1 AI101-06: Logik und KI 1 - Aussagenlogik

1.1 Einführung: Wissensbasierte Agenten

Während reflexbasierte Agenten oder Suchalgorithmen oft nur über begrenztes Verständnis ihrer Umgebung verfügen, nutzen wissensbasierte Agenten explizite Repräsentationen von Wissen, um Schlussfolgerungen zu ziehen und neue Fakten abzuleiten.

Komponenten eines wissensbasierten Agenten

Das Herzstück ist die **Knowledge Base (KB)**: Eine Menge von Sätzen (Sentences) in einer formalen Sprache, die Fakten über die Welt repräsentieren.

- **TELL:** Operation zum Hinzufügen neuen Wissens zur KB.
- **ASK:** Operation zum Abfragen von Wissen. Der Agent muss ableiten können, was aus der KB folgt.

1.2 Die Wumpus-Welt

Die Wumpus-Welt ist eine Standardumgebung zur Illustration logischer Agenten.

- **Umgebung:** 4×4 Gitter, Start bei [1,1].
- **Elemente:** Wumpus (stinkt), Gruben (erzeugen Luftzug), Gold (glitzert).
- **Wahrnehmungen (Percepts):** $[Stench, Breeze, Glitter, Bump, Scream]$.
- **Eigenschaften:** Deterministisch, diskret, statisch, partiell beobachtbar.

1.3 Logik: Syntax und Semantik

- **Syntax:** Definiert die zulässigen Sätze (Formeln) der Sprache.
- **Semantik:** Definiert die "Wahrheit" von Sätzen in Bezug auf eine mögliche Welt.
- **Modell (m):** Eine mathematische Abstraktion, die jedem Symbol einen Wahrheitswert zuweist.

Entailment (Logische Folgerung)

Ein Satz α folgt logisch aus der Wissensbasis KB (geschrieben $KB \models \alpha$), wenn in *jedem* Modell, in dem KB wahr ist, auch α wahr ist.

$$M(KB) \subseteq M(\alpha)$$

(Die Menge der Modelle, die die KB erfüllen, ist eine Teilmenge der Modelle, die α erfüllen.)

1.4 Aussagenlogik (Propositional Logic)

Die Aussagenlogik ist die einfachste Form der Logik, basierend auf Fakten, die wahr oder falsch sein können.

1.4.1 Syntax der Aussagenlogik

- **Atomsätze:** Einzelne Symbole (z.B. $P, Q, W_{1,3}$), die für Propositionen stehen.
- **Komplexe Sätze:** Werden durch logische Verknüpfungen gebildet.

Operatoren (nach absteigender Präzedenz):

1. $\neg$ (Negation / Nicht)
2. $\wedge$ (Konjunktion / Und)

3. $\lor$ (Disjunktion / Oder)
4. $\Rightarrow$ (Implikation / Wenn... dann)
5. $\Leftrightarrow$ (Bikonditional / Genau dann, wenn)

1.4.2 Semantik: Wahrheitstabellen

Die Semantik wird durch Wahrheitstabellen definiert.

P	Q	$\neg P$	$P \land Q$	$P \lor Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	**true**	true
false	true	true	false	true	**true**	false
true	false	false	false	true	**false**	false
true	true	false	true	true	true	true

Table 1: *Wahrheitstabelle.* ***Wichtig:*** *Die Implikation $P \Rightarrow Q$ ist nur falsch, wenn die Prämisse P wahr und die Konklusion Q falsch ist.*

1.5 Inferenz (Schlussfolgern)

1.5.1 Model Checking

Ein einfacher Inferenz-Algorithmus ist die **Truth Table Enumeration**: 1. Iteriere über alle möglichen Modelle (Belegungen der Variablen). 2. Prüfe, ob in allen Modellen, in denen die KB wahr ist, auch α wahr ist. Dies ist *sound* (korrekt) und *complete* (vollständig), aber ineffizient ($O(2^n)$).

