Grundlagen der Künstlichen Intelligenz

8 PL-1-Inferenz und Situationskalkül für logische Agenten

Modus Ponens und Resolution in PL-1, Kontrollstrategien,
Situationskalkül, Modellierung von Aktionen, Zeit und Raum

Volker Steinhage

Inhalt

- Generalisierter Modus Ponens
- Forward Chaining und Backward Chaining
- Binäre Resolution und Faktorisierung
- Vollständige Resolution
- Logische Agenten für die WUMPUS-Welt
- Situationskalkül
  - Beschreibungen von Aktionen: das Rahmenproblem
  - Modellierung von Raum und Bewegungen
  - Kausale & diagnostische Regeln
  - Aktionsauswahl

Wiederholung: Generalized Modus Ponens (GMP)

Mit Hilfe der Unifikation können wir den Generalisierten Modus Ponens als Inferenzmethode für die Prädikatenlogik nutzen:

Beispiel-KB in KNF:

King(x) $\wedge$ *Greedy(x)* $\Rightarrow$ *Evil(x)*	{ { *-King(x), -Greedy(x), Evil(x)* },
King(john)	{ *King(john)* },
Greedy(y)	{ *Greedy(y)* },
Brother(richard, john)	{ *Brother(richard, john)* } }

Satz: Auf prädikatenlogischen definiten Klauselmengen arbeitet der verallgemeinerte Modus Ponens korrekt, vollständig und effizient:

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

mit $p_i' \theta = p_i \theta$ für alle i.

Beispiel:

p_1'	ist	*King(john)*	p_1	ist	*King(x)*
p_2'	ist	*Greedy(y)*	p_2	ist	*Greedy(x)*
θ	ist	$\{x/john, y/john\}$	q	ist	*Evil(x)*
$q\theta$	ist	*Evil(john)*			

Strategien für GMP

Geg. sei: *GMP* als Inferenzoperation, Wissensbasis *KB*, Anfrage *q*.

Frage: welche Strategie ist einzusetzen, um *q* aus *KB* effizient abzuleiten?

Für die Diskussion dazu dient das folg. Beispiel.

Wissensbasis *KB* informell:

> *The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.*

Anfrage bzw. Behauptung *q*:

> *Prove that Colonel West is a criminal*

Beispiel für GMP

... it is a crime for an American to sell weapons to hostile nations:

➡ $American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$

Nono ... has some missiles, i.e., $\exists\, x\; Owns(Nono, x) \wedge Missile(x)$:

➡ $Owns(Nono,M1), Missile(M1)$

... all of its missiles were sold to it by Colonel West

➡ $Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West, x, Nono)$

Missiles are weapons:

➡ $Missile(x) \Rightarrow Weapon(x)$

An enemy of America counts as „hostile“:

➡ $Enemy(x, America) \Rightarrow Hostile(x)$

West, who is American ...

➡ $American(West)$

The country Nono, an enemy of America ...

➡ $Enemy(Nono, America)$

Einfache Vorwärtsableitung mit GMP (1)

Prinzip der Vorwärtsableitung:

von den Fakten der Wissensbasis zur Anfrage!

Genauer:

First-order logic forward chaining

Die Funktion FOL-FC-ASK mit

- Eingabe: Wissensbasis *KB* und atomare Aussage α als Anfrage
- Ausgabe: Unifizierende Substitution oder *false*
- Prozess: iterative Anwendung der Regelklauseln auf Faktenklauseln solange, bis Anfrage α abgeleitet wird oder keine neuen Ableitungen erzeugbar sind.
- Prinzip der Breitensuche.

Einfache Vorwärtsableitung mit GMP (2)

function FOL-FC-ASK(KB, α) **returns** a substitution or $false$

inputs: KB, the knowledge base, a set of first-order definite clauses

α, the query, an atomic sentence

local variables: new, the new sentence inferred on each iteration

repeat until new is empty

$new \leftarrow \{\}$

for each sentence r **in** KB **do**

$(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)

Standardisierung

for each θ such that SUBST($\theta, p_1 \wedge p_2 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge p'_2 \wedge \ldots \wedge p'_n$)

for some $p'_1, p'_2, \ldots, p'_n$ in KB

Suche in KB die p'_i, die zu den Prämissenanteilen p_i der Regel r passen

$q' \leftarrow$SUBST(θ, q)

if q' is not a renaming of some sentence already in KB or new **then do**

add q' to new

$\phi \leftarrow$ UNIFY (q', α)

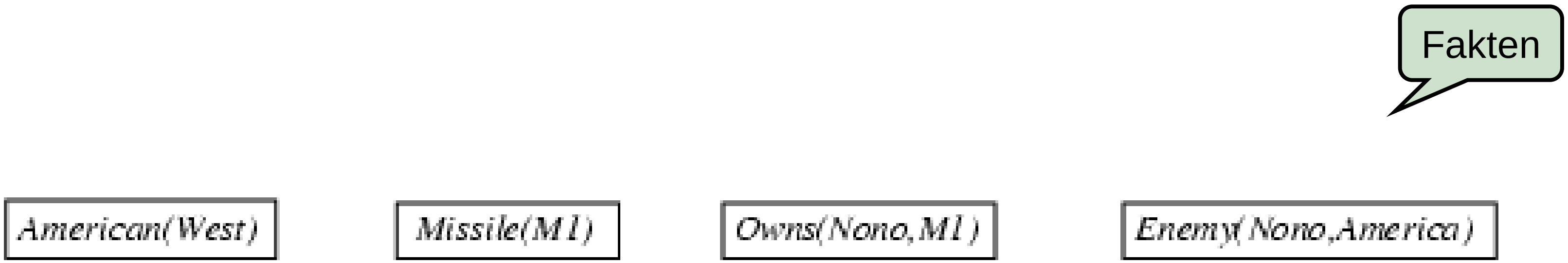
Zieltest

if ϕ is not $fail$ **then return** ϕ

add new to KB

return *false*

Beweis mit einfacher Vorwärtsableitung (1)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

American(x) $\wedge$ *Weapon(y)* $\wedge$ *Sells(x, y, z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x),*

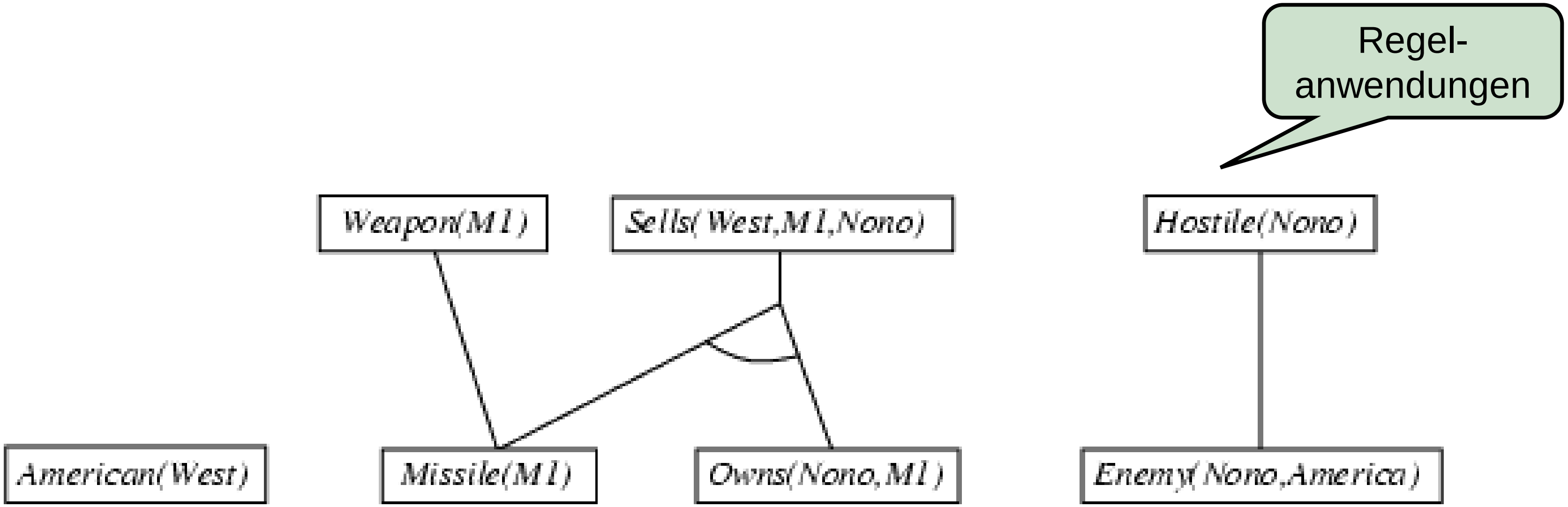
Missile(x) $\wedge$ *Owns(Nono, x)* $\Rightarrow$ *Sells(West,x,Nono),*

Missile(x) $\Rightarrow$ *Weapon(x),*

Enemy(x, America) $\Rightarrow$ *Hostile(x).*

α = *Criminal(West)*

Beweis mit einfacher Vorwärtsableitung (2)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

