Kapitel 5. Wissensrepräsentation (Teil I)

Übersicht

5.1 Aussagenlogik (AL)

5.2 Kalküle für AL

5.3 Die Wumpus Welt in AL

5.4 Das P=NP Problem

5.5 Prädikatenlogik erster Stufe (PL1)

5.6 Herbrand's Theorem

5.7 Robinson's Resolutionskalkül

5.8 Der Situationskalkül

5.9 Probleme der Wissensrepr.

Die Abschnitte 5.1, 5.2, 5.4, 5.5 und 5.6 sind den Studenten bekannt. Das kann man schnell durchgehen. Da die Terminologie aber etwas anders ist, habe ich es der Vollständigkeit halber mit aufgenommen (damit die Studentem nachschauen koennen).

Zeitplan:

- 1 Doppelstunde fuer alles

1 Einleitung

2 Intelligente Agenten

3 Problem-Lösen durch Suchen

4 Informierte Suche

5 Logik und Wissenrepräsentation

5.1 Aussagenlogik (AL)

5.2 Kalküle für die AL

Definition 5.1 (Axiom, Inferenzregel)

Axiome der Aussagenlogik sind die folgenden Formeln:

- $p \to \top$, $\Box \to p$, $\neg\top \to \Box$, $\Box \to \neg\top$,
- $(p \wedge q) \to p$, $(p \wedge q) \to q$,
- $p \to (p \vee q)$, $q \to (p \vee q)$,
- $\neg\neg p \to p$, $(p \to q) \to ((p \to \neg q) \to \neg p)$,
- $p \to (q \to p)$, $p \to (q \to (p \wedge q))$.

Einzige Inferenzregel der Aussagenlogik ist die folgende Abbildung Modus Ponens

$$MP : Fml \times Fml \to Fml : (\varphi, \varphi \to \psi) \mapsto \psi.$$

Wir schreiben dafür auch kurz

$$\textbf{(MP)} \quad \frac{\varphi,\ \varphi \to \psi}{\psi}.$$

Die Inferenzregel ist so zu verstehen, daß φ, ψ beliebige zusammengesetzte Formeln sein können.

Definition 5.2 (Beweis)
Ein Beweis einer Formel φ aus einer Theorie $T \subseteq Fml_{\mathcal{L}}$ ist eine Folge $\varphi_1, \ldots, \varphi_n$ von Formeln so, daß $\varphi_n = \varphi$ und für alle i mit $1 \leq i \leq n$ gilt eine der folgenden Bedingungen:

- *φ_i ist Substitutions-Instanz eines Axioms,*
- *$\varphi_i \in T$,*
- *Es gibt $\varphi_l, \varphi_k = (\varphi_l \rightarrow \varphi_i)$ mit $l, k < i$. φ_i ist also das Resultat der Anwendung des Modus Ponens auf Vorgänger-Formeln von φ_i.*

Wir schreiben: $T \vdash \varphi$.

Theorem 5.3 (Vollständigkeit)
Eine Formel folgt semantisch aus einer Theorie T genau dann, wenn sie abgeleitet werden kann:

$$T \vdash \varphi \ \ \underline{\text{gdw}} \ \ T \models \varphi$$

Theorem 5.4 (Kompaktheit)
Eine Formel folgt aus einer Theorie T genau dann, wenn sie schon aus einer endlichen Teilmenge von T folgt:

$$Cn(T) = \bigcup \{Cn(T') : T' \subseteq T, T' \text{ endlich}\}.$$

Was ist der Unterschied zwischen einer Inferenzregel $\frac{\varphi}{\psi}$ und der Implikation $\varphi \to \psi$? Angenommen, wir haben eine Formelmenge T gegeben und wir wählen zwei Konstanten $p, q \in \mathcal{L}$. Man könnte doch einerseits

(1) T zusammen mit MP und $\{\frac{p}{q}\}$

und andererseits

(2) $T \cup \{p \to q\}$ zusammen mit MP

betrachten. Im ersten Fall bedeutet dies

$$Cn^{\{\frac{p}{q}\}}(T)$$

und im zweiten Fall

$$Cn(T \cup \{p \to q\}).$$

Ein Resolutionskalkül für AL

Gegeben sei eine Menge M von Klauseln der Form

$$A \vee \neg B \vee C \vee \ldots \vee \neg E$$

Solch eine Klausel schreiben wir auch als Menge

$$\{A, \neg B, C, \ldots, \neg E\}.$$

Wir definieren die folgende Inferenzregel:

Definition 5.5 (AL-Resolution)
Leite aus $C_1 \cup \{A\}$ und $C_2 \cup \{\neg A\}$ die Klausel $C_1 \cup C_2$ ab.

Ist dieser Kalkül korrekt und vollständig?

