es gibt **keine globale Kontrolle** der Agenten
- Die Daten sind **dezentralisiert**
- Die Berechnung erfolgt **asynchron**

1.3 Gründe für den Einsatz von MAS

- Problem kann nicht zentralisiert gelöst werden da die **Ressourcen limitiert** sind
- **Reduktion der Ausfall-Wahrscheinlichkeit** in gegenüber einem zentralisierten System
- **Gewährleistung der inter-konnektion und inter-operation** von verschiedenen Systemen
- Lösung von Problemen welche eine **Menge aus autonomen Komponenten behandeln**

1.4 Konkrete Anwendungsgebiete

- Clound-Management
- Ubiquitous Computing
- Grid-Software
- Spiele
- Verschiedene Gebiete der Industrie (Car-Assembly, Factory Management)
- Simulation

2 Rolle der Logik in MAS

2.1 Gründe für Logik

- Wissensbasis + Aktionen mit Voraussetzung und Auswirkung → Plan für Lösung des Problems
- Logik ist ein Framework für das Verstehen von Systemen
- Verifikation, Ausführungsspezifikation, Planung

2.2 Logik-basierende Architektur

- Grundidee: Beschreibung einer Regelmenge für die Beschreibung der besten Aktion für einen gegeben Zustand
- Bestandteile:
  - p: eine Theorie (eine Menge von Regeln)
  - Δ: Datenbank mit den aktuellen Zustand der Welt
  - A: eine Menge von Aktionen welcher ein Agent ausführen kann
  - $\Delta \vdash_p \phi$: d.h. ϕ kann aus der Δ und p abgeleitet werden, mit ϕ =Do(a) können wir aus den aktuellen Zustand der Welt auf die bestmögliche Aktion logisch schließen
- Grundlegender Algorithmus (unabhänging von verwendeter Logik)
  1. see(s,p), generiert Beobachtung aus der neuen Welt
  2. next(Δ,p), update der Datenbank
  3. action(Δ), ermittelt die auszuführende Aktion aus der Datenbank, entweder ist die Aktion direkt beschrieben oder kann aus den Regeln abgeleitet werden kann

2.3 Modal Logik

- Erlaubt Ausdrücke wie: wahrscheinlich wahr, geglaubt wahr, wahr in der Zukunft usw.
- Syntax:
  - Prädikatenlogik mit Erweiterung
  - Prop: eine menge von atomaren Formeln
  - $\diamond$ p: möglicherweise p, manchmal p
  - $\Box$p: immer p, notwendigerweise p
- Semantik:
  - Kripke-Struktur: $< W, R, \mu >$
    W eine Menge von Welten
    R eine Menge von binär Relationen, beschreiben den Übergang zwischen den Welten
    μ Abbildungsfunktion welche jeder Welt Eigenschaften zuordnet ($\mu : W \rightarrow 2^{Prop}$)
  - Definition von $\diamond$ und $\Box$ Operator auf Basis von Erreichbarkeit der Welten in einer Kripke-Struktur
  - $\Box p$: p ist wahr in allen Welten, welche von der aktuellen Welt erreichbar sind
  - $\diamond p$: p ist wahr, wenn mindestens eine Welt erreichbar in welcher p wahr ist

- für R muss zusätzlich gelten:

  reflexiv für jedes $x \in W$ gilt $R(x,x)$

  transitiv für jedes $x,y,z \in W$ gilt $R(x,y) \wedge R(y,z) \implies R(x,z)$

  seriell für jedes $x \in W$ existiert ein y so dass gilt $R(x,y)$

  euklidisch wenn für jedes $x,y,z \in W$ mit $R(x,y)$ und $R(x,z)$ gilt auch $R(y,z)$

- Axiome:

  $\Box p \Rightarrow p$ Wenn immer p gilt folgt daraus das aktuell p gilt

  $\Box p \Rightarrow \Diamond p$ Wenn p immer wahr ist, ist p auch in mindestens einer Welt wahr

  $\Box p \Rightarrow \Box\Box p$ Wenn p ist immer wahr, ist p auch immer wahr wenn wir einen Übergang machen

  $\Diamond p \Rightarrow \Box \Diamond p$ Wenn p in mindestens einer Welt wahr ist, ist p für immer Wahr wenn wir diese Welt erreicht haben

- Anwendung der modal Logik auf Agenten durch Einführung von Indizes, welche entsprechend für Agent gelten

- Axiome und Agenten:

  $K_i p \Rightarrow p$ Wenn Agent glaubt das p wahr ist, p ist auch in Wirklichkeit wahr

  $K_i p \Rightarrow \neg K_i \neg p$ Wenn der Agent p glaubt, glaub er nicht die Negation

  $K_i p \Rightarrow K_i K_i^p$ Wenn der Agent p glaubt, weiß er selbst dass er p glaubt

  $\neg K_i \neg p \Rightarrow K_i \neg K_i \neg p$ Der Agent weiß, was er nicht weiß.