$American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal(x)$,

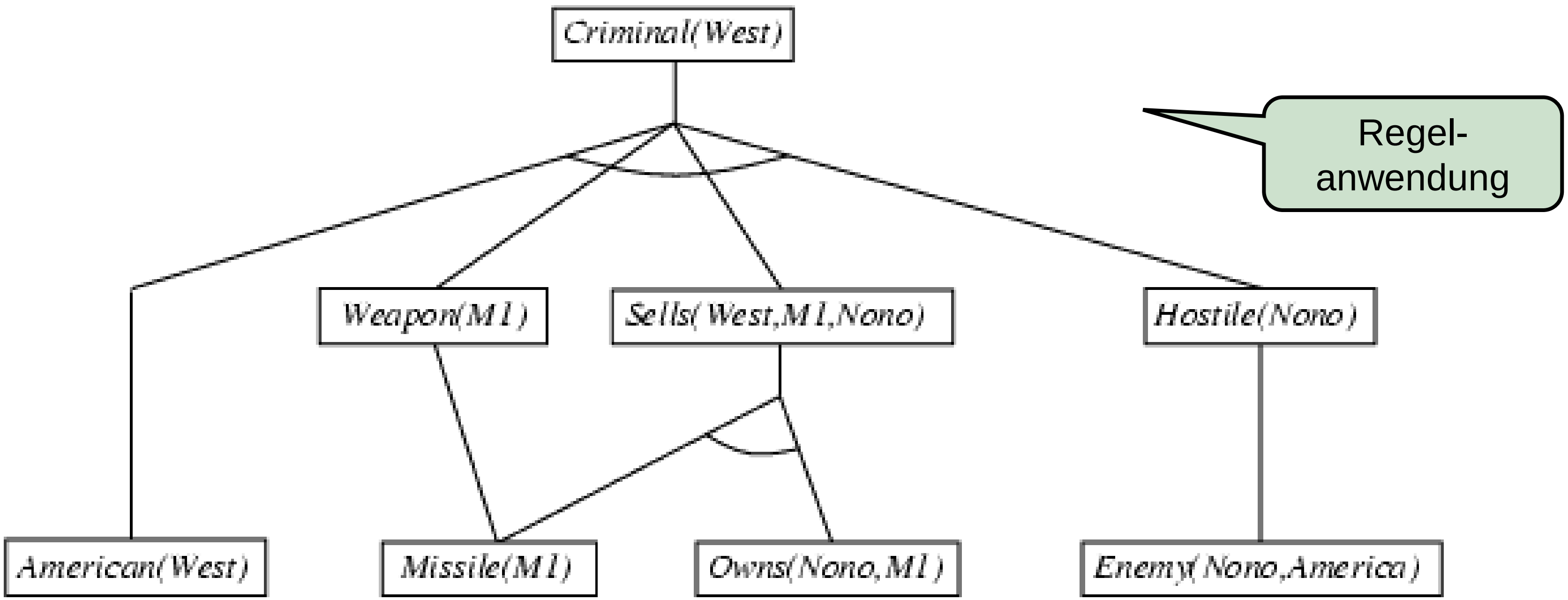
$Missile(x_1) \wedge Owns(Nono, x_1) \Rightarrow Sells(West, x_1, Nono)$,

$Missile(x_2) \Rightarrow Weapon(x_2)$,

$Enemy(x_3, America) \Rightarrow Hostile(x_3)$.

$\alpha = Criminal(West)$

Beweis mit einfacher Vorwärtsableitung (3)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

$American(x_4) \wedge Weapon(y_4) \wedge Sells(x_4, y_4, z_4) \wedge Hostile(z_4) \Rightarrow Criminal(x_4)$,

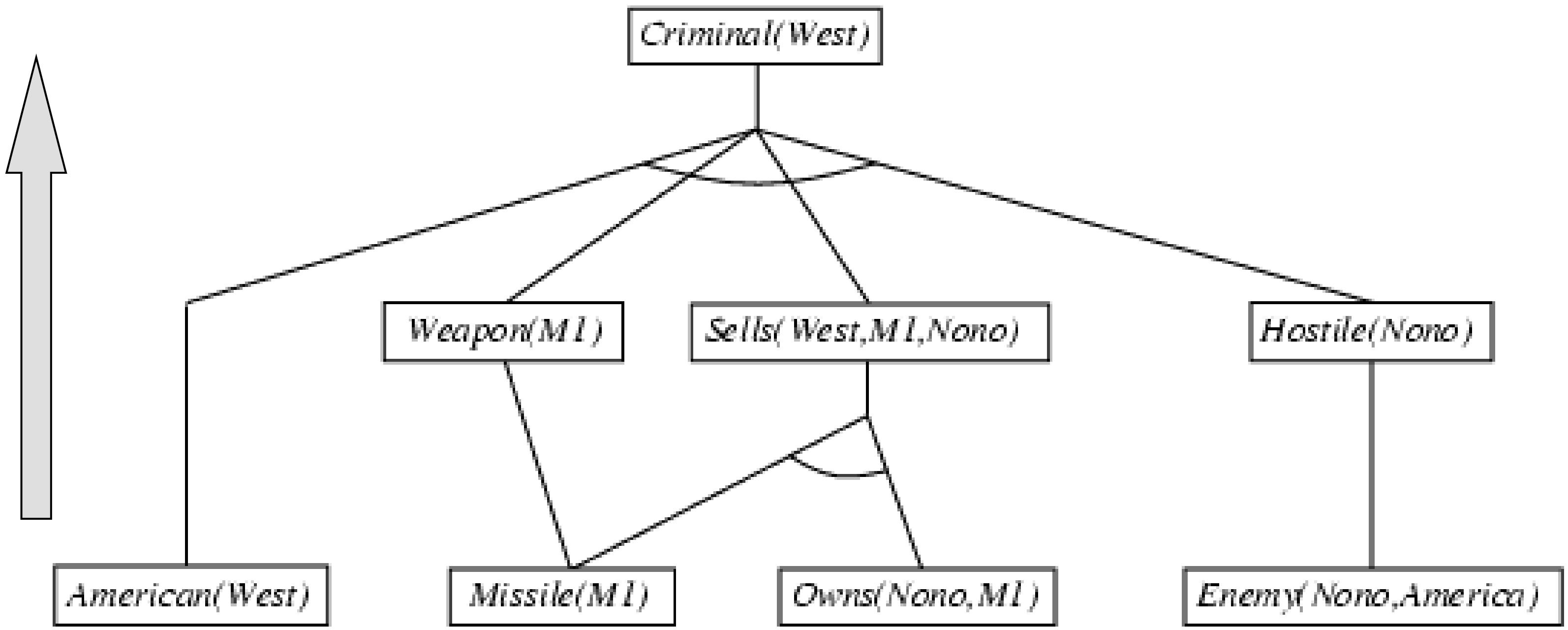
$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West,x,Nono)$,

$Missile(x) \Rightarrow Weapon(x)$,

$Enemy(x, America) \Rightarrow Hostile(x)$.

$\alpha = Criminal(West)$

Beweis mit einfacher Vorwärtsableitung (4)



Ableitung der Anfrage in 2. Iteration.

Ab 2. Iteration auch keine weitere Ableitung möglich!

→ KB in 2. Iteration heißt *Fixpunkt* der Ableitung

Effizienz von einfacher Vorwärtsableitung

- *Datalog* = prädikatenlog. definite Klauseln *ohne echte Funktionen* wie z.B. in der Wissensbasis Crime-KB: $\rightarrow$ für Datalog polynom. Zeitkomplexität.

- Allg. für definite Klauselmengen in PL1 unentscheidbar: Z.B. führen *NatNum(0)* und *NatNum(n)* $\Rightarrow$ *NatNum(s(n))* zu Ableitungen *NatNum(s(0)), NatNum(s(s(0)))*, ...

- Regelauswahl in innerer Schleife ist NP-hart: Vergleich aller Prämissen-Literale (Konjunktion) mit allen 1-Klauseln.

- Beobachtung: in der Auswahl-Schleife besteht in Iteration $k > 1$ kein Grund der (erneuten) Auswahl einer Regel, sofern nicht mindestens ein Literal der Prämisse in Iteration k-1 ***neu*** abgeleitet wurde.

  $\rightarrow$ Auswahl der Regeln, deren Prämisse mind. ein neu abgeleitet. Literal zeigt.

- *Database Indexing:* Fakten-Retrieval in $O(1)$ – in Wissensbasis *Crime-KB* führt z.B. Anfrage *Missile(x)* zu Antwort *Missile(M1).*

- Einsatz von Vorwärtsableitung in *Deduktiven Datenbanken* !

Einfache Rückwärtsableitung mit GMP (1)

Prinzip der Rückwärtsableitung:

von der Anfrage (Ziel) zu den Fakten der Wissensbasis!

Genauer:

Die Funktion FOL-BC-ASK

- Eingabe: Wissensbasis KB und Konjunktion von Teilzielen *goals* als Anfrage.
- Ausgabe: Menge der unifizierenden Substitutionen oder *false*.
- Prozess: iterative Anwendung der Regelklauseln auf die Teilziele, bis alle Teilziele aus KB abgeleitet sind oder keine neuen Ableitungen erzeugbar sind.
- Prinzip der Tiefensuche $\leadsto$ lineare Speicherkomplexität.

Einfache Rückwärtsableitung mit GMP (2)

function FOL-BC-ASK(KB, $goals$, θ) **returns** a set of substitutions

inputs: KB, the knowledge base, a set of first-order definite clauses

$goals$, a list of conjuncts forming a query (θ already applied)

θ, the current substitution, initially the empty substitution

local variables: $answers$, a set of substitutions, initially empty

if $goals$ is empty **then return** θ

Zieltest

$q' \leftarrow$ SUBST(θ,FIRST($goals$))

for each sentence r **in** KB

where STANDARDIZE-APART(r) = $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$

and $\theta' \leftarrow$ UNIFY (q, q') succeeds

$new_goals \leftarrow [p_1, p_2, \ldots p_n|$REST($goals$)$]$

$answers \leftarrow$ FOL-BC-ASK(KB, new_goals,COMPOSE(θ', θ)) $\bigcup$ $answers$

return *answers*

COMPOSE(θ ´,θ) entspricht der Substitution, bei der zunächst θ und dann θ ´ *angewendet wird.*

Beweis mit einfacher Rückwärtsableitung (1)

$\boxed{Criminal(West)}$

American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

American(x) $\wedge$ *Weapon(y)* $\wedge$ *Sells(x, y, z)* $\wedge$ *Hostile(z)* $\Rightarrow$ *Criminal(x),*

