Zaključivanje u logici prvog reda

Poglavlje 9

Prema slajdovima Stuarta Russella (hvala)!

Pregled/Sažetak

- ♢ Reduciranje zaključivanja prvog reda na propozicijsko zaključivanje
- ♢ Unifikacija
- ♢ Generalizirani Modus Ponens
- ♢ Ulančavanje unaprijed (FC) i unatrag (BC)
- ♢ Logičko programiranje — Prolog
- ♢ Rezolucija

Kratka povijest rasuđivanja

450. pr. kr.	stoici	logika sudova, zaključivanje (?)
322. pr .kr.	Aristotel	"silogizmi" (pravila zaključivanja), kvantifikatori
1565.	Cardano	teorija vjerojatnosti (logika sudova + nesigurnost)
1847.	Boole	logika sudova (ponovno)
1879.	Frege	logika prvog reda
1922.	Wittgenstein	dokaz tablicama istinitosti
1930.	Gödel	$\exists$ potpuni algoritam za FOL
1930.	Herbrand	potpuni algoritam za FOL (redukcija na logiku sudova)
1931.	Gödel	$\neg\exists$ potpuni algoritam za aritmetiku
1960.	Davis/Putnam	"praktični" alg. za prop. logiku
1965.	Robinson	"praktični" alg. za FOL – rezolucija

Ideja za zaključivanje u FOL

Pretvaranje KB iz FOL oblika u logiku sudova, a onda
koristiti zaključivanje u LS (to znamo).
Ova tehnika se zove propozicioniranje (propozicionalizacija).

Prvi problem = rečenice s kvantifikatorima
– treba se "riješiti" vezanih varijabli.

Ideja = supstitucija ili instanciranje,
do razine osnovnih termova — izraza bez varijabli.

Univerzalno instanciranje (UI)

Svako instanciranje univerzalno kvantificirane rečenice je relacija logičke posljedice:

$$\frac{\forall\, v \;\; \alpha}{\textsc{Subst}(\{v/g\}, \alpha)}$$

za bilo koju varijablu v i osnovni term g (nema vrijabli).

Oznaka: za danu rečenicu α i supstituciju θ,

$\textsc{Subst}(\theta, \alpha) = \alpha\theta$ označava rezultat "uvrštavanja" θ u α

Oznaka za supstituciju — u obliku liste supstitucija ili vezivanja,

$\theta = \{v/g\}$

Univerzalno instanciranje (UI)

Iz klasičnog aksioma "pohlepni kraljevi su zli"

$$\forall\, x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$

supstitucijama

$$\{x/John\}, \qquad \{x/Richard\}, \qquad \{x/Father(John)\},$$

dobivamo zaključke

$$King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(Father(John)) \land Greedy(Father(John)) \\ \Rightarrow\; Evil(Father(John))$$
$$\vdots$$

Egzistencijalno instanciranje (EI)

Za bilo koju rečenicu α, varijablu v, i konstantni simbol k koji se **ne pojavljuje nigdje drugdje u bazi znanja**:

$$\frac{\exists\, v \quad \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}$$

(Zamjena varijable novim konstantnim simbolom za "neki" objekt.)

Na primjer, iz $\exists\, x \quad Crown(x) \wedge OnHead(x, John)$ zaključujemo

$$Crown(C_1) \wedge OnHead(C_1, John)$$

uz uvjet da je C_1 novi konstantni simbol — tzv. Skolemova konstanta

Još jedan primjer (analiza): iz $\exists\, x \quad d(x^y)/dy = x^y$ dobivamo

$$d(e^y)/dy = e^y$$

uz uvjet da je e novi konstantni simbol.

Instanciranje (UI, EI) — komentari

UI se može primijeniti više puta,
da bi se **dodale** nove rečenice.

Nova KB je logički ekvivalentna staroj.

EI se može primijeniti samo jednom,
da **zamijeni** egzistencijalnu rečenicu (možemo ju baciti iz KB).

Nova KB **nije** logički ekvivalentna staroj,
ali je ispunjiva ako i samo ako je stara KB ispunjiva.

Dakle, nova KB je "posljedično" ekvivalentna staroj
(u smislu zaključivanja).

Redukcija na propozicijsko zaključivanje

Neka KB sadrži samo sljedeće rečenice:

$$\begin{array}{l} \forall\, x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x) \\ King(John) \\ Greedy(John) \\ Brother(Richard, John) \end{array}$$

Instanciranje prve (univerzalne) rečenice na sve moguće načine, a to su

$$\{x/John\}, \qquad \{x/Richard\},$$

daje

$$\begin{array}{l} King(John) \land Greedy(John) \;\Rightarrow\; Evil(John) \\ King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard) \end{array}$$

Nakon toga, prvu rečenicu (s univerzalnim kvantifikatorom) možemo baciti van iz KB.

Redukcija na propozicijsko zaključivanje

Nova KB sadrži sljedeće rečenice:

$$King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

KB je propozicionirana: osnovne atomske rečenice, poput

$King(John),\; Greedy(John),\; Evil(John),\; King(Richard),$ itd.

možemo gledati kao propozicijske simbole.

Zaključak $Evil(John)$ izlazi direktno,
algoritmima za (potpuno) zaključivanje u LS.

Redukcija — problem

Tvrdnja: osnovna rečenica je logička posljedica na novoj KB ako i samo ako je logička posljedica na originalnoj KB.

Tvrdnja: svaka FOL KB može se propozicionirati tako da očuva relaciju logičke posljedice.

Ideja za (potpuni) algoritam:
— propozicionirati KB i upit, primijeniti rezoluciju, vratiti rezultat.

. . . bilo bi prejednostavno, da je baš tako!

Problem: nastaje s funkcijskim simbolima,
skup mogućih supstitucija osnovnim termom je beskonačan
(rekurzivno uvrštavanje)

Primjer: $Father(Father(Father(John)))$, . . .

Redukcija — poluspas

Srećom, postoji i sljedeći poznati rezultat.

Teorem: Herbrand (1930.).
Ako je rečenica α logička posljedica polazne FOL KB, onda je ona logička posljedica **konačnog** podskupa propozicijske KB.

Ideja: Za $n = 0$ do ∞ radi
kreiraj propozicijsku KB instanciranjem termova do dubine n
pogledaj je li α logička posljedica za tu KB.