  ????

3 Planning

3.1 Einführung

- Ziel 1: Intelligentes Verhalten ohne explizite Repräsentation des Wissens
- Ziel 2: Intelligenten Verhalten ohne abstraktes schließen über die Repräsentation des Wissens
- Idee 1: Echte Intelligenz gibt es nur ein einer Welt und nicht losgelöst von dieser wie in Theorem Beweisern und Expertensysteme
- Idee 2: Intelligentes Verhalten entsteht erst als Ergebnis der Interaktion mit der Umgebung

3.2 Subsumtion Architektur

- Traditionell: Die Intelligenz steht zwischen Beobachtung und Aktion d.h. aus Beobachtungen wird geschlossen welche Aktion ausgeführt wird
- Neu: Die Intelligenz ensteht beim Beobachter durch die Aktionen in der Welt d.h. ein Agent ist nicht an sich intelligent sondern wirkt intelligent für einen Beobachter
- Entscheidungsfindung durch verschiedene Aufgaben:
  - Verhalten ist eine Abbildung von Zustand auf Aktion
  - Verarbeitung der Sensorwerte mit Schluss auf den Zustand
  - Zustände und Aktionen sind direkt gekoppelt
- Mechanismus für die Auswahl der Aktionen: Prioritäten
- Formales Modell:
  - Ein Verhalten (Behavior) $b \in Beh$ ist ein Tupel (c,a) mit $c \subseteq P, a \in A$ wobei P ist eine Menge von Beobachtungen und A eine Menge von Aktionen
  - Ein Verhalten wird ausgeführt wenn die Umgebung ist in dem Zustand $s \in S$ und wenn $see(s) \in c$
  - Subsumption Hierachie wird realisiert durch eine Hemmungs-Relation $b_1 \prec b_2$ (b_1 hemmt b_2 d.h. b_1 hat eine höhere Priorität)
- Aktionsauswahl-Algorithmus:
  1. Berechne die Menge von aktivierbaren Aktionen $FB = \{(c,a) | (c,a) \in Beh \wedge see(s) \in c\}$
  2. für jede Aktion in FB überprüfe ob es eine Aktion in FB gibt welche eine höhere Priorität hat

3. wenn Aktion gefunden gib a zurück

- Vorteile:
  - Einfach und hohe Ausdrucksfähigkeit
  - Die Berechnung ist einfach nach zu vollziehen
  - Robust gegen Ausfälle
  - Das gesamte Verhalten entsteht durch Interaktion mit der Umwelt
- Nachteile:
  - Verhalten wird hard codiert unter der Annahme die Umgebung genau zu kennen
  - Schwierige Entscheidung über das Standard Verhalten
  - langwierige Entscheidungen schwer möglich
  - skaliert nicht in größeren Systemen

3.3 Planning

- Umsetzung durch z.B STRIPS Planner
  - Repräsentation der Umgebung durch Ontologie (Begriffe + Relationen)
  - Beschreibung der aktuellen Welt durch Verwendung der Ontologie Begriffe (closed world assumption: alles was nicht angegeben wird ist falsch)
  - Jede Aktion hat Name, Pre-Condition List (alle Bedingungen müssen wahr sein bevor Aktion ausgeführt werden kann), Delete-List (Bedingungen welche nach der Ausführung nicht mehr wahr sind), Add-List (Bedingungen welche nach der Ausführung der Aktion gelten) (können alle Variablen erhalten für allgemeine Aussagen)
- Ein Plan ist eine Liste von Aktionen, mit Variablen ersetzt durch Konstanten. Die Ausführung der Aktionen führt von den aktuellen Zustand in ein einen Zustand welche das Ziel erfüllt. Der Plan ist vollständig (keine weiteren Aktionen notwendig) und konsistent (alle Pre-Conditions sind erfüllt) und die Schritte können hintereinander ausgeführt werden ohne dass die Ausführung eines Schrittes beeinflusst wird.
- Formal: partielly ordered Plans
  - Plan Schritt mit Partieller Ordnung $\prec$: $S_i \prec S_j$ bedeutet dass S_i vor S_j ausgeführt werden muss
  - Eine Menge von Variablen Zuordnungen $x = t$ mit x ist eine Variable und t ist eine Konstante
  - Eine Menge von Kausalen Relationen: $S_i \rightarrow S_j$ bedeutet die Ausführung S_i macht die Vorbedingungen von S_j wahr
- Formal: Eigenschaften Konsistenz und Vollständigkeit
  - Vollständigkeit:
    - es gilt: $\forall S_j$ mit $c \in Precond(S_j)$ und $\exists S_i$ mit $S_i \prec S_j$ und $c \in Effect(S_i)$ (die Vorbedingungen für S_j sind Teil des Effektes von S_i)
    - für eine Sequenz gilt:$\forall S_k$ mit $S_i \prec S_k \prec S_j, \neg c \notin EffectS_k$
  - Konsistenz: Wenn $S_i \prec S_j$ denn $S_j \prec S_i$ und wenn $x = A$ denn $x \neq B$ für verschiedene A und B für die Variable x.
- Vorlesungfolien für Beispiel!!!
- iterative Erstellung eines Plans durch rückwärts anwenden der Regeln d.h. in jedem Schritt wird eine offene Bedingung durch die entsprechende Aktion erfüllt
- dadurch können Konflikte entstehen
  - Ein Konflikt gdw. wenn S_3 bedroht die kausale Ordnung zwischen S_1 und S_2
  - Lösung 1: Demotion - S_3 vor S_1 und S_2 ausführen
  - Lösung 2: Promotion - S_3 nach S_1 und S_2 ausführen
  - sollte dies wieder zu Konflikte führen Backtrack und eine andere Lösung ausprobieren in dem S_3 neben S_1 zu S_2 eingeordnet wird