Jedes Problem der Form "$T \vdash \phi$" ist äquivalent zu

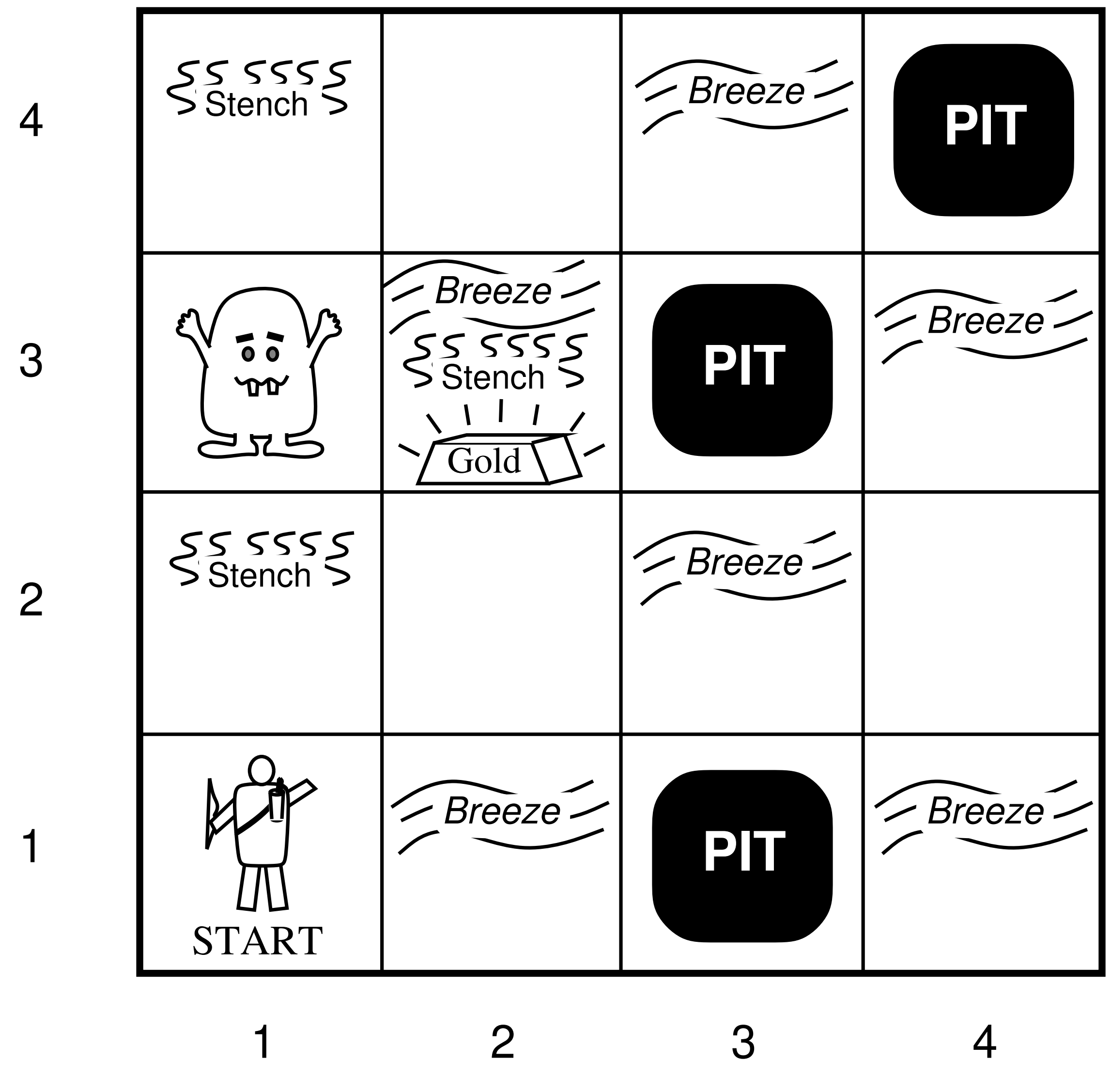
"$T \cup \{\neg\phi\}$ ist unerfüllbar"

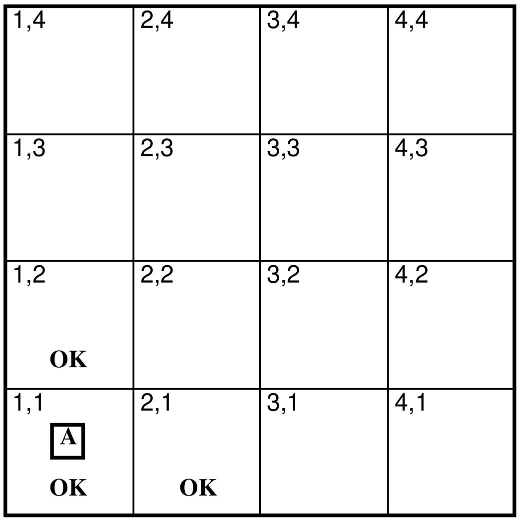
bzw. zu

$$T \cup \{\neg\phi\} \vdash \square.$$

Theorem 5.6 (Widerlegungsvollständigkeit)
Ist M eine unerfüllbare Menge von Klauseln, so kann aus M allein mit der Resolutionsregel die leere Klausel $\square$ abgeleitet werden.