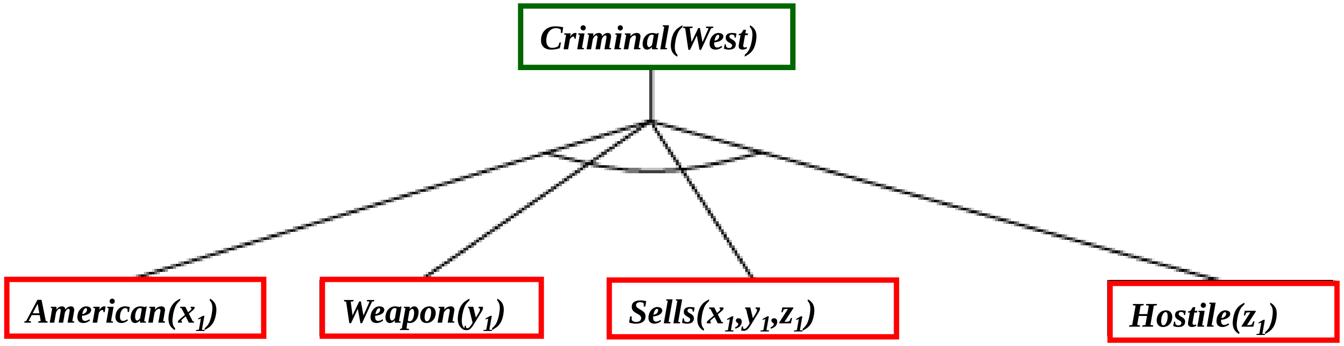
Missile(x) $\wedge$ *Owns(Nono, x)* $\Rightarrow$ *Sells(West,x,Nono),*

Missile(x) $\Rightarrow$ *Weapon(x),*

Enemy(x, America) $\Rightarrow$ *Hostile(x).*

α = *Criminal(West),* θ = {}

Beweis mit einfacher Rückwärtsableitung (2)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America)‘,

$American(x_1) \wedge Weapon(y_1) \wedge Sells(x_1, y_1, z_1) \wedge Hostile(z_1) \Rightarrow Criminal(x_1)$,

$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West,x,Nono)$,

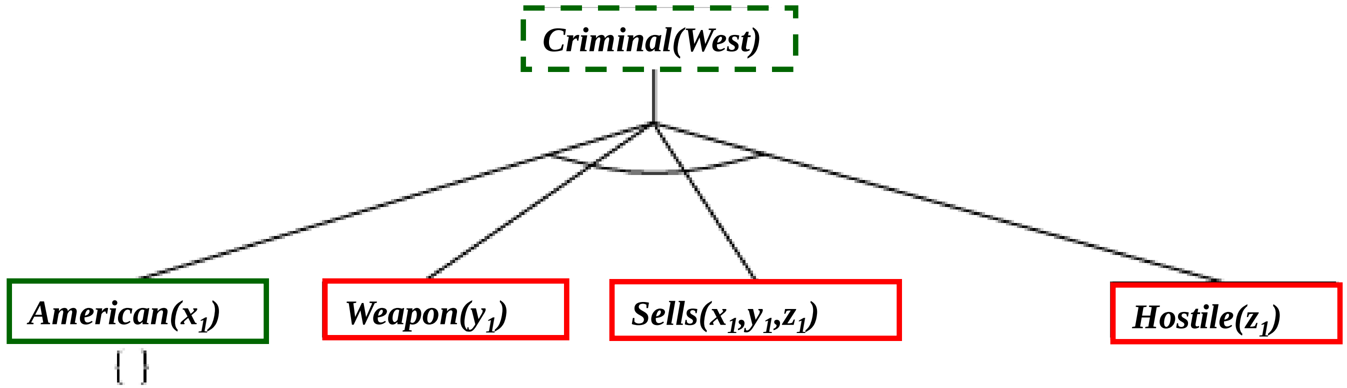
$Missile(x) \Rightarrow Weapon(x)$,

$Enemy(x, America) \Rightarrow Hostile(x)$.

$\alpha = Criminal(West)$,

goals = $\{American(x_1), Weapon(y_1), Sells(x_1, y_1,z_1), Hostile(z_1)\}$, $\theta = \{\}$, $\theta' = \{x_1/West\}$

Beweis mit einfacher Rückwärtsableitung (3)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

American(x_1) $\wedge$ Weapon(y_1) $\wedge$ Sells(x_1, y_1, z_1) $\wedge$ Hostile(z_1) $\Rightarrow$ Criminal(x_1),

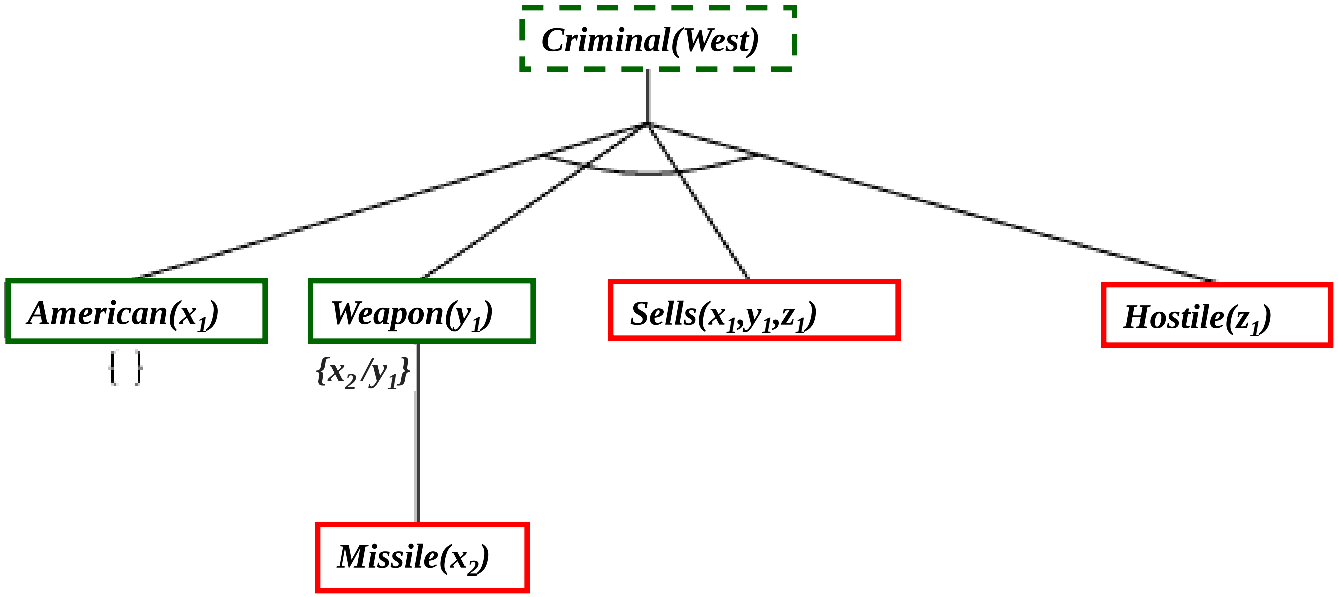
Missile(x) $\wedge$ Owns(Nono, x) $\Rightarrow$ Sells(West,x,Nono),

Missile(x) $\Rightarrow$ Weapon(x),

Enemy(x, America) $\Rightarrow$ Hostile(x).

goals = {Weapon(y_1), Sells(x_1, y_1, z_1), Hostile(z_1)}, θ = {x_1/West}, θ‘ = {}

Beweis mit einfacher Rückwärtsableitung (4)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

$American(x_1) \wedge Weapon(y_1) \wedge Sells(x_1, y_1, z_1) \wedge Hostile(z_1) \Rightarrow Criminal(x_1)$,

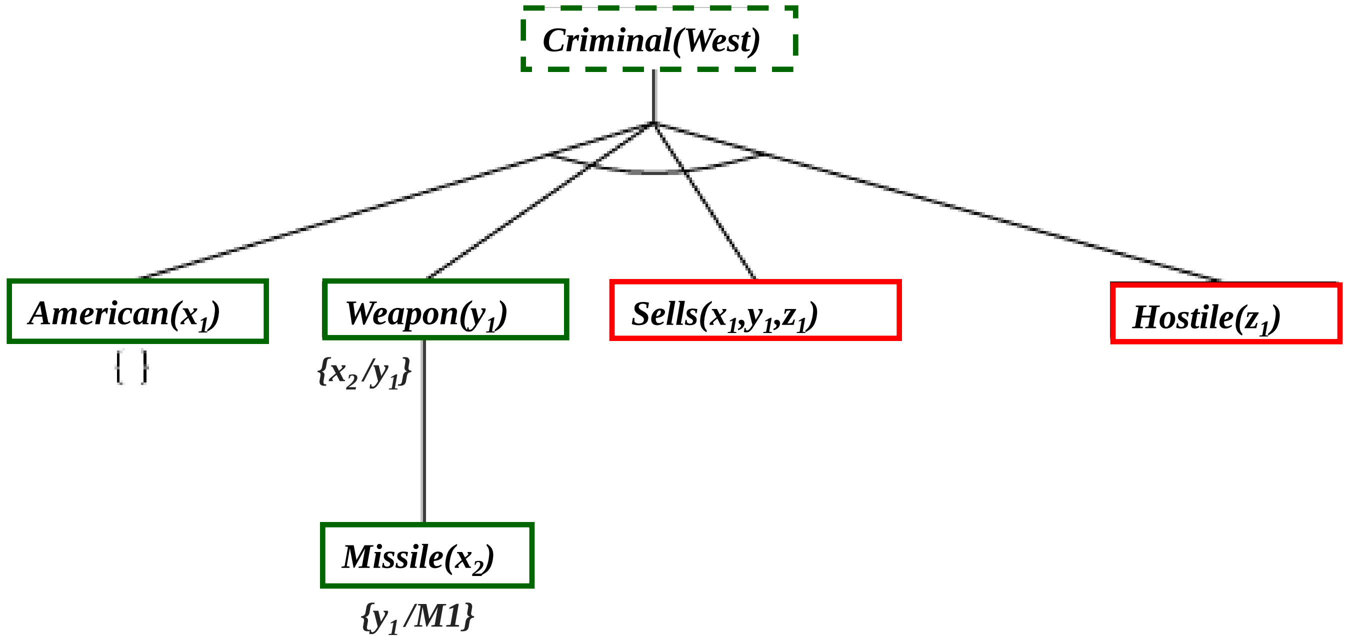
$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West,x,Nono)$,