Problem: ovo radi ako α je logička posljedica,
međutim, ne mora stati (beskonačna petlja) ako α to nije.

Teorem: Turing (1936.), Church (1936.)
Relacija logičke posljedice u FOL je poluodlučiva.

Problemi s propozicionalizacijom

Na primjer, iz

$$\begin{aligned}
&\forall\, x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)\\
&King(John)\\
&\forall\, y \;\; Greedy(y)\\
&Brother(Richard, John)
\end{aligned}$$

očito je $Evil(John)$ — jedino $John$ je kralj.

Stanje do ranih 1960. godina: Propozicionalizacija producira mnogo činjenica, kao što je $Greedy(Richard)$, koje su nevažne za zaključak ($Richard$ nije kralj).

S p k-narnih predikata i n konstanti, postoji $p \cdot n^k$ instanci

S funkcijskim simbolima situacija postaje još mnogo lošija!

Traži se "prava" supstitucija — kako ju prepoznati?

Unifikacija

Zaključak možemo odmah dobiti ako nađemo supstituciju θ takvu da
$King(x)$ i $Greedy(x)$ (iz implikacije) odgovaraju
$King(John)$ i $Greedy(y)$ u rečenicama iz ostatka baze.

$\theta = \{x/John, y/John\}$ radi — dobivamo iste rečenice = unifikacija

Oznaka za unifikaciju

$$\textsc{Unify}(p,q) = \theta \ \text{ ako } \ p\theta = q\theta, \ \text{ tj. } \ \textsc{Subst}(\theta,p) = \textsc{Subst}(\theta,p)$$

Unifikacija — primjeri

$\textsc{Unify}(p, q) = \theta$ ako $p\theta = q\theta$, tj. $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, p)$

Neka je $p = Knows(John, x)$.

Primjer unifikacije p s raznim rečenicama q:

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unifikacija — primjeri

$\textsc{Unify}(p, q) = \theta$ ako $p\theta = q\theta$, tj. $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, p)$

Neka je $p = Knows(John, x)$.

Primjer unifikacije p s raznim rečenicama q:

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unifikacija — primjeri

$\textsc{Unify}(p,q) = \theta$ ako $p\theta = q\theta$, tj. $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, p)$

Neka je $p = Knows(John, x)$.

Primjer unifikacije p s raznim rečenicama q:

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	
$Knows(John, x)$	$Knows(x, OJ)$	

Unifikacija — primjeri

$\text{UNIFY}(p, q) = \theta$ ako $p\theta = q\theta$, tj. $\text{SUBST}(\theta, p) = \text{SUBST}(\theta, p)$

Neka je $p = Knows(John, x)$.

Primjer unifikacije p s raznim rečenicama q:

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	

Unifikacija — primjeri

$\textsc{Unify}(p, q) = \theta$ ako $p\theta = q\theta$, tj. $\textsc{Subst}(\theta, p) = \textsc{Subst}(\theta, p)$

Neka je $p = Knows(John, x)$.

Primjer unifikacije p s raznim rečenicama q:

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$

Problem u zadnjoj unifikaciji = ista varijabla u obje rečenice

Rješenje: Odvojena standardizacija (engl. standardizing apart)

= reimenovanje varijabli da se izbjegne "preklapanje"

Na primjer, $Knows(z_{17}, OJ)$, gdje je z_{17} nova varijabla.

Generalizirani Modus Ponens (GMP)

Proces zaključivanja opisan je jednim pravilom = GMP

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

gdje je $p_i{}'\theta = p_i\theta$ za sve i.

U ranijem primjeru za $Evil(John)$, imamo

$p_1{}'$	je	$King(John)$	p_1	je	$King(x)$
$p_2{}'$	je	$Greedy(y)$	p_2	je	$Greedy(x)$
θ	je	$\{x/John, y/John\}$	q	je	$Evil(x)$
$q\theta$	je	$Evil(John)$			

GMP se koristi za KB koje sadrže definitne klauzule (**točno** jedan pozitivan literal)
Pretpostavlja se da su sve varijable univerzalno kvantificirane

Adekvatnost GMP-a — skica dokaza

Treba pokazati da je

$$p_1', \ldots, p_n', \ (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\theta$$

uz uvjet da je $p_i'\theta = p_i\theta$ za sve i.

Lema: Za svaku definitnu klauzulu p, imamo $p \models p\theta$, korištenjem UI

1. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \Rightarrow q\theta)$
2. $p_1', \ldots, p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\theta \wedge \ldots \wedge p_n'\theta$
3. Iz 1 i 2, $q\theta$ slijedi korištenjem običnog Modus Ponensa

Primjer baze znanja

Zakon kaže da je kriminal ako Amerikanac prodaje oružje neprijateljskim zemljama. Zemlja Nono, neprijatelj Amerike, ima nešto projektila, a sve projektile prodao im je pukovnik West, koji je Amerikanac.

Dokažite da je pukovnik West kriminalac.

Primjer baze znanja (nastavak)

. . . kriminal je ako Amerikanac prodaje oružje neprijateljskim zemljama:

Primjer baze znanja (nastavak)

. . . kriminal je ako Amerikanac prodaje oružje neprijateljskim zemljama:

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

Nono . . . ima nešto projektila

Primjer baze znanja (nastavak)

. . . kriminal je ako Amerikanac prodaje oružje neprijateljskim zemljama:

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

Nono . . . ima nešto projektila, tj., $\exists\, x\ Owns(Nono, x) \land Missile(x)$:

$Owns(Nono, M_1)$ i $Missile(M_1)$ (Skolemova konstanta M_1)

. . . sve projektile prodao im je pukovnik West

Primjer baze znanja (nastavak)

... kriminal je ako Amerikanac prodaje oružje neprijateljskim zemljama:

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

Nono ... ima nešto projektila, tj., $\exists\, x \;\; Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ i $Missile(M_1)$

... sve projektile prodao im je pukovnik West

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

Projektili su oružje:

Primjer baze znanja (nastavak)

... kriminal je ako Amerikanac prodaje oružje neprijateljskim zemljama:

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

Nono ... ima nešto projektila, tj., $\exists\, x\;\, Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ i $Missile(M_1)$

... sve projektile prodao im je pukovnik West

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

Projektili su oružje:

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

Neprijatelj Amerike broji se kao "neprijatelj":

Primjer baze znanja (nastavak)

... kriminal je ako Amerikanac prodaje oružje neprijateljskim zemljama:

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

Nono ... ima nešto projektila, tj., $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$:

$Owns(Nono, M_1)$ i $Missile(M_1)$

... sve projektile prodao im je pukovnik West

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

Projektili su oružje:

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

Neprijatelj Amerike broji se kao "neprijatelj":

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

West, koji je Amerikanac ...