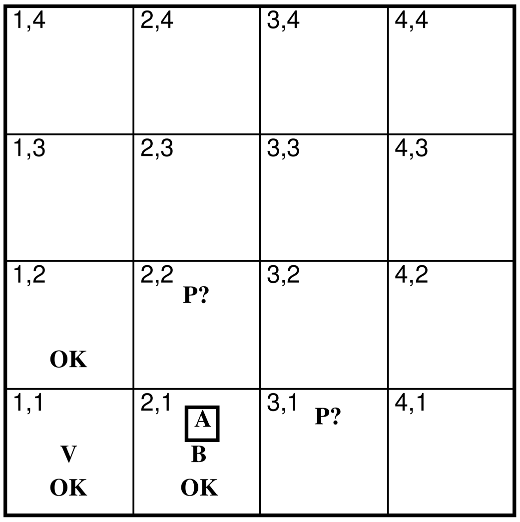
5.3 Die Wumpus-Welt in AL



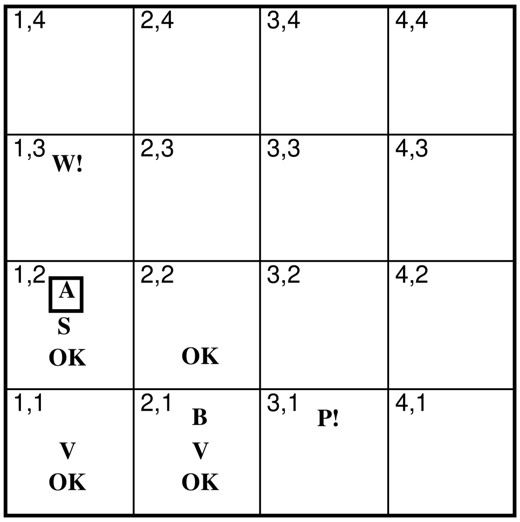


(a)

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

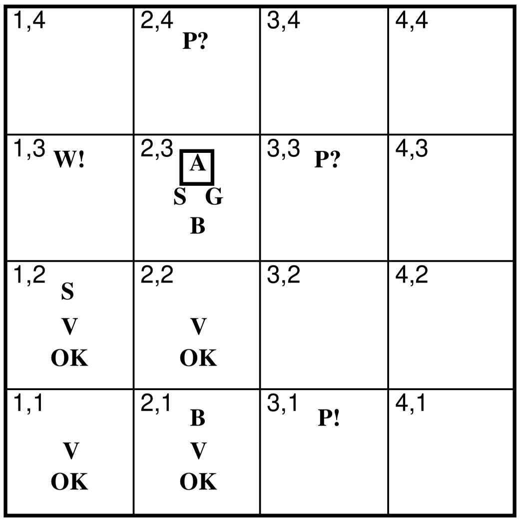


(b)



(a)

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*



(b)

Festlegung der Sprache:

$S_{i,j}$	stinkt
$B_{i,j}$	ist kalt
$Grube_{i,j}$	ist eine Grube
$Gl_{i,j}$	glitzert
$W_{i,j}$	enthält Wumpus

Allgemeines Wissen:

$$\neg S_{1,1} \longrightarrow (\neg W_{1,1} \land \neg W_{1,2} \land \neg W_{2,1})$$
$$\neg S_{2,1} \longrightarrow (\neg W_{1,1} \land \neg W_{2,1} \land \neg W_{2,2} \land \neg W_{3,1})$$
$$\neg S_{1,2} \longrightarrow (\neg W_{1,1} \land \neg W_{1,2} \land \neg W_{2,2} \land \neg W_{1,3})$$

$$S_{1,2} \longrightarrow (W_{1,3} \land W_{1,2} \land W_{2,2} \land W_{1,1})$$

Wissen nach dem 3. Zug:

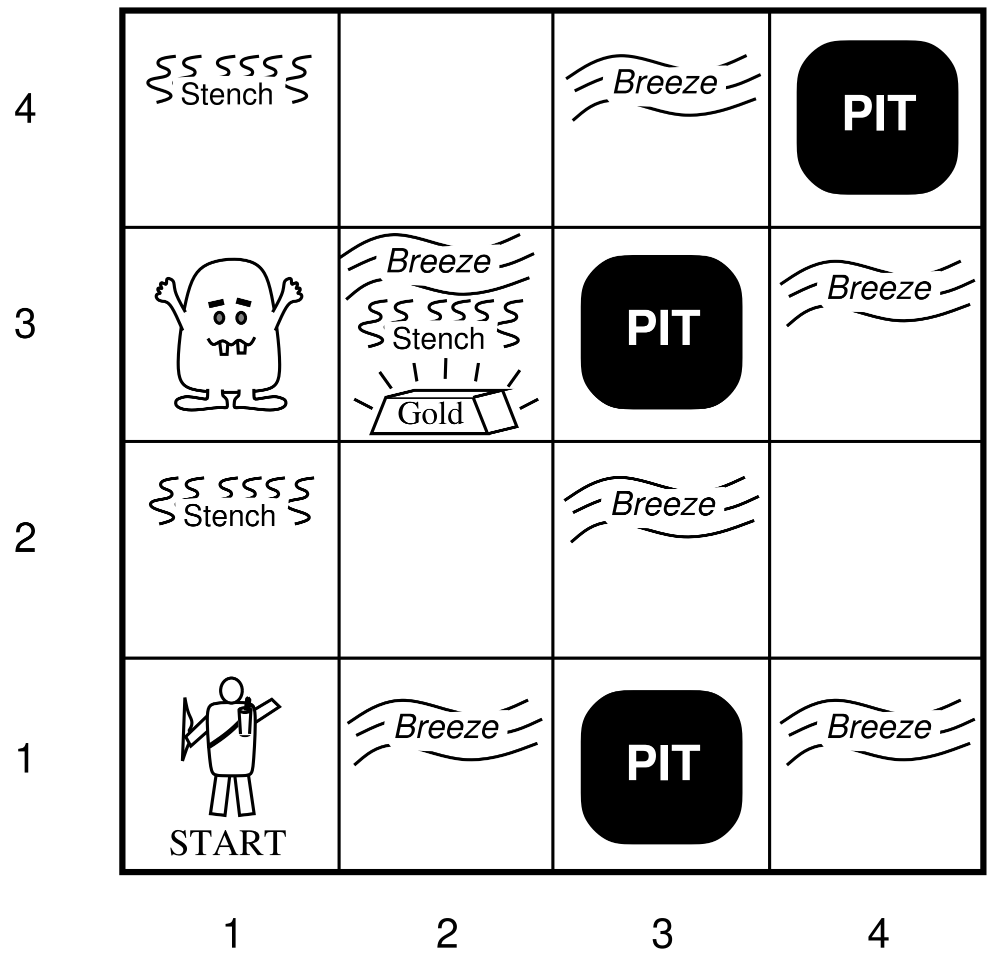
$$\neg S_{1,1} \wedge \neg S_{2,1} \wedge S_{1,2} \wedge \neg B_{1,1} \wedge \neg B_{2,1} \wedge \neg B_{1,2}$$