$Missile(x_2) \Rightarrow Weapon(x_2)$,

$Enemy(x, America) \Rightarrow Hostile(x)$.

goals = $\{Missile(x_2), Sells(x_1, y_1, z_1), Hostile(z_1)\}$, $\theta = \{x_1/West\}$, $\theta' = \{x_2/y_1\}$

Beweis mit einfacher Rückwärtsableitung (5)



$American(West)$, $Missile(M1)$, $Owns(Nono, M1)$, $Enemy(Nono, America)$,

$American(x_1) \wedge Weapon(y_1) \wedge Sells(x_1, y_1, z_1) \wedge Hostile(z_1) \Rightarrow Criminal(x_1)$,

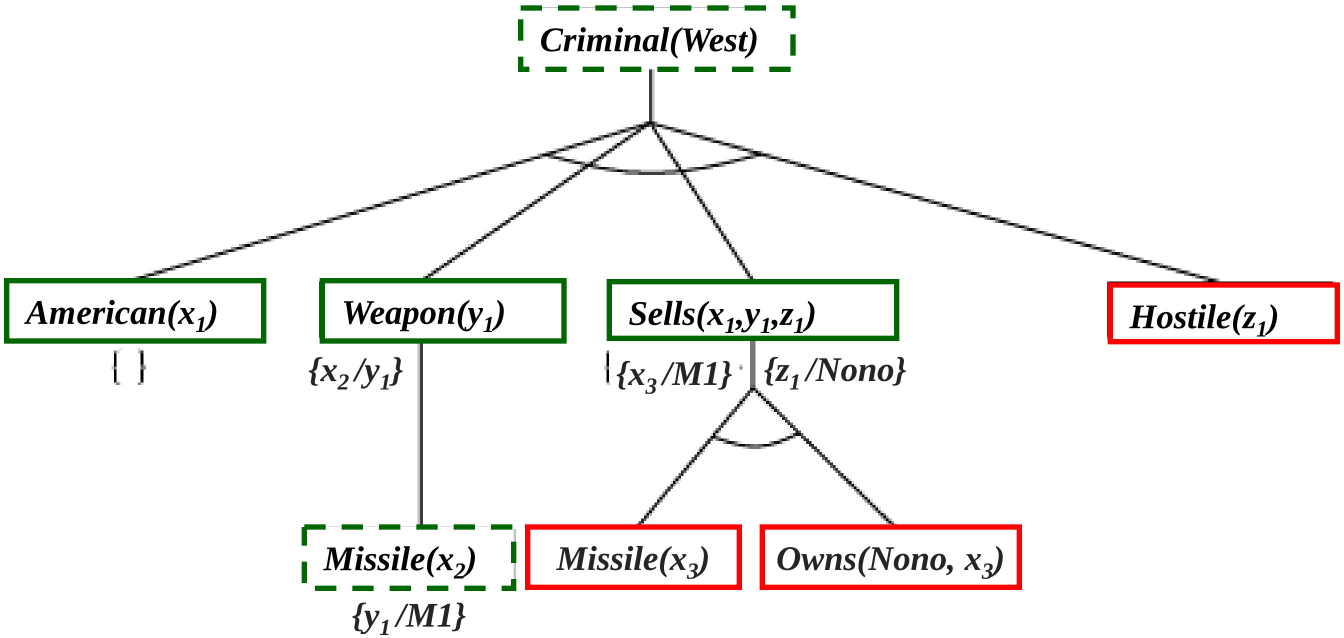
$Missile(x) \wedge Owns(Nono, x) \Rightarrow Sells(West,x,Nono)$,

$Missile(x_2) \Rightarrow Weapon(x_2)$,

$Enemy(x, America) \Rightarrow Hostile(x)$.

$goals = \{Sells(x_1, y_1, z_1), Hostile(z_1)\}$, $\theta = \{x_1/West, x_2/y_1\}$, $\theta' = \{y_1/M1\}$

Beweis mit einfacher Rückwärtsableitung (6)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

$American(x_1) \wedge Weapon(y_1) \wedge Sells(x_1, y_1, z_1) \wedge Hostile(z_1) \Rightarrow Criminal(x_1)$,

$Missile(x_3) \wedge Owns(Nono, x_3) \Rightarrow Sells(West, x_3, Nono)$,

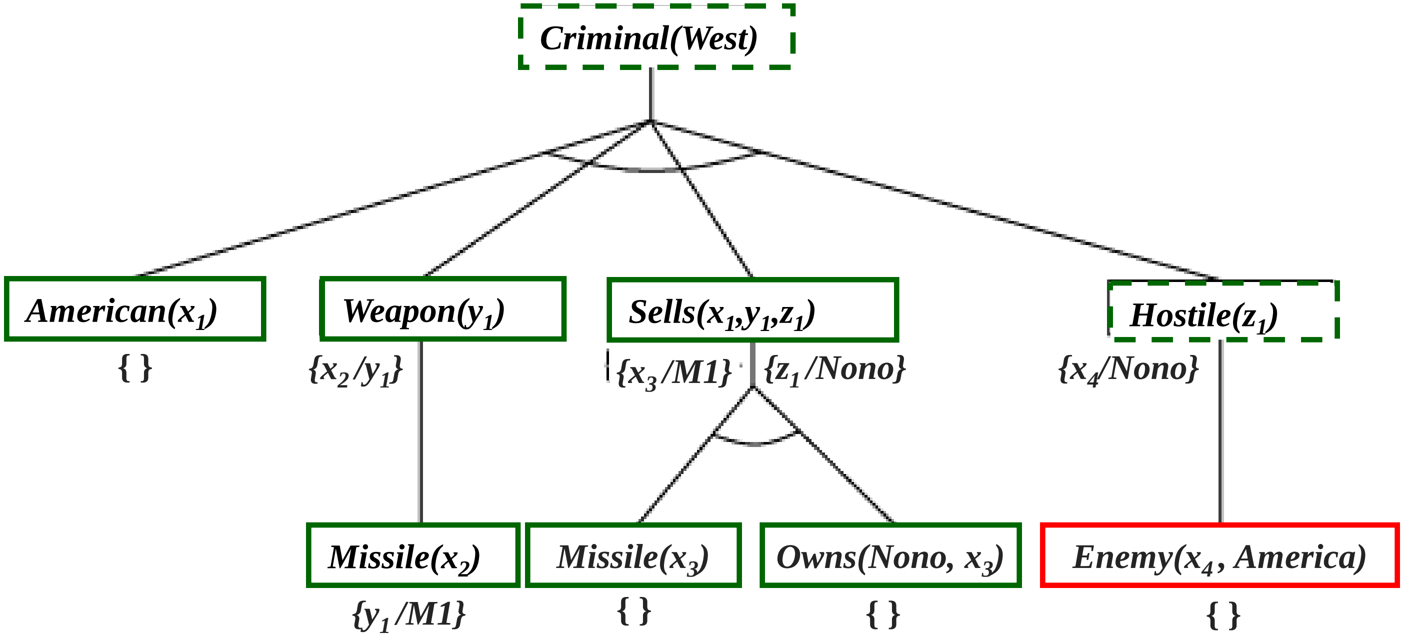
$Missile(x_2) \Rightarrow Weapon(x_2)$,

$Enemy(x, America) \Rightarrow Hostile(x)$.

goals = $\{Missile(x_3), Owns(Nono, x_3), Hostile(z_1)\}$,

$\theta = \{x_1/West, x_2/y_1, y_1/M1\}$, $\theta' = \{x_3/M1, z_1/Nono\}$

Beweis mit einfacher Rückwärtsableitung (7)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

$American(x_1) \wedge Weapon(y_1) \wedge Sells(x_1, y_1, z_1) \wedge Hostile(z_1) \Rightarrow Criminal(x_1)$,

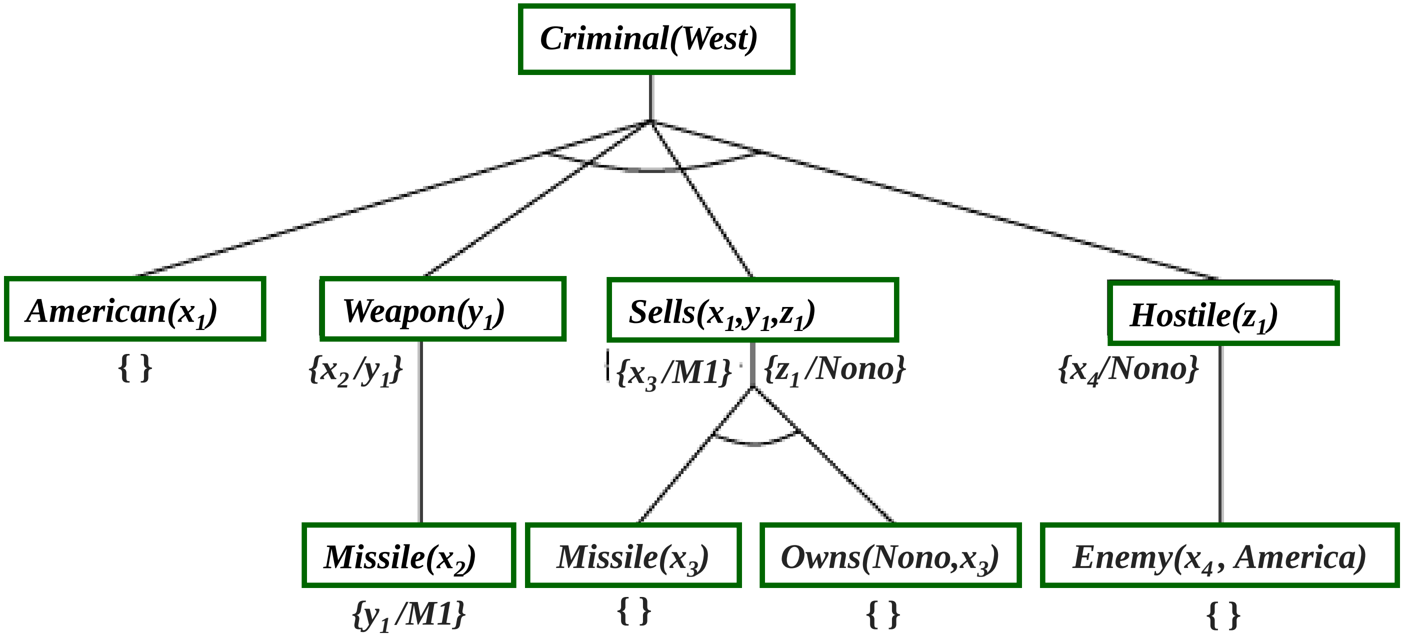
$Missile(x_3) \wedge Owns(Nono, x_3) \Rightarrow Sells(West, x_3, Nono)$,