$American(West)$

Zemlja Nono, neprijatelj Amerike ...

$Enemy(Nono, America)$

Algoritam ulančavanja unaprijed

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

```
repeat until new is empty
    new ← { }
    for each sentence r in KB do
        ( p1 ∧ . . . ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
        for each θ such that (p1 ∧ . . . ∧ pn)θ = (p'1 ∧ . . . ∧ p'n)θ
                        for some p'1, . . . , p'n in KB
            q' ← SUBST(θ, q)
            if q' is not a renaming of a sentence already in KB or new then do
                add q' to new
                φ ← UNIFY(q', α)
                if φ is not fail then return φ
    add new to KB
return false
```

α je upit.

Dokaz ulančavanjem unaprijed

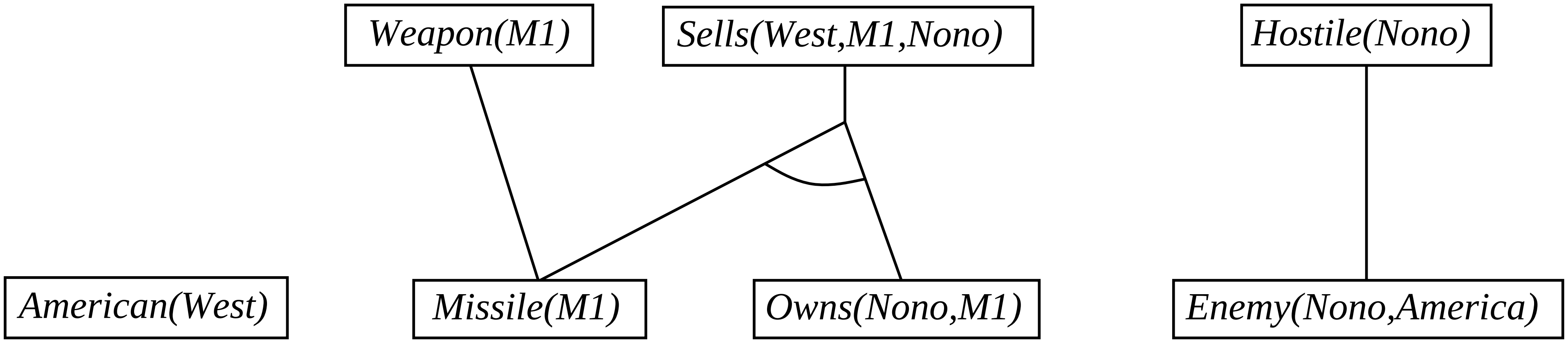
American(West) *Missile(M1)* *Owns(Nono,M1)* *Enemy(Nono,America)*

Krećemo od poznatih činjenica.

Iz njih pokušavamo (supstitucijama) zadovoljiti implikacije.

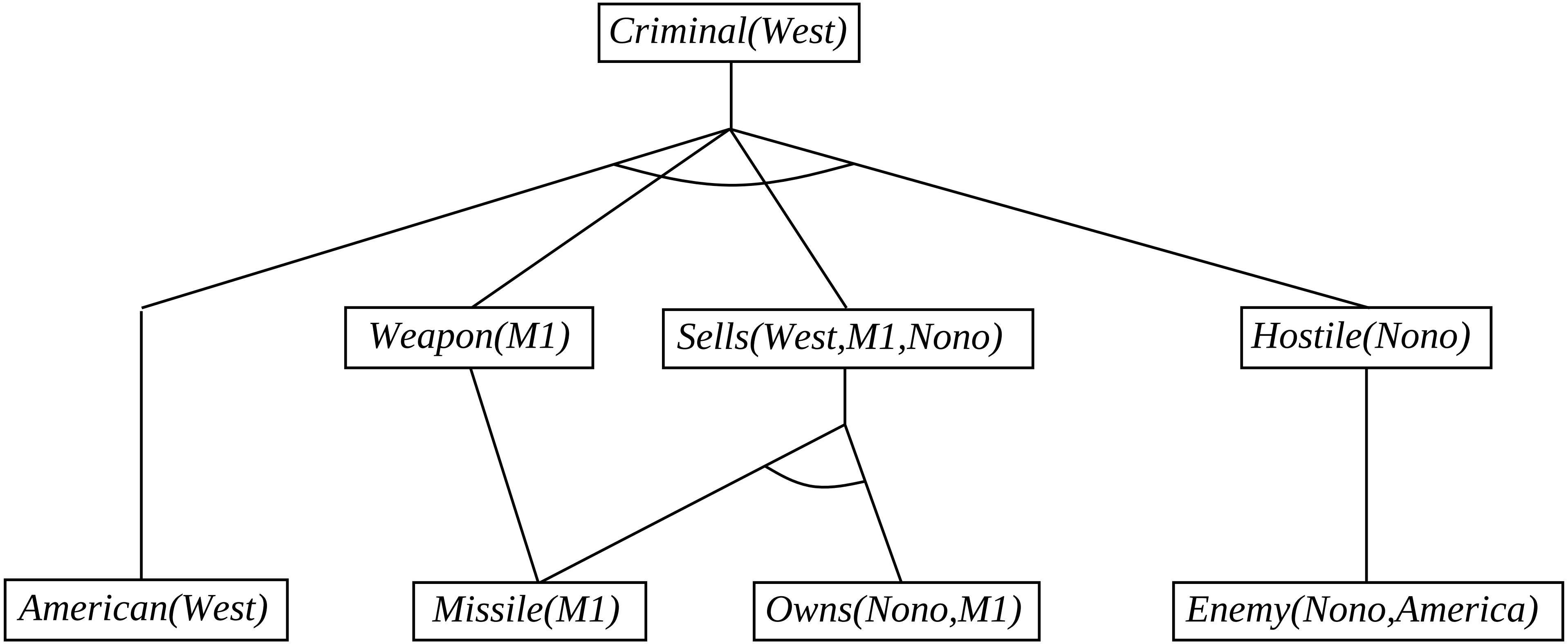
Ako uspijemo, dodajemo nove činenice u KB.