Können wir herleiten, dass der Wumpus in (1,3) sitzt? Ja, etwa mit Resolution oder unserem Hilbert-Kalkül.

Eigentlich wollen wir mehr: zu gegebener Situation möchten wir gerne die **beste Aktion** auswählen. Das aber geht in AL nicht. Jedoch können wir für jede Aktion prüfen, ob sie ausgeführt werden sollte oder nicht. Dazu benötigen wir zusätzliche Axiome:

Zusätzliche Axiome:

$$\begin{array}{lcl}
A_{1,1} \wedge East \wedge W_{2,1} & \longrightarrow & \neg Forward \\
A_{1,1} \wedge East \wedge Grube_{2,1} & \longrightarrow & \neg Forward \\
A_{i,j} \wedge Gl_{i,j} & \longrightarrow & Take_{Gold}
\end{array}$$



4
3
2
1
Stench
Breeze
PIT
Breeze
Stench
Gold
PIT
Breeze
Stench
Breeze
START
Breeze
PIT
Breeze
1
2
3
4

Nachteile:

- Aktionen können nur geraten werden,
- Datenbasis muss fortlaufend geändert werden,
- Regelmenge werden sehr gross, da keine Variablen vorhanden.

Bei geeigneter Formalisierung (zusätzliche Axiome) kann man jedenfalls prüfen ob

$$\mathrm{KB} \vdash \neg\, \mathrm{aktion} \quad \text{bzw.} \quad \mathrm{KB} \vdash \mathrm{aktion}$$

Es kann auch sein, dass weder das Eine noch das Andere ableitbar ist.

5.4 Das P=NP Problem

Beobachtung: Polynomiale Algorithmen sind schnell, exponentielle dagegen langsam.

Um eine AL-Formel mit der Wahrheitstafel-Methode als unerfüllbar nachzuweisen, braucht man im schlimmsten Fall 2^n Belegungen (bei n-Konstanten).

Kann man dieses Problem vielleicht mit anderen Methoden in polynomialer Zeit lösen?

Wir betrachten Ja-Nein Probleme. Für jede Instanz $\mathcal{I}$ eines Problems können wir fragen ob $\mathcal{I}$ die entsprechende Eigenschaft hat oder nicht: $\mathcal{I}$ ist Ja oder Nein-Instanz.

Definition 5.7 (Klasse P)
Ein Problem liegt in der Klasse **P** *der in polynomialer deterministischer Zeit erkennbaren Probleme, falls es einen Algorithmus* Alg. *und ein Polynom $p(n)$ gibt, so dass für jede Instanz $\mathcal{I}$ gilt:*

$\mathcal{I}$ *ist Ja-Instanz* *gdw* Alg. *antwortet "Ja" auf Eingabe von $\mathcal{I}$ nach höchstens $p($länge$(\mathcal{I}))$-Schritten.*

Definition 5.8 (Klasse NP)
Ein Problem liegt in der Klasse **NP** *der in polynomialer nicht-deterministischer Zeit erkennbaren Probleme, falls es einen Algorithmus* Check *und ein Polynom $p(n)$ gibt so dass für jede Instanz $\mathcal{I}$ gilt:*

$\mathcal{I}$ ist Ja-Instanz	*gdw*	*es gibt eine Instanz $\mathcal{I}'$ eines anderen Problems mit*
		(1) länge($\mathcal{I}'$) $<$ p(länge($\mathcal{I}$)),
		(2) Check *antwortet "Ja" auf Eingabe von $\mathcal{I}'$*
		nach höchstens p(länge($\mathcal{I}$))-Schritten.

Man sieht leicht, dass die Erfüllbarkeit einer AL-Formel in **NP** liegt, und dass **NP** jedenfalls **P** umfasst.

Was sind die schwierigsten Probleme in **NP**?