3.4 Planning Agents

- Erster Ansatz für den Planning Agent:
  1. beobachte die Umgebung
  2. aktualisiere das interne Modell der Umgebung
  3. ermittle welche Intention als nächstes erreicht werden soll
  4. benutze means-end Resoning für die Erstellung des Plans welche die Intention erreicht
  5. führe Plan aus
- means-end Resoning: Gib den Agent eine Repräsentation der Ziele/Intentionen welche erreicht werden sollen, Aktionen welche er ausführen kann, der Umgebung → Agent nutzt Repräsentationen um einen Plan zu generieren
- Problem: Planung und Ermittelung welches Ziel als nächstes erreicht werden soll kosten Zeit
- dadurch kann die Situation entstehen dass der Agent ein Ziel erreichen will welches nach dessen Ermittelung nicht mehr optimal ist
- wenn Planung und Ermittelung schnell genug sind und die Welt sich nicht verändert hat
- Agenten Algorithmus formal 1:
  1. B = B_0 - Ausgangs-Beliefs
  2. führe unendlich lange aus:
  3. beobachte Umwelt - p
  4. B = brf(B,p) - Aktualisierung des der eigenen Beliefs
  5. I = deliberate(B) - Ermittelung
  6. π = plan(B,I) - Plane
  7. execute(π) - führe Plan aus
- Ermittelung durch: option-generation (Erstelle mögliche Ziele) und filter (Auswahl des Ziels)
- Agenten Algorithmus formal 2 - BDI Agent:
  1. B = B_0 - Ausgangs-Beliefs
  2. führe unendlich lange aus:
  3. beobachte Umwelt - p
  4. B = brf(B,p) - Aktualisierung des der eigenen Beliefs
  5. D = options(B,i) - Desires (Ziele)
  6. I = filter(B,D,I) - Ermittelung der Intentionen durch Beliefs(Wissen über Umwelt), Desires(Ziele) und Intentions(gewählten Ziel)
  7. π = plan(B,I) - Plane
  8. execute(π) - führe Plan aus
- Begriffserklärungen:
  - Beliefs - Weltwisssen
  - Desires - Ziele
  - Intentions - Absicht (Ziel welches ich jetzt erreichen möchte)

3.5 Commitments

- Strategien:
  - Blind Commitment - Agent will Intention erreichen bis er glaubt die Intention erreicht zu haben
  - Single-minded Commitment - Agent will Intention erreichen bis er glaubt diese erreicht zu haben oder es nicht mehr möglich ist
  - Open-minded Commitment - Agent will Intention solange er glaubt dieser erreichen zu können
- Anpassungen des Agenten Algorithmus

1. wie bisher → Blind Commitment
2. wie bisher - Ermittle Plan und solange dieser nicht leer ist führe diesen Schrittweise aus, nach jedem Schritt hole neuen Percept + Belief Update: wenn Ziel nicht mehr erreicht werden kann ermittle neuen Plan → Single-minded da keine neuen Intentionen erwogen werden
3. wie bisher - Ermittle Plan und solange dieser nicht leer ist führe diesen Schrittweise aus solange Intention nicht erreicht und dieser noch nicht unmöglich ist, nach jedem Schritt hole neuen Percept + Belief Update + Desires ermitteln + Intentions ermitteln :wenn Ziel nicht mehr erreicht werden kann ermittle neuen Plan

- Problem: Agent will Intentions verfolgen selbst wenn schon klar ist das Ziel nicht mehr erreicht werden kann und Agent will konstant berücksichtigten dass evtl unnötige Zeit verschwendet wurde und deshalb nie dass Ziel erreichen
- Lösung: Meta-Level Control welches entscheidet wann der Agent seine Intention verwirft