Dokaz ulančavanjem unaprijed — 1. iteracija



1. $Missile(x) \Rightarrow Weapon(x)$ ide s $\{x/M_1\}$
2. $\forall\, x \;\; Missile(x) \wedge Owns(Nono, x) \;\Rightarrow\; Sells(West, x, Nono)$ ide s $\{x/M_1\}$
3. $Enemy(x, America) \;\Rightarrow\; Hostile(x)$ ide s $\{x/Nono\}$

Dokaz ulančavanjem unaprijed — 2. iteracija



1. Polazna implikacija $\ldots \Rightarrow Criminal(x)$ ide s $\{x/West,\ y/M_1,\ z/Nono\}$

Dobivamo zaključak!

Svojstva ulančavanja unaprijed

Ispravnost i potpunost za definitne klauzule prvoga reda
(dokaz sličan propozicijskom dokazu)

Posebno, tzv. Datalog baze znanja
= definitne klauzule prvog reda + **bez funkcija**
(na primjer, KB kriminala)
Za Datalog baze, FC završava u polinomno mnogo iteracija
(baza ima najviše $p \cdot n^k$ literala)

Općenito, ne mora završiti, ako α nije logička posljedica

Ovo je neizbježno:
relacija logičke posljedice s definitnim klauzulama je poluodlučiva

Efikasnost ulančavanja unaprijed

Jednostavno zapažanje:

nema potrebe zadovoljavati pravilo u k-toj iteraciji
ako pretpostavka nije dodana u prethodnoj iteraciji $k-1$

$\Rightarrow$ zadovoljavaj svako pravilo kojemu pretpostavke sadrže neki novododani literal.

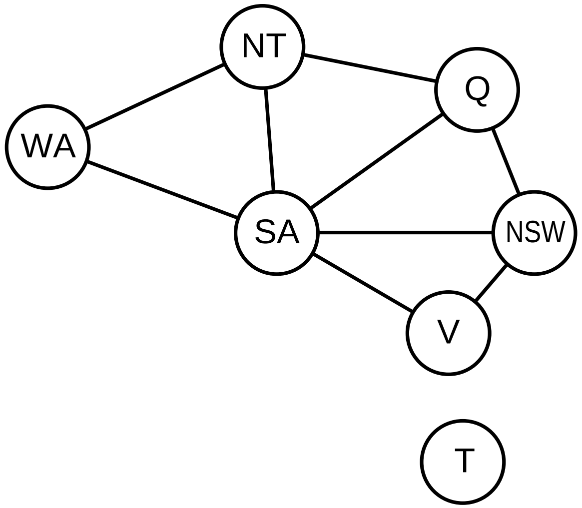
Sparivanje (za zadovoljavanje premisa) samo za sebe može biti skupo

Indeksiranje baze podataka dozvoljava vađenje poznatih činjenica u $O(1)$ vremena — na primjer, svaki $Missile(x)$ vadi $Missile(M_1)$

Problem "sparivanja" konjunktivno vezanih pretpostavki s poznatim činjenicama je NP-težak

Ulančavanje unaprijed je u širokoj upotrebi u deduktivnim bazama podataka

Primjer teškog sparivanja



$$\begin{aligned}&Diff(wa, nt) \wedge Diff(wa, sa) \wedge \\ &\quad Diff(nt, q) Diff(nt, sa) \wedge \\ &\quad Diff(q, nsw) \wedge Diff(q, sa) \wedge \\ &\quad Diff(nsw, v) \wedge Diff(nsw, sa) \wedge \\ &\quad Diff(v, sa) \;\Rightarrow\; Colorable()\end{aligned}$$

$$Diff(Red, Blue) \quad Diff(Red, Green)$$
$$Diff(Green, Red) \quad Diff(Green, Blue)$$
$$Diff(Blue, Red) \quad Diff(Blue, Green)$$

$Colorable()$ je izvedeno ako i samo ako CSP ima rješenje

CSP-i uključuju 3SAT kao specijalni slučaj, pa je sparivanje NP-teško

Algoritam ulančavanja unatrag

```
function FOL-BC-Ask(KB, goals, θ) returns a set of substitutions
   inputs: KB, a knowledge base
           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}
   q' ← Subst(θ, First(goals))
   for each sentence r in KB
           where Standardize-Apart(r) = ( p1 ∧ ... ∧ pn ⇒ q)
           and θ' ← Unify(q, q') succeeds
       new_goals ← [ p1, ..., pn|Rest(goals)]
       answers ← FOL-BC-Ask(KB, new_goals, Compose(θ', θ)) ∪ answers
   return answers
```

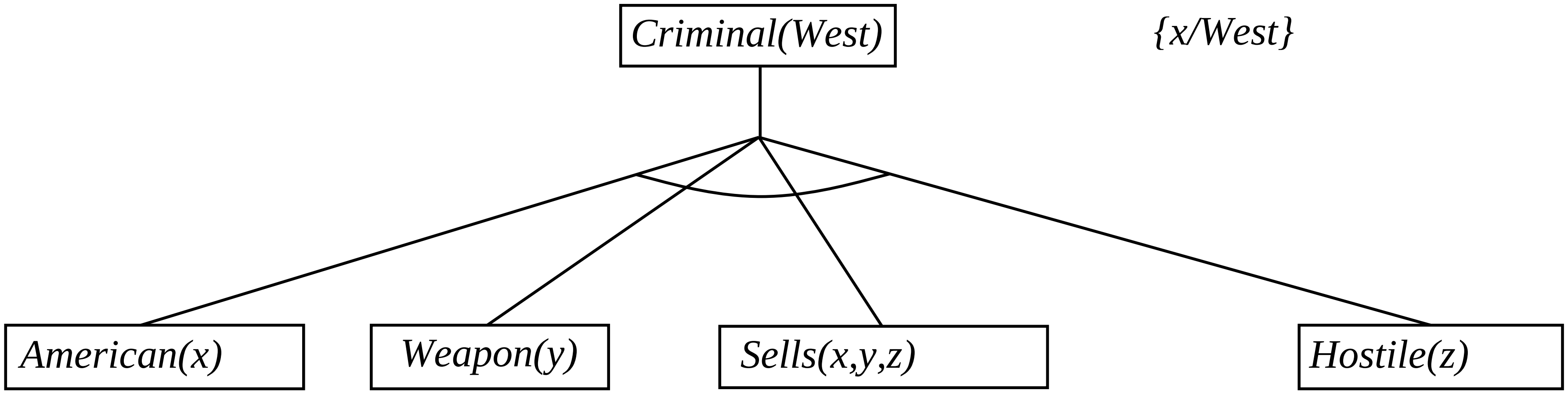
Primjer ulančavanja unatrag

$\boxed{Criminal(West)}$

Krećemo od zadanog upita (dokaži ili opovrgni).