Offenbar sind Probleme die man polynomial aufeinander reduzieren kann gleichwertig. Ein Problem Prob_1 ist polynomial transformierbar zu einem Problem Prob_2, falls man für jede Instanz $\mathcal{I}_1$ von Prob_1 in polynomialer Zeit eine Instanz $\mathcal{I}_2$ von Prob_2 berechnen können, so dass gilt:

$$\mathcal{I}_1 \text{ ist Ja-Instanz von } \text{Prob}_1$$

genau dann wenn

$$\mathcal{I}_2 \text{ ist Ja-Instanz von } \text{Prob}_2$$

Definition 5.9 (NP-Vollständigkeit)
Ein Problem aus **NP** *heisst NP-vollständig, wenn sich alle anderen Probleme aus* **NP** *in polynomialer Weise auf es reduzieren lassen.*

Folgende Probleme sind NP-vollständig:

- Gegeben ein endlicher Graph und $k \in \mathbb{N}$. Kann der Graph mit k-Farben gefärbt werden?
- Gegeben ein endlicher Graph. Gibt es einen Hamilton-Kreis?
- Gegeben eine AL-Formel ϕ. Ist ϕ erfüllbar?
- Gegeben eine Menge von Klauseln in den Konstanten $c_1, \ldots, c_n$. Ist die Menge erfüllbar?
- Gegeben eine Menge von Klauseln in den Konstanten $c_1, \ldots, c_n$. Jede Klausel enthält höchstens 3 Literale. Ist die Menge erfüllbar?

5.5 Prädikatenlogik erster Stufe (PL1)

5.6 Herbrand's Theorem

Die bisher eingeführte Relation $T \models \phi$ sagt aus, daß jedes Modell von T auch eines von ϕ ist. Da es aber sehr viele Modelle mit sehr grossen Universen gibt, stellt sich die Frage, ob man das nicht einschränken kann.

Theorem 5.10 (Löwenheim-Skolem)
$T \models \phi$ *gilt genau dann, wenn* ϕ *in allen abzählbaren Modellen von* T *gilt.*

Oftmals bestehen die Universen der Modelle die uns eigentlich interessieren genau aus den Grundtermen $Term_{\mathcal{L}}(\emptyset)$. Dies führt uns zu dem folgenden Begriff:

Definition 5.11 (Herbrand-Modell)
Ein Modell $\mathcal{A}$ heißt Herbrand-Modell bezüglich einer Sprache $\mathcal{L}$, falls das Universum von $\mathcal{A}$ genau aus $Term_{\mathcal{L}}(\emptyset)$ besteht und die Funktionssymbole f_i^k wie folgt interpretiert werden:

$$f_i^{k\,\mathcal{A}} : Term_{\mathcal{L}}(\emptyset) \times \ldots \times Term_{\mathcal{L}}(\emptyset) \; \to Term_{\mathcal{L}}(\emptyset); \; (t_1, \ldots, t_k) \mapsto f_i^k(t_1, \ldots, t_k)$$

Wir schreiben $T \models_{Herb} \phi$ falls jedes Herbrand Modell von T auch Modell von ϕ ist.

Theorem 5.12
Ist T universell und ϕ existentiell, so gilt

$$T \models_{Herb} \quad \textit{genau dann wenn} \quad T \models_{Herb} \phi$$

Frage: Ist $T \models_{Herb} \phi$ nicht viel einfacher, da man nur Herbrand-Modelle beachten muß? Ist es vielleicht sogar entscheidbar? Mehr darüber in Kapitel 6.

Der folgende Satz von Herbrand ist die Grundlage der Resolution. Er führt, sozusagen, PL1 auf AL zurück.

Theorem 5.13 (Herbrand)
Sei T universell und ϕ enthalte keine Quantoren. Dann gilt:

$$T \models \exists \phi \quad \textit{gdw} \quad \textit{es gibt } t_1, \ldots, t_n \in Term_{\mathcal{L}}(\emptyset)$$
$$\textit{mit: } T \models \phi(t_1) \vee \ldots \vee \phi(t_n)$$

Oder: Sei M eine Menge von Klauseln der PL1 (also Formeln der Form $P_1(t_1) \vee \neg P_2(t_2) \vee \ldots \vee P_n(t_n)$ wobei $t_i \in Term_{\mathcal{L}}(X)$). Dann ist M unerfüllbar genau dann, wenn es eine endliche unerfüllbare Menge M_{inst} von Grundinstanzen von M gibt.

Im automatischen Beweisen interessiert uns immer die Frage

$$M \models \exists x_1, \ldots x_n \ \bigwedge_i \phi_i$$

Dann ist $M \cup \{\neg \exists x_1, \ldots x_n \ \bigwedge_i \phi_i\}$ eine Menge von Klauseln, die genau dann unerfüllbar ist, wenn die obige Relation gilt.

5.7 Robinson's Resolution

Unifikation: Gegeben sei eine endliche Menge von Gleichungen zwischen Termen oder Gleichungen zwischen Literalen. Dann gibt es einen Algorithmus der entweder eine allgemeinste Lösungssubstitution berechnet (d.h. eine Substitution für die beteiligten Variablen derart, daß linke und rechte Seiten aller Gleichungen syntaktisch identisch werden) oder *fail* zurückgibt. Im letzten Fall gibt es gar keine Lösungssubstitution. Im ersten Fall ist die allgemeinste Lösungssubstitution bis auf Variablenumbenennungen eindeutig bestimmt: sie heißt

mgU, *allgemeinster Unifikator*

- $p(x, a) = q(y, b)$,
- $p(g(a), f(x)) = p(g(y), z)$. Grundsubstitutionen sind: $[y/a, x/a, z/f(a)]$, $[y/a, x/f(a), z/f(f(a))], \ldots$
  Der mgU lautet: $[y/a, z/f(x)]$.