$Missile(x_2) \Rightarrow Weapon(x_2)$,

$Enemy(x_4, America) \Rightarrow Hostile(x_4)$.

goals = { $Enemy(x_4, America)$ }, θ = {x_1/West, x_2/y_1, y_1/M1, x_3/M1, z_1/Nono}, θ' = {x_4/Nono}

Beweis mit einfacher Rückwärtsableitung (8)



American(West), Missile(M1), Owns(Nono, M1), Enemy(Nono, America),

$American(x_1) \wedge Weapon(y_1) \wedge Sells(x_1, y_1, z_1) \wedge Hostile(z_1) \Rightarrow Criminal(x_1)$,

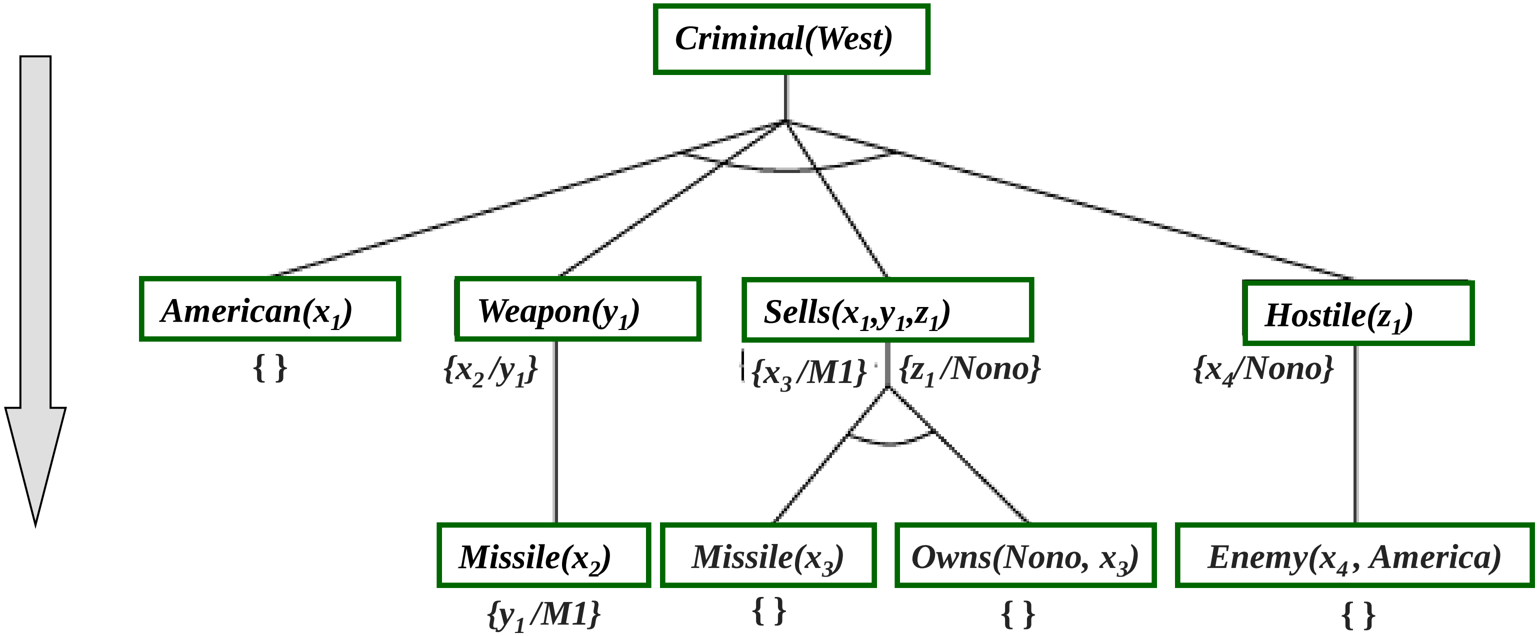
$Missile(x_3) \wedge Owns(Nono, x_3) \Rightarrow Sells(West, x_3, Nono)$,

$Missile(x_2) \Rightarrow Weapon(x_2)$,

$Enemy(x_4, America) \Rightarrow Hostile(x_4)$.

goals = { }, $\theta = \{x_1/West, x_2/y_1, y_1/M1, x_3/M1, z_1/Nono, x_4/Nono\}$, $\theta' = \{\ \}$

Beweis mit einfacher Rückwärtsableitung (8)



Anfrage *Criminal(West)* führt über 1. Regel zu vier neuen Unterzielen

Substitutionen validierter Unterziele werden auf nachfolgende Unterziele angewandt!

Effizienz der einfachen Rückwärtsableitung

- Lineare Speicherkomplexität wegen Tiefensuche.
- Allg. für definite Klauselmengen in PL1 unentscheidbar: Z.B. führen $NatNum(0)$ und $NatNum(n) \Rightarrow NatNum(s(n))$ zu Ableitungen $NatNum(s(0)), NatNum(s(s(0))),$ ...
- Regelauswahl über rekursive Aufrufe ist ebenfalls NP-hart:

  Vergleich aller Prämissen-Literale (Konjunktion) mit allen 1-Klauseln und

  Regelkonklusionen (Pattern-Matching); Backtracking-Punkte.
- Einsatz von Rückwärtsableitung in *Logischer Programmierung*.
- Im Mittel effizienter als Vorwärtsableitung wegen zielorientierter Ableitung.

Prädikatenlogische Resolution

Binäre prädikatenlog. Resolvente zweier Klauseln L = $\{l_1,...l_k\}$ und M = $\{m_1,...,m_n\}$

durch Unify $(l_i\,,\neg m_j) = \theta$:

$$\frac{l_1 \vee \ldots \vee l_k,\quad m_1 \vee \ldots \vee m_n}{(l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)\theta}.$$

Beispiel:

$$\frac{\neg Rich(x) \vee Unhappy(x), Rich(Ken)}{Unhappy(Ken)}$$

mit $\theta = \{x/Ken\}$.

GMP ist nur auf definiten Klauselmengen vollständig. AL-Resolution war wider-legungsvollständig auf allen Klauselmengen.

Wir sagen, dass D aus einer prädikatenlogischen Formelmenge Θ mit Hilfe von (binären) Resolventenbildungen *abgeleitet* werden kann, d.h.

$$\Theta \vdash D$$

wenn es $C_1, C_2, \ldots, C_n = D$ gibt, so dass

$$C_i \in R(\Theta \cup \{C_1, \ldots, C_{i-1}\}), \text{ für } 1 \leq i \leq n.$$

Unvollständigkeit der binären prädikatenlogischen Resolution

Die binäre prädikatenlogische Resolvente

- heißt binär, weil genau zwei Literale beider Elternklauseln resolviert werden;
- ist aber unvollständig für die Prädikatenlogik 1. Stufe! *

Bspl. für Unvollständigkeit:

Klausel 1: $\{+P(x_1),\ \ +P(y_1)\}$,

Klausel 2: $\{–P(x_2),\ \ –P(y_2)\}$.

Offensichtlich ist durch Gleichsetzung aller Variablen (z.B. zu X_1) ein logischer Widerspruch ableitbar:

$$(P(x_1) \vee P(x_1)) \wedge (-P(x_1) \vee -P(x_1))$$

$$\rightarrow P(x_1) \wedge -P(x_1) \text{ (Vereinfachung)}$$

$$\rightarrow \mathit{false} \text{ (Widerspruch).}$$

Durch binäre Resolution sind aber nur neue 2-Klauseln (z.B. $+P(x_1) \vee -P(x_2)$) durch Resolution über $+P(y_1)$ u. $-P(y_2)$) ableitbar, nicht aber die leere Klausel □.

---

* Binäre Resolution ist widerspruchsvollständig für Hornklauselmengen (Klauseln mit höchstens einem positiven Literal) und definite Klauselmengen (Klauseln mit genau einem positiven Literal)!

Vollständigkeit von binärer Resolution und Faktorisierung in PL-1

Wird die binäre prädikatenlogische Resolution um die Faktorisierung ergänzt, so ist die Kombination wieder vollständig für PL-1.

Aus der Aussagenlogik ist bekannt, dass zwei identische Variable in einer Klausel reduzierbar sind auf eine Variable, z.B.: $a \vee b \vee b \vee c \rightarrow a \vee b \vee c$.

Die Faktorisierung ist die Erweiterung dieser Reduktion in die PL-1 so, dass zwei Literale einer Klausel durch Unifikation zu einem Literal verschmolzen werden können, sofern diese Literale dasselbe Vorzeichen haben.

Bemerkung: Alternativ zur Kombination von binärer Resolution und Faktorisierung führt eine Erweiterung der binären Resolution auf *Untermengen* von Literalen auch zur Widerlegungs-vollständigkeit der Resolution für PL-1. Diese Variante ist im Anhang skizziert. In dieser Vorlesung und den Übungsaufgaben werden aber nur binäre Resolution und Faktorisierung eingesetzt.

Beispiel für Faktorisierung mit binärer Resolution

Die Faktorisierung am Beispiel von eben:

Klausel 1: $\{+P(x_1),\ +P(y_1)\}$

Klausel 2: $\{-P(x_2),\ -P(y_2)\}$

Faktor 1: $\{+P(x_1)\}$ durch Faktorisierung $\{y_1/x_1\}$ aus Klausel 1

Faktor 2: $\{-P(x_2)\}$ durch Faktorisierung $\{y_2/x_2\}$ aus Klausel 2

Bin. Resolvente: $\square$ durch bin. Resolut. $\{x_1/x_2\}$ auf Faktoren 1 und 2

Bemerkung: In einem Gesamtbeweis sind also für jeden Resolutionsschritt alle Klauseln und deren Faktoren als Kandidaten für die Elternklauseln zu betrachten.