4 Kooperation von Agenten

4.1 Einführung

- Ein Multiagenten System besteht aus meheren Agenten welche
  - Interagieren durch Kommunikation
  - Mit der Umgebung interagieren können
  - Haben unterschiedliche Einflussbereiche auf die Umgebung
  - stehen in Beziehungen zu einander (Organisation)
- Egoistische Agenten bergen Potential für Konflikte da jeder seine Ziele verfolgen möchte
- Lösung: Kooperation mittels Entscheidungstheorie

4.2 Entscheidungstheorie - Spieltheorie

- Wir haben zwei Agenten A_i und A_j
- jeder Agent hat die gleiche Menge $\Omega = \{\omega_1, \omega_2, \dots\}$ eine Menge von Ergebnissen welche für den Agenten von Bedeutung ist
- die Bedeutung für einen Agent wird durch eine Utilty-Funktion repräsentiert $u_{i,j} : \Omega \to \mathbb{R}$ (jeder Agent hat eine Utility-Funktion)
- Utility-Funktion bildet Ordnung der Ergebnisse: $\omega \geq_i \omega'$ bedeutet also $u_i(\omega) \geq u_i(\omega')$
- Utility beschreibt den Nutzen nicht GELD!
- Modell für mehrere Agenten:
  - Agenten führen Aktion gleichzeitig aus, als Ergebnis etwas aus Ω
  - das aktuelle Ergebnis hängt von Kombinationen von Aktionen ab
  - Jeder Agent hat nur zwei mögliche Aktionen: C und D
  - $\tau : A_c \times A_c \to \Omega$ eine Funktion welche die Aktion von den beiden Agenten auf ein Ergebnis abbildet
  - Eine Welt lässt sich mittels τ beschreiben unter Verwendung aller möglichen Kombinationen von Aktionen → hat ein Ergebnis
  - jeder Agent kann die Utility von dem Ergebnis berechnen
- Darstellung mittels Payoff Matrix (Ein Agent ist Spalte (linker Rechter), ein Agent ist Zeile (linker Eintrag)

	D	C
D	1,1	1,4
C	4,1	4,4

- Beispiel: $\tau(D, C) = \omega_1$ mit $u_i(\omega_1) = 1$ und $u_j(\omega_1) = 4$

4.3 Strategien

- Dominate-Strategie
  - Aus einer gegebenen Strategie s (Beispiel: D oder C) für Agent i ergeben sich verschiedene Ergebnisse (ω)
  - Strategie s_1 dominiert Strategie s_2 wenn jedes Ergebnis wenn s_1 gespielt wird bevorzugt wird gegenüber s_2
  - Rationale Agenten werden niemals dominierte Strategien spielen d.h. es gibt eine Strategie welche besser ist
  - Ziel: dominierte Strategie elimieren
  - Anmerkung: es ist nicht immer möglich eine nur eine nicht dominierte Strategie zu finden
- Nash-Equilibrium (Nash-Gleichgewicht):
  - Strategie s_i und s_j sind ein einem Nash-Gleichgewicht gdw: Unter der Annahme dass Agent i spielt s_i und Agent j kann nicht besser sein als wenn er s_j spielt UND unter der Annahme dass Agent j spielt s_j und Agent i kann nicht besser sein als wenn er s_i spielt
  - Aber: nicht jede Interaktion hat ein Nash-Gleichgewicht und einige Interaktionen haben mehrere Nash-Gleichgewichte
- Pareto Optimum:
  - Gegeben einer initialen Zuordnung von Gütern für eine Menge von Agenten
  - Eine Änderung welche für einen Agent besser ist aber für keinen anderen Agenten schlechter nennt sich Pareto-Verbesserung
  - Eine Zuweisung ist Pareto-Optimal gdw. keine weiteren Pareto-Verbesserungen gemacht werden können
- Kompetative und Null-Summen-Interaktionen
  - Szenarios in denen Agenten gegenläufige Preferänzen haben sind streng kompetativ
  - Null-Summen Spiele gdw. die Summe aller Utilitys der Agenten = 0 sind: $u_i(\omega) + u_j(\omega) = 0, \forall \omega \in \Omega$
  - Null-Summen Spiele sind streng kompetativ, im Real-Life selten
- Beispiel: Prison Dilemma
  - zwei Männer wurden verhaftet wegen Diebstahl
  - wenn jmd gesteht und der andere nicht geht der Geständige 3 Jahre ins Gefägniss und der andere kommt frei
  - wenn beide gestehen beide gehen für 3 Jahre ins Gefängnis
  - Wenn keiner gesteht geht jeder 1 Jahr ins Gefängnis
  - TODO: ???