Wir wollen den auf der vorigen Folie erwähnten Algorithmus nun anhand eines Beispiels kurz skizzieren.

$$\text{Gegeben: } f(x, g(h(y), y)) = f(x, g(z, a))$$

Der Algorithmus berechnet nacheinander folgende Mengen von Gleichungen:

$$\begin{array}{l}
\{\ x = x,\ \ g(h(y), y) = g(z, a)\ \} \\
\{\ g(h(y), y) = g(z, a)\ \} \\
\{\ h(y) = z,\ \ y = a\ \} \\
\{\ z = h(y),\ \ y = a\ \} \\
\{\ z = h(a),\ \ y = a\ \}
\end{array}$$

Damit ergibt sich der mgU als: $[x/x, y/a, z/h(a)]$.

Definition 5.14 (Robinson's Resolution für PL1)
Die Resolution besteht aus zwei Regeln

$$\textit{Resolution: } \frac{C_1 \cup \{A_1\} \quad C_2 \cup \{\neg A_2\}}{(C_1 \cup C_2) mgU(A_1, A_2)}$$

wobei $C_1 \cup \{A_1\}$ *und* $C_2 \cup \{A_2\}$ *als variablendisjunkt vorausgesetzt werden.*

$$\textit{Faktorisierung: } \frac{C_1 \cup \{L_1, L_2\}}{(C_1 \cup \{L_1\}) mgU(L_1, L_2)}$$

Als Beispiel betrachte man $M = \{r(x) \vee \neg p(x), p(a), s(a)\}$ und die Frage $M \models \exists x(s(x) \wedge r(x))$?

Warum braucht man Faktorisierung? Man betrachte

$$M = \{s(x_1) \vee s(x_2), \neg s(y_1) \vee \neg s(y_2)\}$$

Resolviert man beide Klauseln miteinander, erhält man

$$\{s(x_1)\} \cup \{\neg s(y_1)\}$$

oder Varianten davon. Resolviert man diese neue Klausel mit eine der beiden in M, erhält man nur Varianten der entsprechenden Klausel in M.

Also ist $\square$ niemals ableitbar (mit Resolution allein).

Faktorisierung löst das Problem sofort, denn man kann sowohl $s(x)$ als auch $\neg s(y)$ ableiten, und daraus die leere Klausel.

5.8 Der Situationskalkül

Wie axiomatisieren wir die Wumpus-Welt in PL1?

```
function KB-AGENT(percept) returns an action
   static: KB, a knowledge base
           t, a counter, initially 0, indicating time

   TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
   action ← ASK(KB, MAKE-ACTION-QUERY(t))
   TELL(KB, MAKE-ACTION-SENTENCE(action, t))
   t ← t + 1
   return action
```

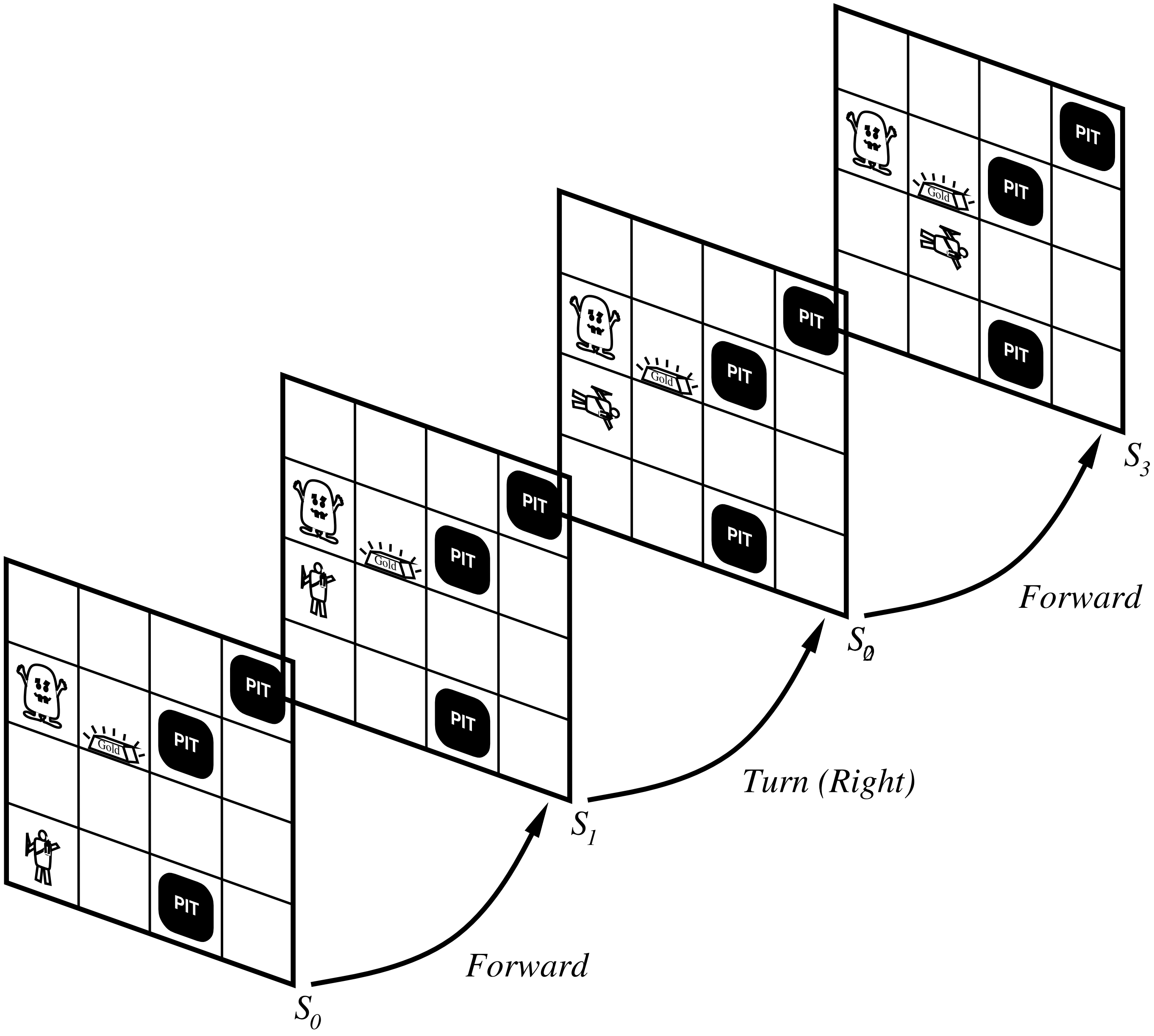
Idee: Um Aktionen, bzw. deren Effekte konsistent zu beschreiben, stellen wir die Welt als Folge von Situationen (Schnappschüsse der Welt) dar. Dazu müssen wir jedes Prädikat um ein zusätzliches Argument erweitern.