Korrektheit und Vollständigkeit der prädikatenlogischen Resolution

Satz: Die Anwendung der vollständigen prädikatenlogischen Resolution (Folge von Resolventen- und Faktorenbildungen oder von auf Untermengen erweiterte Resolutionsschritte) auf die $KNF(KB \wedge \neg\alpha)$ ist *korrekt* und *widerspruchsvollständig* für die Prädikatenlogik 1. Stufe.

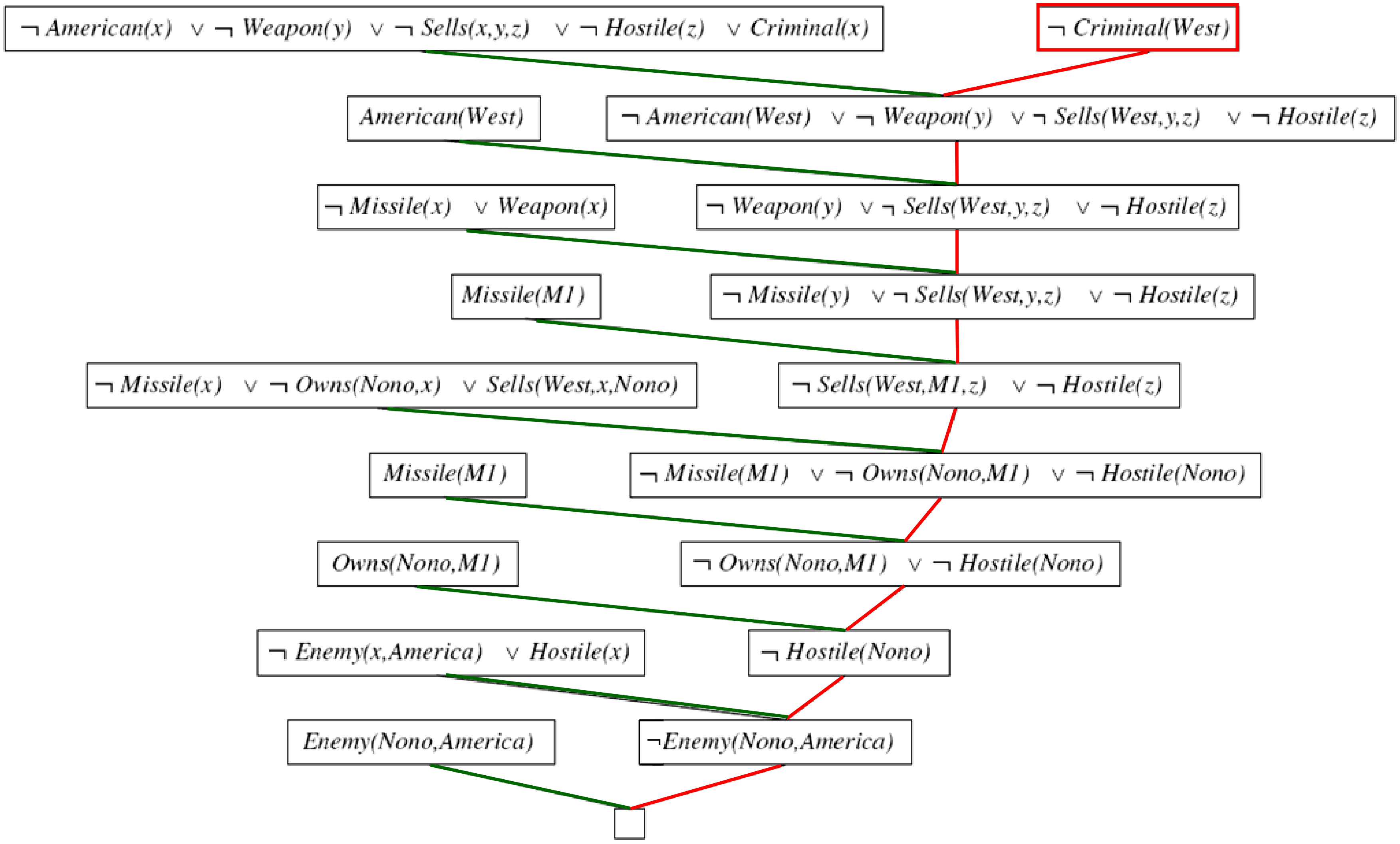
Beweisidee: Korrektheit und Vollständigkeit der AL-Resolution mit Lifting-Lemma.

Lifting Lemma: Seien C_1 und C_2 zwei prädikatenlogische Klauseln ohne gemeinsame Variablen. Seien C_1' und C_2' Grundinstanzen von C_1 bzw. C_2. Wenn C' Grundresolvente von C_1' und C_2' ist, dann existiert eine prädikatenlogische Klausel C mit

1. C ist Resolvente von C_1 und C_2,

2. C' ist Grundinstanz von C.

Beweis mit prädikatenlogischer Resolution



Kontrollstrategie für Resolution in Prolog

Sei Θ Klauselmenge mit Frageklausel C_0.

Lineare Resolution der Klausel C_n:

1. C_{i+1} ist Resolvente von sog. Centerklausel C_i und Seitenklausel B_i $\forall i = 0,...,n$.
2. Jede Seitenklausel B_i ist entweder aus Θ oder frühere Centerklausel C_j mit $j < i$ $\forall i = 0,...,n$.

Lineare Resolution mit definiten Seitenklauseln (LD-Resolution) von C_n: [a]

1. Die Ableitung von C_n ist eine lineare Ableitung.
2. Jede Centerklausel C_i ist eine Klausel, die nur *negative* Literale enthält.
3. Jede Seitenklausel B_i ist eine definite Regel- oder Faktenklausel.

Die Centerklauseln sind aber Hornklauseln bei LD und SLD (s. Folie 26)

Selektive lineare Resolution mit definiten Seitenklauseln (SLD-Resolution) von C_n: [b]

1. Die Ableitung von C_n ist eine LD-Ableitung.
2. Die Seitenklausel B_i wird für die Centerklausel C_i so gewählt (*Selective LD*), dass über das erste Literal der Centerklausel resolviert werden kann.

[a] Z.T. auch als Lineare Resolution mit negativen Centerklauseln, also LN-Resolution, bezeichnet.
[b] Z.T. auch als Selektive linear. Res. mit negat. Centerklauseln, also SLN-Resolution, bezeichnet.

Resolutionsbeweise mit Klauselkopien

Auch bei der Resolutionsbeweisen sind ggf. Klauselkopien nötig:

Klauselmenge:

$P(x) \vee \neg Q(a) \vee \neg Q(b)$

$Q(x)$

Frage: $P(x)$

Standardisierte Klauselmenge:

K_1: $P(x_1) \vee \neg Q(a) \vee \neg Q(b)$

K_2: $Q(x_2)$

K_3: $\neg P(c)$

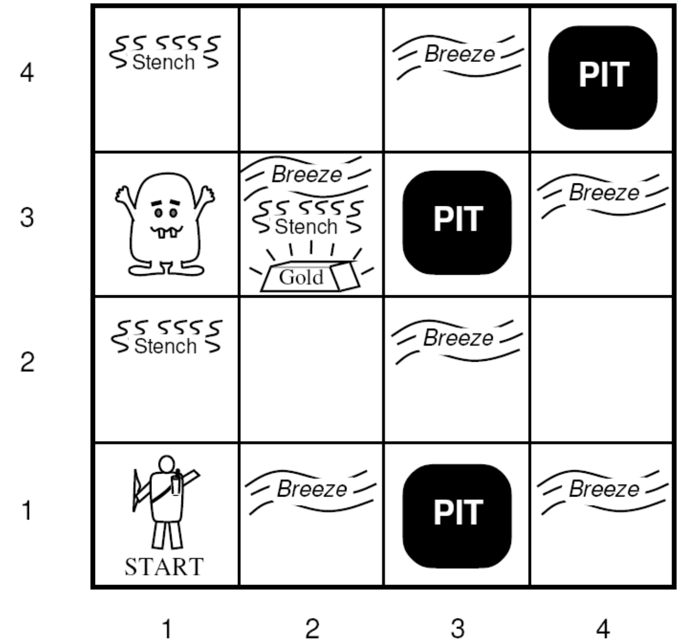
Beweis:

- Res([$K_1 = P(x_1) \vee \neg Q(a) \vee \neg Q(b)$], [$K_3 = \neg P(c)$]) $\rightarrow K_4 = \neg Q(a) \vee \neg Q(b)$
  mit $\{x_1/c\}$
- Res([$K_4 = \neg Q(a) \vee \neg Q(b)$], [$K_2 = Q(x_2)$]) $\rightarrow K_5 = \neg Q(b)$
  mit $\{x_2/a\}$
- Res([$K_5 = \neg Q(b)$], [$K_6 = Q(x_6)$]) $\rightarrow \Box$
  mit $\{x_6/b\}$ durch neue Klauselkopie $K_6 = Q(x_6)$ von $K_2 = Q(x_2)$

Logikbasierte Agenten (1)

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*))
  $t \leftarrow t + 1$
  return *action*



Zentraler Aspekt: Schließen über die Ergebnisse von Aktionen

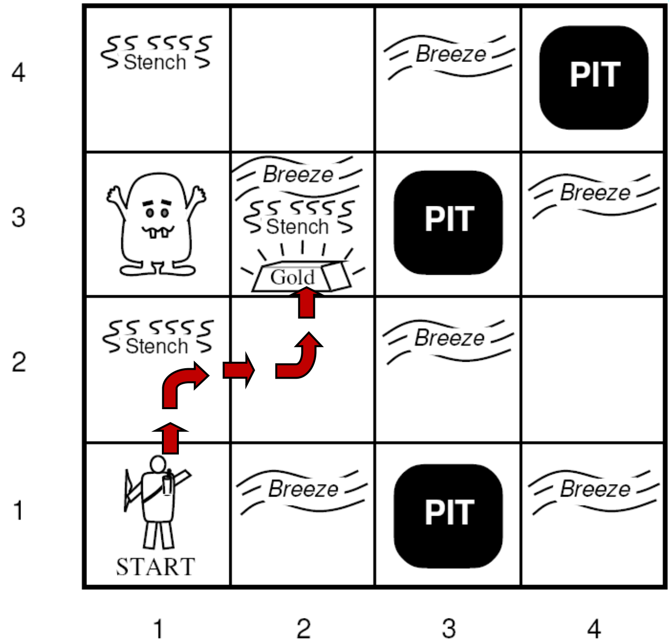
- Nachteil der Aussagenlogik: für jeden Zeitpunkt und Ort eine separate Kopie einer Aktionsbeschreibung

- Jetzt: Verwendung der Logik 1. Stufe in Form des *Situationskalküls*

Logikbasierte Agenten (2)

```
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
```



Anfrage (Make-Action-Query): $\exists a \; Action(a, t)$

Variablenbindungen an a sollen in der WUMPUS-Welt für die obige erste Teilaufgabe folgende Antworten geben:

forward, turn(right), forward, turn(left), forward, grab, ...