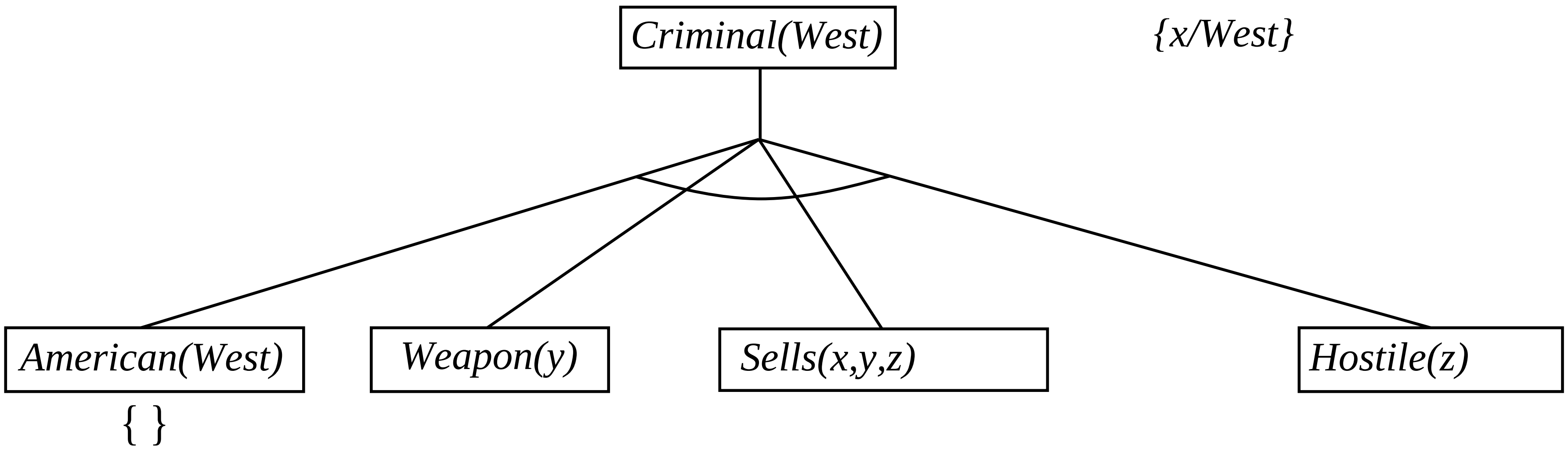
Tražimo klauzulu (ili više njih) iz u kojih bi upit mogao slijediti uz odgovarajuću supstituciju.

Primjer ulančavanja unatrag



Upit može slijediti samo iz polazne implikacije $\ldots \Rightarrow Criminal(x)$
uz $\{x/West\}$

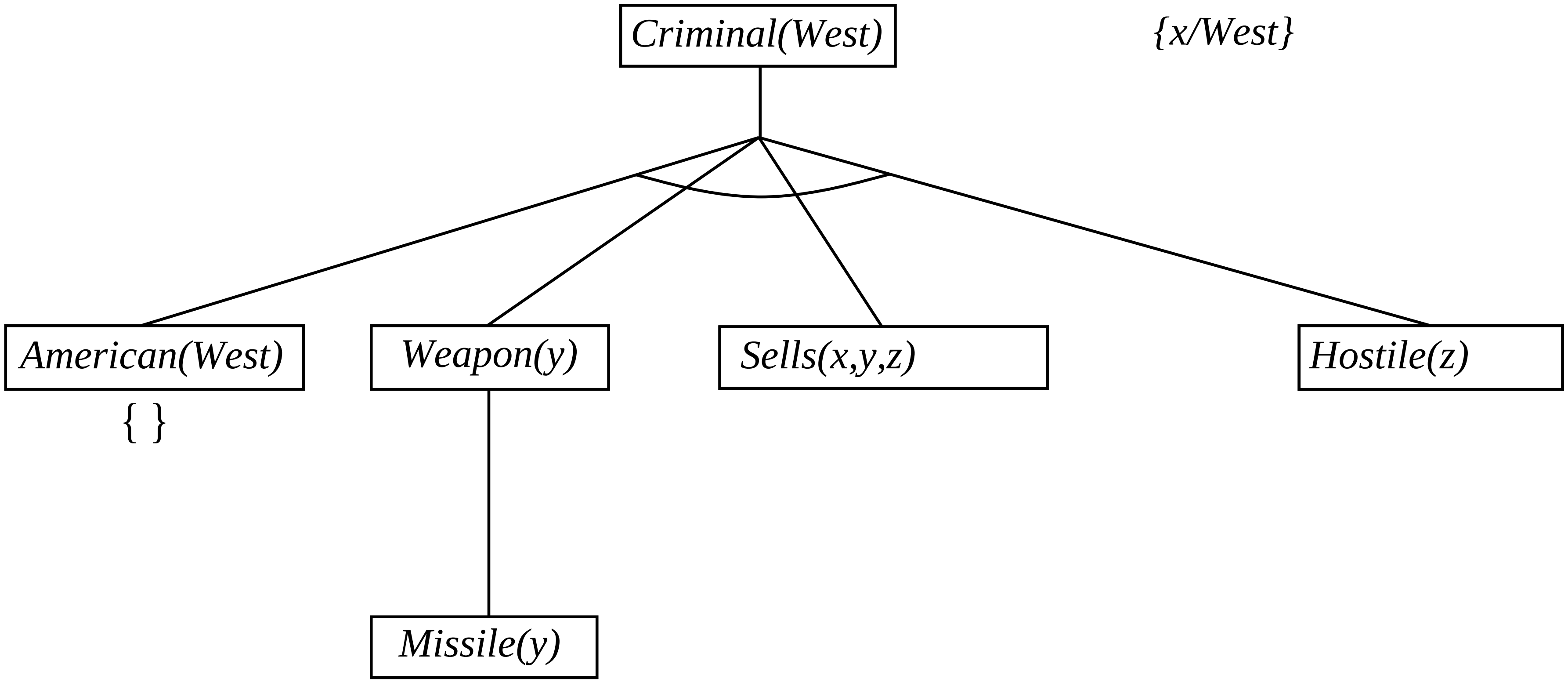
Primjer ulančavanja unatrag



Uvrštavanje u $American(x)$ daje poznatu činjenicu

$\Rightarrow$ taj dio je dokazan (prva $=$ najljevija grana)