Wir benutzen ein Funktionssymbol

$$result(aktion, situation)$$

als Term für die Situation die ensteht, wenn in $situation$ die Aktion $aktion$ ausgeführt wird.

Aktionen: $Turn\_right$, $Turn\_left$, $Foreward$, $Shoot$, $Grab$, $Release$, $Climb$.


PIT
Gold
Forward
Turn (Right)
Forward
S_0
S_1
S_2
S_3

Wir brauchen auch ein Gedächtnis, in Form eines 3-stelligen Prädikates

$$At(person, location, situation)$$

wobei $person$ entweder $Wumpus$ oder der $Agent$ ist und $location$ für den momentanen Ort, codiert als Paar [i,j] steht.

Wichtige Axiome sind die sogenannten "Successor-state Axioms", die beschreiben, welche Auswirkungen Aktionen auf Situationen haben. Die allgemeine Form dieser Axiome ist

$$\begin{array}{lcl} \text{true afterwards} & \Longleftrightarrow & \text{an action made it true} \\ & & \text{or it is already true and} \\ & & \text{no action made it false} \end{array}$$

Axiome über $At(p, l, s)$:

$$\begin{array}{lcl} At(p, l, result(a, s)) & \leftrightarrow & ((l = location\_ahead(p, s) \wedge \neg Wall(l) \wedge a = Forward) \\ & & \quad \vee (At(p, l, s) \wedge \neg a = Forward)) \end{array}$$

$$\begin{array}{lcl} At(p, l, s) & \rightarrow & Location\_ahead(p, s) = Location\_toward(l, Orient.(p, s)) \\ Wall([x, y]) & \leftrightarrow & (x = 0 \vee x = 5 \vee y = 0 \vee y = 5) \end{array}$$

$$Location\_toward([x, y], 0) = [x + 1, y]$$
$$Location\_toward([x, y], 90) = [x, y + 1]$$
$$Location\_toward([x, y], 180) = [x - 1, y]$$
$$Location\_toward([x, y], 270) = [x, y - 1]$$

$$Orient.(Agent, s_0) = 90$$
$$\begin{aligned} Orient.(p, result(a, s)) = d \quad \leftrightarrow \quad & \big((a = turn\_right \wedge d = mod(Orient.(p, s) - 90, 360)) \\ & \vee (a = turn\_left \wedge d = mod(Orient.(p, s) + 90, 360)) \\ & \vee (Orient.(p, s) = d \wedge \neg(a = Turn\_right \vee a = Turn\_left)\big) \end{aligned}$$

Dabei ist $mod(x, y)$ die eingebaute "modulo"-Funktion, die jedes x auf einen eindeutigen Wert zwischen 0 und y setzt.

Axiome über Wahrnehmungen, sinnvolle neue Begriffe durch Definitionen:

$$
\begin{aligned}
Percept([Stench, b, g, u, c], s) &\rightarrow Stench(s) \\
Percept([a, Breeze, g, u, c], s) &\rightarrow Breeze(s) \\
Percept([a, b, Glitter, u, c], s) &\rightarrow At\_Gold(s) \\
Percept([a, b, g, Bump, c], s) &\rightarrow At\_Wall(s) \\
Percept([a, b, g, u, Scream], s) &\rightarrow Wumpus\_dead(s)
\end{aligned}
$$

$$
\begin{aligned}
At(Agent, l, s) \land Breeze(s) &\rightarrow Breezy(l) \\
At(Agent, l, s) \land Stench(s) &\rightarrow Smelly(l)
\end{aligned}
$$

$$\begin{array}{lcl} Adjacent(l_1, l_2) & \leftrightarrow & \exists\, d\ l_1 = Location\_toward(l_2, d) \\ Smelly(l_1) & \rightarrow & \exists l_2\ At(Wumpus, l_2, s) \wedge (l_2 = l_1 \vee Adjacent(l_1, l_2)) \end{array}$$

$$\begin{array}{lcl} Percept([none, none, g, u, c], s) \wedge At(Agent, x, s) \wedge Adjacent(x, y) & \rightarrow & OK(y) \\ (\neg At(Wumpus, x, t) \wedge \neg Pit(x)) & \rightarrow & OK(y) \end{array}$$

$$\begin{array}{lcl} At(Wumpus, l_1, s) \wedge Adjacent(l_1, l_2) & \rightarrow & Smelly(l_2) \\ At(Pit, l_1, s) \wedge Adjacent(l_1, l_2) & \rightarrow & Breezy(l_2) \end{array}$$