Zunächst zum Reflex-Agenten

Reflex-Agenten sind einfach – sie reagieren nur auf Perzepte!

Beispiel für eine Perzept-Aussage (zum Zeitpunkt *t=5*):

*Percept(**s**tench, **b**reeze, **g**litter, no b**u**mp, no s**c**ream, 5)*

1. Schritt: Abstraktion auf wesentliche Eigenschaften für Aktionsauswahl

$$\forall b, g, u, c, t \;\; [\; Percept(stench, b, g, u, c, t) \;\; \Rightarrow \;\; Stench(t) \;]$$

$$\forall s, g, u, c, t \;\; [\; Percept(s, breeze, g, u, c, t) \;\; \Rightarrow \;\; Breeze(t) \;]$$

$$\forall s, b, u, c, t \;\; [\; Percept(s, b, glitter, u, c, t) \;\; \Rightarrow \;\; AtGold(t) \;]$$

. . .

2. Schritt: Aktionsauswahl

$$\forall t \; [\; AtGold(t) \; \Rightarrow Action(grab,t) \;]$$

. . .

Aber: Der Reflex-Agent weiß nicht, wann er aus der Höhle klettern soll, und kann keine Endlosschleifen verhindern, da er weder Perzept noch Gedächtnis für den Besitz des Goldes und die daraus folgende neue Zielsetzung hat.

Modellbasierte Agenten

. . . haben ein internes Modell

- aller wesentlichen Aspekte ihrer Umgebung,
- der Ausführbarkeit und Wirkung von Aktionen,
- von weiteren wesentlichen Gesetzmäßigkeiten der Welt,
- und den eigenen Zielen.

Wichtigster Aspekt: Wie ändert sich die Welt?

→ **Situationskalkül** (McCarthy 63).

Methodischer Ansatz um die Auswirkungen von Aktionen auf Situationen in einer Modellwelt unter Verwendung der Prädikatenlogik zu beschreiben

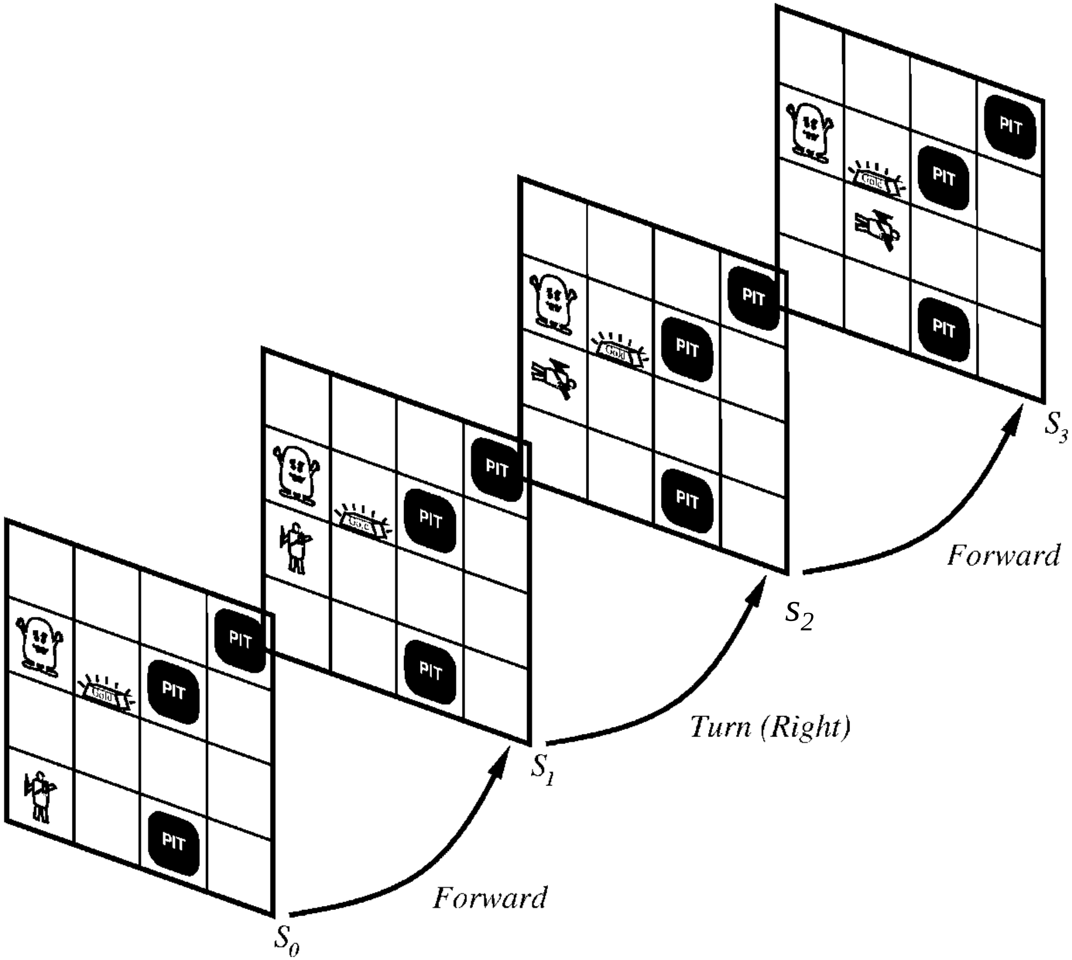
Situationskalkül

- Ein Weg, um mit PL1 **dynamische Welten** zu beschreiben.
- Die dynamische Welt wird als Folge von **Zuständen** (auch Situationen) beschrieben.
- **Zustände** werden durch ***Terme*** repräsentiert.
- Die Welt ist in einem **Zustand** s und kann nur durch **Ausführung** do einer **Aktion** a geändert werden: $do(a,s)$ ist der **Nachfolgezustand** nach Ausführung von Aktion a.
- **Aktionen** werden durch ihre **Vorbedingungen** und ihre **Effekte** beschrieben.
- **Relationen**, *die ihren Wahrheitswert über die Zeit ändern*, heißen **Fluents**. Die Darstellung erfolgt durch ein zusätzliches Argument, das ein Zustandsterm ist. $Holding(x, s)$ bedeutet z.B., dass in der Situation s Objekt x gehalten wird.
- **Konstante** oder **atemporäre Relationen** wie $Even(8)$ benötigen kein Zustands-argument.

Beispiel der WUMPUS-Welt

Sei s_0 die Ausgangssituation mit den Folgesituationen

$s_1 = do(forward, s_0),\quad s_2 = do(turn(right), s_1),\quad s_3 = do(forward, s_2).$



Beschreibung von Aktionen

Vorbedingungen: um etwas *aufzuheben*, muss es *vorhanden* sein und *tragbar* sein:

$$\forall x, s\ [\ Poss(grab(x), s) \;\Leftrightarrow\; Present(x, s) \wedge Portable(x)\ ]$$

Verallgemeinerte Aktionsbeschreibungen durch Quantoren, Objekte, Zustände und Relationen

In der WUMPUS-Welt z. B.:

$$Portable(gold) \quad \text{und} \quad \forall s\ [AtGold(s) \Rightarrow Present(gold, s)]$$

Aktionsbeschreibung durch *positives **Effekt**axiom:*

$$\forall x, s\ [\ Poss(grab(x), s) \;\Rightarrow\; Holding(x, do(grab(x), s))\ ]$$

Aktionsbeschreibung durch *negatives **Effekt**axiom:*

$$\forall x, s\ [\ \neg Holding(x, do(release(x), s))\ ]$$

Beschreibungen positiver und negativer Effekte alleine aber nicht ausreichend!!!

Das Rahmenproblem

Es gelte: $Holding(gold, s_0)$.

Folgt $Holding(gold, do(forward, s_0))$?

Es gelte: $\neg Holding(gold, s_0)$.

Folgt $\neg Holding(gold, do(turn(right), s_0))$?

$\rightarrow$ Man muss auch spezifizieren, welche *Fluents* *nicht* geändert werden!

$\rightarrow$ Das **Rahmenproblem** (*Frame problem*):

Spezifikation der Eigenschaften, die sich bei Aktionen *nicht* ändern.

$\rightarrow$ Es müssen auch **Rahmenaxiome** spezifiziert werden.

Anzahl der Rahmenaxiome

Rahmenaxiom, das spezifiziert, dass alle Aktionen außer $release(x)$ die Relation $Holding(x,s)$ im Nachfolgezustand nicht ändern:

$$\forall a, x, s\ [\ Holding(x, s) \wedge (a \neq release(x)) \Rightarrow Holding(x, do(a, s))\ ]$$

Rahmenaxiom, das spezifiziert, dass alle Aktionen außer $grab(x)$ für greifbare Objekte x die Relation $\neg Holding(x,s)$ im Nachfolgezustand nicht ändern:

$$\forall a, x, s\ [\ \neg Holding(\mathrm{x}, \mathrm{s}) \wedge \{(a \neq grab(x)) \vee \neg Poss(grab(x), s)\} \Rightarrow \neg Holding(x, do(a, s))\ ]$$

Spezifikation der Rahmenaxiome kann sehr aufwendig werden, da bei F Fluents und A Aktionen ggf. $O(|F| \times |A|)$ Axiome spezifiziert werden müssen.