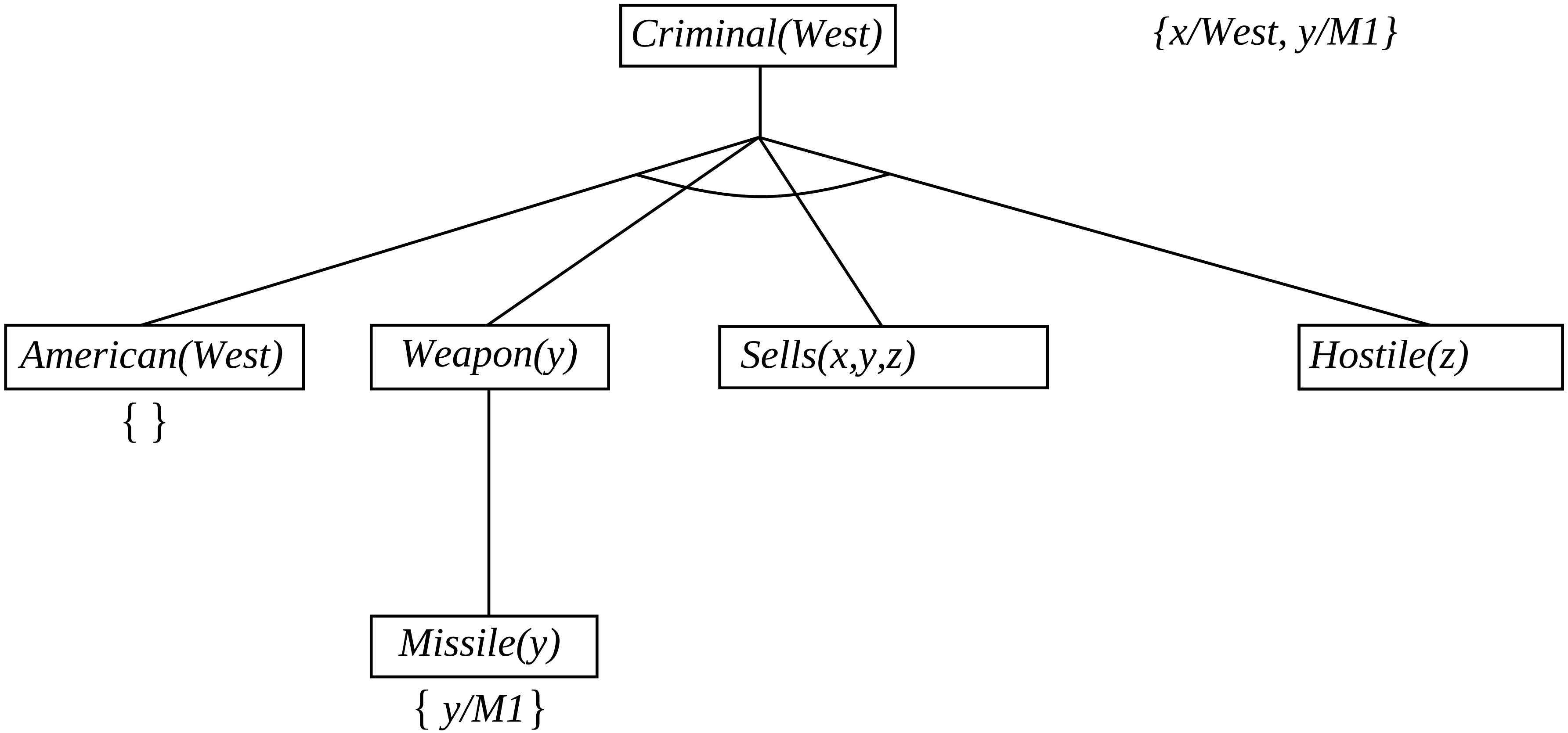
Primjer ulančavanja unatrag



$Weapon(y)$ slijedi iz $Missile(x) \Rightarrow Weapon(x)$

stavimo y umjesto x

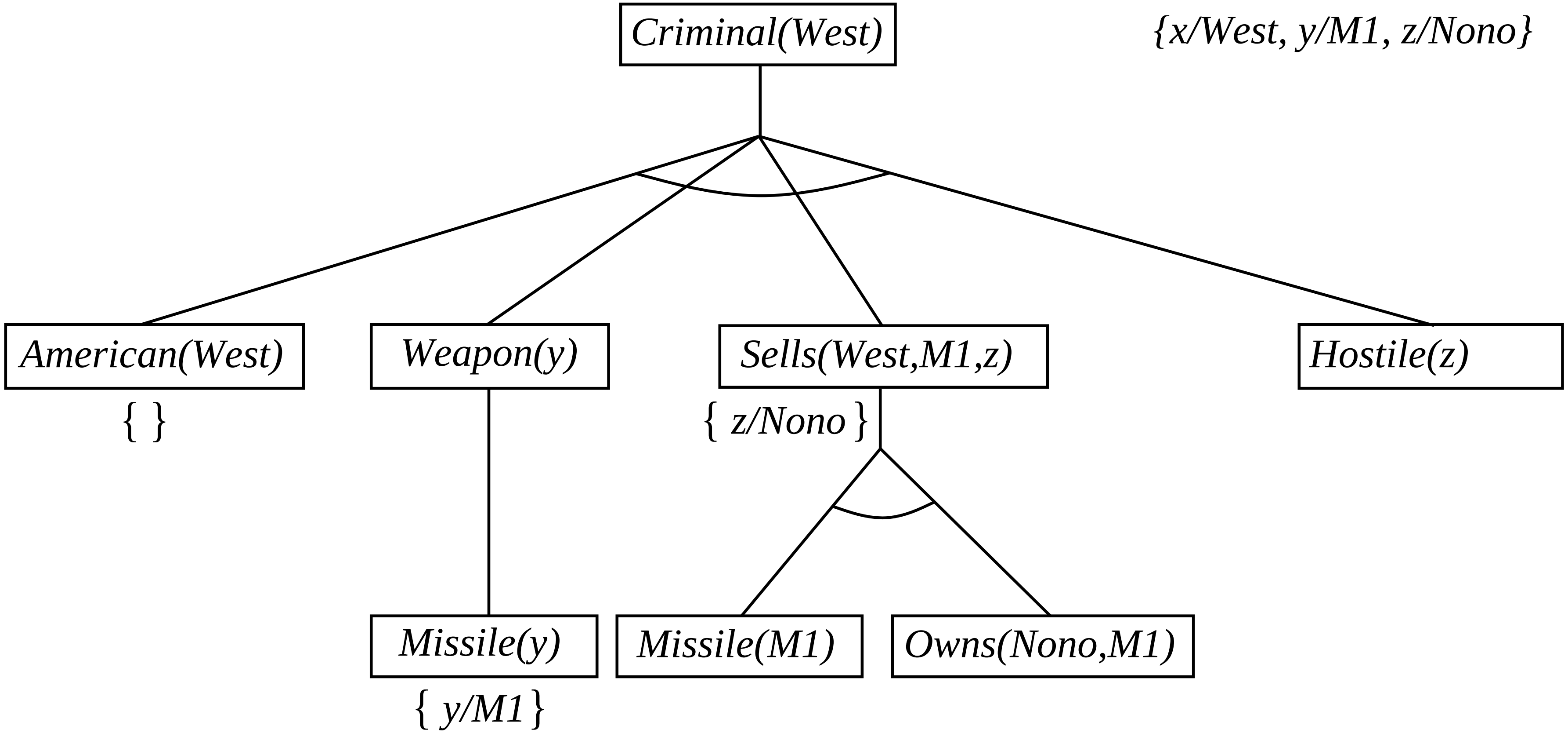
Primjer ulančavanja unatrag



Poznata činjenica u bazi daje supstituciju

$\{y/M_1\}$ $\Rightarrow$ i druga grana je dokazana

Primjer ulančavanja unatrag

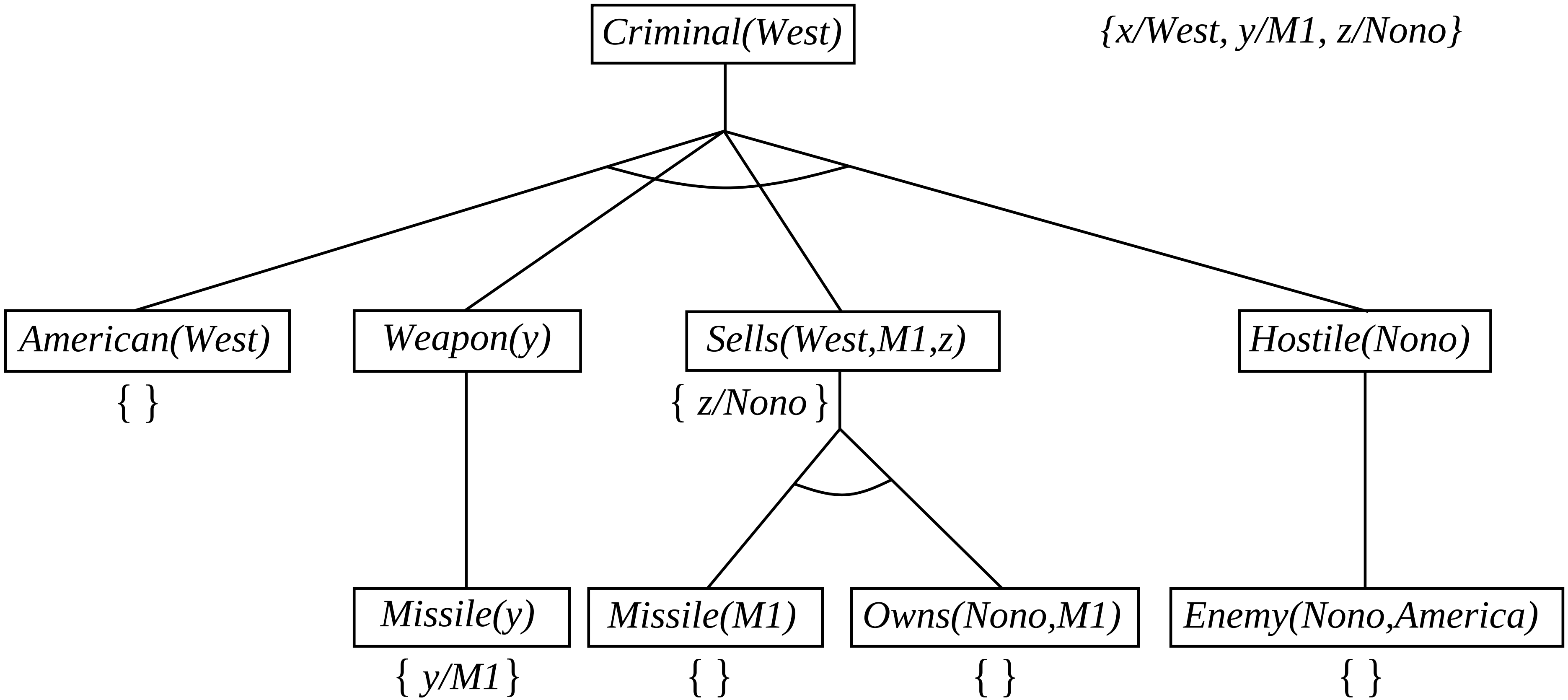


U $Sells$ uvrstimo ranije supstitucije $\{x/West\}$, $\{y/M_1\}$

$Missile(M_1) \wedge Owns(Nono, M_1) \Rightarrow Sells(West, M_1, Nono)$

daje supstituciju $\{z/Nono\}$ $\Rightarrow$ i treća grana je dokazana

Primjer ulančavanja unatrag



U $Hostile$ uvrstimo $\{z/Nono\}$, a to slijedi iz $Enemy$
Ostatak (nakon supstitucije) se svodi na poznatu činjenicu u bazi
$\Rightarrow$ cilj je dokazan i još imamo potrebno vezivanje varijabli.