Axiome zur Beschreibung von Aktionen:

$$
\begin{array}{lcl}
Holding(Gold, result(Grab, s)) & \leftrightarrow & (At\_Gold(s) \vee Holding(Gold, s)) \\
Holding(Gold, result(Release, s)) & \leftrightarrow & \Box \\
Holding(Gold, result(Turn\_right, s)) & \leftrightarrow & Holding(Gold, s) \\
Holding(Gold, result(Turn\_left, s)) & \leftrightarrow & Holding(Gold, s) \\
Holding(Gold, result(Forward, s)) & \leftrightarrow & Holding(Gold, s) \\
Holding(Gold, result(Climb, s)) & \leftrightarrow & Holding(Gold, s)
\end{array}
$$

Alle Effekte müssen sorgsam beschrieben werden.

Axiome die Präferenzen zwischen Aktionen beschreiben:

$$
\begin{array}{lcl}
Great(a,s) & \rightarrow & Action(a,s) \\
(Good(a,s) \land \neg\exists b\, Great(b,s)) & \rightarrow & Action(a,s) \\
(Medium(a,s) \land \neg\exists b\, (Great(b,s) \lor Good(b,s))) & \rightarrow & Action(a,s) \\
(Risky(a,s) \land \neg\exists b\, (Great(b,s) \lor Good(b,s) \lor Medium(a,s))) & \rightarrow & Action(a,s)
\end{array}
$$

$$
\begin{array}{lcl}
At(Agent,[1,1],s) \land Holding(Gold,s) & \rightarrow & Great(Climb,s) \\
At\_Gold(s) \land \neg Holding(Gold,s) & \rightarrow & Great(Grab,s) \\
At(Agent,l,s) \land \neg Visited(Location\_ahead(Agent,s)) \land & & \\
\land OK(Location\_ahead(Agent,s)) & \rightarrow & Good(Forward,s)
\end{array}
$$

$$
\begin{array}{lcl}
Visited(l) & \leftrightarrow & \exists s\, At(Agent,l,s)
\end{array}
$$

Eigentlich geht es nicht nur darum das Gold zu finden, sondern auch wieder lebend zurückzukehren. Man bsaucht also noch Axiome wie $Holding(Gold,s) \rightarrow Go\_back(s)$ usw.

5.9 Wichtige Probleme der Wissensrepräsentation

Es haben sich 3 sehr wichtige Repräsentationsprobleme bei der Axiomatisierung einer sich ändernden Welt herausgestellt:

Frame-Problem: *Aktionen ändern meist nur sehr wenig. Man braucht sehr viele Aktionen um invariante Eigenschaften zu beschreiben.*

Ideal wäre es, nur was sich **nicht ändert** zu axiomatisieren und dann eine Aussage der Form "Sonst ändert sich nichts" hinzuzunehmen.

Qualification-Problem: *Das Aufzählen aller Bedingungen, unter denen eine Aktion Erfolg hat, ist nötig.* Z.B.

$$\begin{array}{lll} \forall x & (Vogel(x) & \wedge\neg Pinguin(x) \wedge \neg Tot(x)\wedge \\ & & \wedge\neg Strauss(x) \wedge \neg Flügel\_gebr.(x)\wedge \\ & & \wedge \ldots \\ & \longrightarrow Fliegt(x) & \end{array}$$

Ideal wäre es, nur “Normalerweise fliegen Vögel” aufzuschreiben.

Ramification-Problem: *Wie sollen implizite Konsequenzen von Aktionen gehandhabt werden?* Z.B $Grab(Gold)$ $Gold$ kann radioaktiv verseucht sein. Die Aktion $Grab(Gold)$ ist dann jedenfalls nicht optimal.

Programmieren versus Knowledge Engineering.

Programmieren	Knowledge Engineering
Wähle Prog.-sprache	Wähle Logik
Schreibe Programm	Definiere KB
Schreibe Compiler	Implementiere Kalkül
Lass Programm laufen	Leite neue Fakten ab

Aus Wittgenstein's Tractatus

1. Die Welt ist alles was der Fall ist.

   1.1 Die Welt ist die Gesamtheit der Tatsachen, nicht der Dinge.

2. Was der Fall ist, die Tatsache, ist das Bestehen von Sachverhalten.

3. Das logische Bild der Tatsachen ist der Gedanke.

4. Der Satz ist eine Wahrheitsfunktion der Elementarsätze.

5. Die allgemeine Form der Wahrheitsfunktion ist: $[\overline{p}, \overline{\xi}, N(\overline{\xi})]$.

6. Wovon man nicht sprechen kann, darüber muß man schweigen.