Lösung des Rahmenproblems

Eine elegantere Handhabung des Rahmenproblems ist die vollständige Beschreibung durch **Nachfolgezustands-Axiome**, die die Anteile der Effekt- und Rahmenaxiome kombinieren:

> *wahr* nach Aktion $\Leftrightarrow$ [Aktion hat es *wahr* gemacht $\vee$
> bereits *wahr* und durch Aktion nicht *falsch* geworden].

Beispiel für *Holding*:

$$\forall a, x, s\ [\ Holding(x, do(a, s)) \Leftrightarrow [\ a = grab(x) \wedge Poss(a, s) \vee Holding(x, s) \wedge a \neq release(x)\ ]\ ].$$

Kann auch automatisch aus der Angabe der entspr. Rahmen- und Effektaxiome kompiliert werden („Erklärungsabschluss"). Dabei wird die Annahme gemacht, dass nur die spezifizierten Effekte auftreten können.

Grenzen dieser Version des Situationskalküls

- Keine explizite **Zeit**. Lediglich chronologisch geordnete diskrete Zustände bzw. Situationen. Man kann nicht darüber reden, *wie lange* eine Aktion für ihre Ausführung benötigt.

- **Nur ein Agent**. Im Prinzip können aber auch mehrere Agenten modelliert werden (→ Multiagentensysteme).

- **Keine parallele Ausführung** von Aktionen.

- **Abgeschlossene Welt und Determinismus**: Nur der Agent verändert den Zustand und Aktionen werden immer mit absoluter Sicherheit korrekt ausgeführt.

→ Trotzdem für viele Fälle ausreichend!

- Abschließend daher noch zur Modellierung von Aktionen im Raum, von Exploration und sowie Zielorientierung in der Wumpus-Modellwelt.

Modellierung von Raum und Bewegung (1)

- Der Agent hat eine ***Orientierung***: 0 (nach Osten), 90 (nach Norden), usw:

$$orientation(s) = 0, \quad orientation(s) = 90, \quad \ldots$$

- Mit der Orientierung und der aktuellen Position können wir die ***Folgeposition***, d.h. die Zelle bestimmen, zu der wir als nächstes kommen:

$$\forall x, y\ [\ locationToward([x, y], 0) = [x + 1, y]\ ],$$

$$\cdots$$

Aus der aktuellen Situation heraus:

$$\forall x, y, s\ [\ At([x, y], s) \Rightarrow locationAhead(s) = locationToward([x, y], orientation(s))\ ].$$

- ***Begrenzungen***, z.B. in der Form von Wänden im Wumpus-Szenario:

$$\forall x, y\ [\ Wall([x, y]) \Leftrightarrow (x = 0 \lor x = 5 \lor y = 0 \lor y = 5)\ ].$$

Modellierung von Raum und Bewegung (2)

→ Damit kann man dann die entsprechenden Nachfolgezustands-Axiome für *forward* und *turn* angeben:

$$\forall a, x, y, s \quad [\quad At([x, y], do(a, s)) \Leftrightarrow$$
$$[\ (a = forward \wedge [x, y] = locationAhead(s) \wedge \neg Wall([x, y])) \vee$$
$$(At([x, y], s) \wedge a \neq forward) \ ] \quad] \ *$$

$$\forall a, d, s \quad [\quad orientation(do(a, s)) = d \quad \Leftrightarrow$$
$$[\ (a = turn(right) \wedge d = mod(orientation(s) - 90{,}360)) \vee$$
$$(a = turn(left) \wedge d = mod(orientation(s) + 90{,}360)) \vee$$
$$(orientation(s) = d \wedge \neg(a = turn(right) \vee a = turn(left))) \ ] \quad]$$

* zur Erinnerung: keine Seitwärtsbewegungen möglich, sondern nur drehen und vorwärts gehen

Explorieren

Für die Exploration der Umwelt stehen zwei Arten der Inferenz zur Verfügung:

1. **Diagnostische Regeln**, die aus Perzepten Umgebungseigenschaften ableiten – sich z. B. merken, wo es stinkt und wo ein Luftzug weht *:

$$\forall l, s \quad [\, At(Agent,l,s) \wedge Breeze(s) \Rightarrow Breezy(l) \,],$$

$$\forall l, s \quad [\, At(Agent,l,s) \wedge Stench(s) \Rightarrow Smelly(l) \,].$$

Keine Fluents!

2. **Kausale Regeln**, welche die *Ursachen* der Perzepte „erklären“. Bspl.: auf den WUMPUS und die Fallgruben schließen.

$$\forall l_1, l_2, s \quad [\, At(Wumpus, l_1, s) \wedge Adjacent(l_1, l_2) \Rightarrow Smelly(l_2) \,],$$

$$\forall l_1, l_2, s \quad [\, At(Pit, l_1, s) \wedge Adjacent(l_1, l_2) \Rightarrow Breezy(l_2) \,].$$

---

* Orts- bzw. Lokationsvariable *l* als Abkürzung für [*x,y*].

Aktionsauswahl

Je nach Zustand können wir die möglichen Aktionen, deren Vorbedingungen erfüllt sind, nach einer ***Präferenz*** klassifizieren.

Beispielskala für Klassifikation : *Great*, *Good*, *Medium*, *Risky*.

Beispiel: Wenn wir Gold nehmen können oder mit dem Gold die Höhle verlassen können, ist das „*Great*", usw.

$$\forall a, s \ [\ Poss(grab(gold), s) \Rightarrow Great(grab(gold), s)\ ].$$

Wir können dann mit Hilfe einer Klassifikation der Aktionen auf die Anfrage $\exists a\ Action(a, t)$ auswählen:

$$\begin{aligned} \forall a, s \ [\ Great(a, s) &\Rightarrow Action(a, s)\ ], \\ \forall a, s\ [\ Good(a, s) \wedge \neg \exists b\ Great(b, s) &\Rightarrow Action(a, s)\ ], \\ &\ldots \end{aligned}$$

Man könnte die Aktionsauswahl natürlich auch direkt in den Aktionen kodieren, aber das ist wenig *modular*.

Zielvorgabe $\Rightarrow$ zielorientierter Agent

Sobald das Gold gefunden ist, ist die Strategie radikal zu ändern: das Startfeld ist anzusteuern, um die Wumpuswelt zu verlassen:

$$\forall s\ [\ Holding(gold, s) \Rightarrow GoalLocation([1, 1], s)\ ]$$

Explizite Ziele erlauben die Erarbeitung von Aktionssequenzen, um diese Ziele zu erreichen.

Zusammenfassung (1)

- Der Generalisierte Modus Ponens ist korrekt und vollständig für definite PL1-Klauselmengen.

- Die beiden Ansätze des Forward-Chaining und Backward-Chaining werden in Deduktiven Datenbanken bzw. der Logischen Programmierung eingesetzt.

- Die prädikatenlogischen Resolutionsverfahren sind bei Einsatz vollständiger Suchverfahren korrekte und widerspruchsvollständige Ableitungsverfahren.

Zusammenfassung (2)

- Das Situationskalkül ist ein erster Ansatz um mit PL1 dynamische Welten für einen modellbasierten Agenten zu beschreiben. Das Modell des Agenten beschreibt seine möglichen Aktionen und seine Ziele sowie seine Umgebung und deren Gesetzmäßigkeiten.

  - Die Beschreibung der Aktionen führt zum Rahmenproblem, d.h. der Spezifikation der Eigenschaften, die sich bei Aktionen nicht ändern. Dafür müssen Rahmenaxiome bzw. Nachfolgezustands-Axiome eingeführt werden.

  - Agenten agieren häufig mit einem Raum-Zeit-Bezug. Die Modellierung von Raum und Zeit kann in die Nachfolgezustands-Axiome integriert werden.

  - Für die Exploration können kausale Regeln und/oder diagnostische Regeln eingesetzt werden.

  - Um Entscheidungen treffen zu können, müssen Aktionen bzgl. ihres Nutzens bewertet werden sowie Teilziele formuliert werden.

Anhang: Die erweiterten prädikatenlogischen Resolution

Die Erweiterung der binären prädikatenlog. Resolution auf eine Resolution von *Untermengen* von Literalen führt auch zur Vollständigkeit für PL-1.

Vollständige prädikatenlog. Resolution: Seien K_1 und K_2 zwei nichtleere PL1-Klauseln sowie M_1 und M_2 nichtleere Teilmengen von Literalen aus K_1 bzw. K_2. Sei die Menge N der Atomformeln aus M_1 und M_2 unifizierbar mit allg. Unifikator σ_N, so dass $\sigma_N(M_1)$ und $\sigma_N(M_2)$ einelementig sind und verschiedene Vorzeichen haben. Dann heißt $\sigma_N(K_1/M_1) \cup \sigma_N(K_2/M_2)$ prädikatenlogische Resolvente von K_1 und K_2.

Die vollständige prädikatenlogische Resolution am Beispiel von eben:

K_1:	$\{+P(x_1), \ +P(y_1)\}$		$M_1 = K_1$: $\{+P(x_1), \ +P(y_1)\}$
K_2:	$\{–P(x_2), \ –P(y_2)\}$		$M_2 = K_2$: $\{–P(x_2), \ –P(y_2)\}$
N:	$\{P(x_1), P(y_1), P(x_2), P(y_2)\}$		$\sigma_N = \{x_1/y_2, x_2/y_2, y_1/y_2\}$
$\sigma_N(N)$:	$\{P(y_2)\}$,		$\sigma_N(M_1)$: $+p(y_2)$, $\sigma_N(M_2)$: $-P(y_2)$,

$\sigma_N(K_1 /M_1) \cup \sigma_N(K_2 /M_2)$: $\square$