Svojstva ulančavanja unatrag

Rekurzivno traženje dokaza u pretragom u dubinu (DFS):
— prostor je linearan u veličini dokaza

Nepotpun zbog (mogućih) beskonačnih petlji
Rješenje: provjerom trenutnog cilja obzirom na svaki cilj koji je već na stogu

Neefikasan zbog ponovljenih potciljeva (i za uspjeh i za neuspjeh)
Rješenje: spremanje prethodnih rezultata (dodatni prostor!)

U širokoj upotrebi (bez poboljšanja!) za logičko programiranje

Logičko programiranje

Mogući pogled: računanje kao zaključivanje na logičkim KB

	Logičko programiranje	Obično programiranje
1.	Identifikacija problema	Identifikacija problema
2.	Skupljanje informacija	Skupljanje informacija
3.	Čajanka :-)	Algoritam za rješenje problema
4.	Kodiranje informacija u KB	Programsko rješenje
5.	Kodiranje instance problema kao činjenica	Kodiranje instance problema kao podataka
6.	Sastavljanje upita	Primjena programa na podatke
7.	Pronalaženje pogrešnih činjenica	"Debuhiranje" proceduralnih grešaka

Trebalo bi biti jednostavnije naći grešku u $Capital(NewYork, US)$ nego u $x := x + 2$!

Prolog sistemi

Baza: ulančavanje unatrag s Hornovim klauzulama + "ukrasi"
U širokoj upotrebi u Europi i Japanu (baza za računala 5. generacije)
Tehnike kompajliranja ⇒ približavaju se milijardi LIPS

Program = skup klauzula = `head :- literal`$_1$`, ... literal`$_n$`.`

```
criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).
```

Efikasna unifikacija otvorenim kodom (engl. open coding)
Efikasno pronalaženje sparenih klauzula direktnim vezivanjem

Ulančavanje unatrag, prvo u dubinu, pa slijeva→udesno

Ugrađeni predikati za aritmetiku i sl., na primjer, `X is Y*Z+3`

Pretpostavka "zatvorenog svijeta" (= "negacija kao neuspjeh")
na primjer, dano `alive(X) :- not dead(X).`
`alive(joe)` uspijeva ako `dead(joe)` ne uspijeva

Prolog primjeri

Pretraga u dubinu od početnog stanja `X` (rekurzivno, uz završetak):

```
dfs(X) :- goal(X).
dfs(X) :- successor(X,S),dfs(S).
```

Nema potrebe za petljom po `S`: `successor` uspijeva za svaki

Spajanje dvije liste da se dobije treća:

```
append([],Y,Y).
append([X|L],Y,[X|Z]) :- append(L,Y,Z).

upit:      append(A,B,[1,2]) ?
odgovori:  A=[]    B=[1,2]
           A=[1]   B=[2]
           A=[1,2] B=[]
```

Više u posebnom dodatku o Prologu.

Rezolucija — kratki sažetak

Verzija rezolucije za logiku prvog reda je

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{(\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

gdje je $\text{UNIFY}(\ell_i, \neg m_j) = \theta$.

U logici sudova je dovoljno da je $\ell_i = \neg m_j$ (komplementarni literali), a ovdje ide jednakost supstitucija u njih i supstitucija u zaključak.

Na primjer,

$$\frac{\begin{array}{l}\neg Rich(x) \vee Unhappy(x) \\ Rich(Ken)\end{array}}{Unhappy(Ken)}$$

s unificirajućom supstitucijom $\theta = \{x/Ken\}$.

Izravna rezolucija za dokaz upita α nije potpuna (kao i prije).

Potpuna verzija za FOL $=$ rezolucija opovrgavanjem (kao i prije):
Primijeni korake rezolucije na $CNF(KB \wedge \neg\alpha)$

Pretvorba u CNF za FOL

Rečenica: "Svatko tko voli sve životinje je voljen od nekog":

$$\forall x\ [\forall y\ \ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ \ Loves(y,x)]$$

1. Eliminacija bikondicionala i implikacija $(a \Rightarrow b$ u $\neg a \vee b)$

$$\forall x\ [\neg\forall y\ \ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ \ Loves(y,x)]$$

2. Micanje $\neg$ prema unutra (De Morgan) za kvantifikatore i sudove

$$\neg\forall x\ \ p \ \equiv\ \exists x\ \ \neg p, \qquad \neg\exists x\ \ p \ \equiv\ \forall x\ \ \neg p$$

i eliminacija dvostrukih negacija:

$$\forall x\ [\exists y\ \ \neg(\neg Animal(y) \vee Loves(x,y))] \vee [\exists y\ \ Loves(y,x)]$$
$$\forall x\ [\exists y\ \ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ \ Loves(y,x)]$$
$$\forall x\ [\exists y\ \ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ \ Loves(y,x)]$$

Pretvorba u CNF za FOL (nastavak)

3. Standardizacija varijabli: svaki kvantifiktor treba uzeti različitu

$$\forall x \;\; [\exists y \;\; Animal(y) \wedge \neg Loves(x, y)\,] \vee [\,\exists z \;\; Loves(z, x)\,]$$

4. Skolemizacija: općenitija forma egzistencijskog instanciranja.
   Svaka egzistencijalna varijabla se zamjenjuje sa Skolem funkcijom okolnih univerzalno kvantificiranih varijabli koje na njih djeluju:

$$\forall x \;\; [\,Animal(F(x)) \wedge \neg Loves(x, F(x))\,] \vee Loves(G(x), x)$$

5. Ispuštanje univerzalnih kvantifikatora (nakon toga):

$$[\,Animal(F(x)) \wedge \neg Loves(x, F(x))\,] \vee Loves(G(x), x)$$

6. Distribucija $\wedge$ prema $\vee$:

$$[\,Animal(F(x)) \vee Loves(G(x), x)\,]$$
$$\wedge [\,\neg Loves(x, F(x)) \vee Loves(G(x), x)\,]$$

Dokaz rezolucijom — CNF (uz definitne klauzule)