INFERÊNCIA EM LÓGICA DE PRIMEIRA ORDEM CAP 9

Parcialmente adaptado de

http://aima.eecs.berkeley.edu

Resumo

- Redução de inferência em lógica de primeira ordem à inferência em lógica proposicional
- Unificação
- Modus Ponens Generalizado
- Encadeamento para a frente e para trás
- Programação em Lógica
- Resolução

Perspectiva Histórica

450 A.C.	Estóicos	logica proposicional, inferência (possivelmente)
322 A.C.	Aristóteles	“silogismos” (regras de inferência), quantificadores
1565	Cardano	teoria da probabilidade (logica proposicional + incerteza)
1847	Boole	lógica proposicional (novamente)
1879	Frege	lógica de primeira ordem
1922	Wittgenstein	prova por tabelas de verdade
1930	Gödel	$\exists$ algoritmo completo para LPO
1930	Herbrand	algoritmo completo para LPO (redução ao caso proposicional)
1931	Gödel	$\neg\exists$ algoritmo completo para aritmetica
1960	Davis/Putnam	algoritmo “eficaz” para logica proposicional
1965	Robinson	algoritmo “eficaz” para LPO – resolução

Métodos de Inferência para LPO

- Proposicionalização
- Resolução
- Sequentes
- Dedução Natural
- Tableaux
- Conexão de Matrizes
- Reescrita de termos

Instanciação Universal (IU)

- Qualquer instanciação de uma frase universalmente quantificada é consequência desta:

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

- Para qualquer variável v e termo básico (concreto) g.
- E.g. $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ origina:
    - $King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
    - $King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
    - $King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$
    - …

Instanciação Existêncial (IE)

- Para qualquer frase α, variável v, e símbolo de constante k que não ocorre na base de conhecimento:

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

- E.g. $\exists x\ Crown(x) \land OnHead(x, John)$ origina:
  - $Crown(C_1) \land OnHead(C_1, John)$
  - Desde que C_1 seja um novo símbolo de constante, designado por constante de Skolem
- Outro exemplo: de $\exists x\ d(x^y)/dy = x^y$ obtemos:
  - $d(e^y)/dy = e^y$
  - Desde que e seja um novo símbolo de constante.

Instanciações Universal e Existêncial

- IU pode ser aplicado repetidamente para adicionar novas frases;
    - a nova KB é logicamente equivalente à inicial
- IE pode ser aplicada uma vez para substituir a frase existencial;
    - a nova KB não é equivalente à inicial, mas é satisfazível sse a KB inicial era satisfazível!

Redução à inferência proposicional

- Suponhamos que a KB contém apenas o seguinte:
  - $\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$
  - $King(John)$
  - $Greedy(John)$
  - $Brother(Richard, John)$
- Instanciando a frase universal de todas as maneiras possíveis, ficamos com:
  - $King(John) \land Greedy(John) \Rightarrow Evil(John)$
  - $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$
  - $King(John)$
  - $Greedy(John)$
  - $Brother(Richard, John)$
- A nova KB foi proposicionalizada: os símbolos proposicionais são:
  - $King(John)$, $Greedy(John)$, $Evil(John)$, $King(Richard)$, etc.

Redução à inferência proposicional

- Proposição: Uma frase básica é consequência da nova KB sse é consequência da KB original
- Proposição: Toda a KB em LPO pode ser proposicionalizada preservando a relação de consequência lógica
- Ideia: proposicionalizar KB e pergunta, aplicar resolução, devolver resultado
- Problema: com símbolos de função, existe um número infinito de termos básicos,
  - e.g., $Father(Father(Father(John)))$

Redução à inferência proposicional

- Teorema: Herbrand (1930). Se frase α é consequência de uma KB em LPO, então é consequência de um subconjunto finito da KB proposicional
- Ideia: De $n=0$ até ∞ fazer:
  - gerar uma KB prop. instanciando os termos com profundidade-n
  - verificar se α é consequência desta KB
- Problema: funciona se é consequência, pode não terminar se α não é consequência
- Teorema: Turing (1936), Church (1936), consequência em LPO é semidecidível
  - Existem algoritmos que respondem “sim” a todas as frases que são consequência lógica, mas não existem algoritmos que também respondam “não” a todas as frases que não são consequencia lógica.

Problemas com a proposicionalização

- A proposicionalização pode gerar inúmeras frases irrelevantes.
  - E.g., de
    - $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
    - $King(John)$
    - $\forall y\ Greedy(y)$
    - $Brother(Richard, John)$
  - é obvio que $Evil(John)$, mas a proposicionalização produz muitos factos, por exemplo $Greedy(Richard)$, que são irrelevantes
- Com p predicados k-ários e n constantes, existem $p \cdot n^k$ instanciações!