1.5.2 Logische Eigenschaften

- **Gültigkeit (Tautologie):** Ein Satz ist in *allen* Modellen wahr (z.B. $P \lor \neg P$).
- **Erfüllbarkeit (Satisfiability):** Ein Satz ist in *mindestens einem* Modell wahr.
- **Unerfüllbarkeit (Contradiction):** Ein Satz ist in *keinem* Modell wahr (z.B. $P \land \neg P$).

Wichtiger Zusammenhang (Beweis durch Widerspruch):

$$KB \models \alpha \quad \text{genau dann, wenn} \quad (KB \land \neg\alpha) \text{ ist unerfüllbar.}$$

1.5.3 Logische Äquivalenzen

Zwei Sätze sind äquivalent ($\alpha \equiv \beta$), wenn sie in denselben Modellen wahr sind. Wichtige Umformungen für die Prüfung:

- **De Morgan:** $\neg(P \land Q) \equiv (\neg P \lor \neg Q)$ und $\neg(P \lor Q) \equiv (\neg P \land \neg Q)$
- **Implikations-Eliminierung:** $P \Rightarrow Q \equiv \neg P \lor Q$
- **Bikonditional-Eliminierung:** $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \land (Q \Rightarrow P)$
- **Distributivgesetze:** $(P \land (Q \lor R)) \equiv ((P \land Q) \lor (P \land R))$

1.6 Resolution

Die Resolution ist ein Inferenzverfahren, das Widersprüche aufdeckt. Es arbeitet auf Sätzen in der **Konjunktiven Normalform (CNF)**.

CNF (Conjunctive Normal Form)

Eine Konjunktion von Klauseln. Jede Klausel ist eine Disjunktion von Literalen. Beispiel: $(A \lor \neg B) \land (B \lor C \lor \neg D)$

1.6.1 Umwandlung in CNF

Jeder aussagenlogische Satz kann in CNF umgewandelt werden: 1. Eliminiere $\Leftrightarrow$. 2. Eliminiere $\Rightarrow$ (ersetze $A \Rightarrow B$ durch $\neg A \vee B$). 3. Verschiebe $\neg$ nach innen (De Morgan, doppelte Negation). 4. Wende Distributivgesetz an ($\vee$ über $\wedge$).

1.6.2 Resolutions-Algorithmus

Um zu zeigen, dass $KB \models \alpha$: 1. Füge $\neg\alpha$ zur KB hinzu: $KB \wedge \neg\alpha$. 2. Wandle alles in CNF um. 3. Wende wiederholt die **Resolutionsregel** an:

$$\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

wobei l_i und m_j komplementäre Literale sind (z.B. P und $\neg P$). 4. Wenn die **leere Klausel** (Widerspruch) abgeleitet wird, ist α bewiesen.

1.7 Horn-Klauseln und Chaining

Resolution ist mächtig, aber NP-vollständig. Für eingeschränkte Formen gibt es effizientere Algorithmen (lineare Zeit).

- **Definite Klausel:** Genau ein positives Literal. (Äquivalent zu einer Implikation: $(A \wedge B) \Rightarrow C$).
- **Horn-Klausel:** Höchstens ein positives Literal.

1.7.1 Algorithmen für Definite Klauseln

- **Forward Chaining:** Startet bei den bekannten Fakten in der KB und wendet Regeln an, um neue Fakten zu generieren, bis das Ziel (Query) erreicht ist. (Datengetrieben).
- **Backward Chaining:** Startet beim Ziel (Query) und sucht rückwärts nach Regeln, die dieses Ziel beweisen können. (Zielgetrieben).

Beide basieren auf der *Modus Ponens* Regel: $\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$.

1.8 Grenzen der Aussagenlogik

- **Mangelnde Ausdruckskraft:** Keine Objekte oder Relationen.
- **Regel-Explosion:** Für jede Instanz muss eine eigene Regel geschrieben werden (z.B. $Breeze_{1,1} \Leftrightarrow ...$, $Breeze_{1,2} \Leftrightarrow ...$).
- **Lösung:** Prädikatenlogik erster Stufe (First-Order Logic).