Unificação

- Podemos imediatamente obter a conclusão se conseguirmos encontrar uma substituíção tal que $King(x)$ e $Greedy(x)$ concordem com $King(John)$ e $Greedy(y)$
  - $\theta=\{x/John, y/John\}$ funciona
- $Unificar(\alpha,\beta)=\theta$ se $\alpha\theta=\beta\theta$

α	β	θ
$Knows(John,x)$	$Knows(John,Jane)$	
$Knows(John,x)$	$Knows(y,OJ)$	
$Knows(John,x)$	$Knows(y,Mother(y))$	
$Knows(John,x)$	$Knows(x,OJ)$	

Unificação

- Podemos imediatamente obter a conclusão se conseguirmos encontrar uma substituíção tal que $King(x)$ e $Greedy(x)$ concordem com $King(John)$ e $Greedy(y)$
  - $\theta=\{x/John,y/John\}$ funciona
- $Unificar(\alpha,\beta)=\theta$ se $\alpha\theta=\beta\theta$

α	β	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*

- Standardização evita colisões de nomes de variáveis, e.g., $Knows(z_{17},OJ)$

Unificação

- Para unificar $Knows(John,x)$ e $Knows(y,z)$:
  - $\theta=\{y/John,x/z\}$ ou $\theta=\{y/John, x/John, z/John\}$
- A primeira substituição é mais geral do que a segunda.
- Existe sempre um unificador mais geral (most general unifier – MGU) que é único a menos de renomeação de variáveis.
  - $MGU=\{y/John,x/z\}$

Algoritmo de Unificação

function UNIFY(x, y, θ) **returns** a substitution to make x and y identical
inputs: x, a variable, constant, list, or compound
y, a variable, constant, list, or compound
θ, the substitution built up so far

if θ = failure **then return** failure
else if $x = y$ **then return** θ
else if VARIABLE?(x) **then return** UNIFY-VAR(x, y, θ)
else if VARIABLE?(y) **then return** UNIFY-VAR(y, x, θ)
else if COMPOUND?(x) **and** COMPOUND?(y) **then**
return UNIFY(ARGS[x], ARGS[y], UNIFY(OP[x], OP[y], θ))
else if LIST?(x) **and** LIST?(y) **then**
return UNIFY(REST[x], REST[y], UNIFY(FIRST[x], FIRST[y], θ))
else return failure

Algoritmo de Unificação (cont.)

function UNIFY-VAR(var, x, θ) **returns** a substitution
  inputs: var, a variable
    x, any expression
    θ, the substitution built up so far

  if $\{var/val\} \in \theta$ **then return** UNIFY(val, x, θ)
  else if $\{x/val\} \in \theta$ **then return** UNIFY(var, val, θ)
  else if OCCUR-CHECK?(var, x) **then return** failure
  else return add $\{var/x\}$ to θ

Modus Ponens Generalizado (MPG)

$$\frac{p_1', \ldots, p_n', \qquad p_1 \wedge \ldots \wedge p_n \Rightarrow q}{q\theta} \quad onde\ p_i'\theta = p_i\theta\ para\ todo\ i$$

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

- MPG utilizado com KB de cláusulas definidas (exactamente um literal positivo).
- Todas as variáveis estão implicitamente quantificadas universalmente.

Exemplo de Base de Conhecimento

- A lei afirma que é crime um Americano vender armas a nações hostis. O país Nono, inimigo da América, tem alguns mísseis, e todos esses mísseis forma vendidos pelo Coronel West, que é Americano.
- Provar que o Coronel West é criminoso.

Exemplo de Base de Conhecimento

- ... é crime um Americano vender armas a nações hostis:
  $American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)$
- Nono … tem alguns mísseis, i.e., $\exists x\ Owns(Nono,x) \land Missile(x)$:
  $Owns(Nono,M1)$ and $Missile(M1)$ (por IE)
- … todos os seus mísseis foram-lhe vendidos pelo Coronel West
  $Missile(x) \land Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$
- Mísseis são armas:
  $Missile(x) \Rightarrow Weapon(x)$
- Um enimigo da America é "hostil":
  $Enemy(x,America) \Rightarrow Hostile(x)$
- West, é Americano …
  $American(West)$
- O país Nono, é inimigo da América…
  $Enemy(Nono,America)$

- Recordar que todas as variáveis estão implicitamente quantificadas universalmente

Encadeamento para a frente

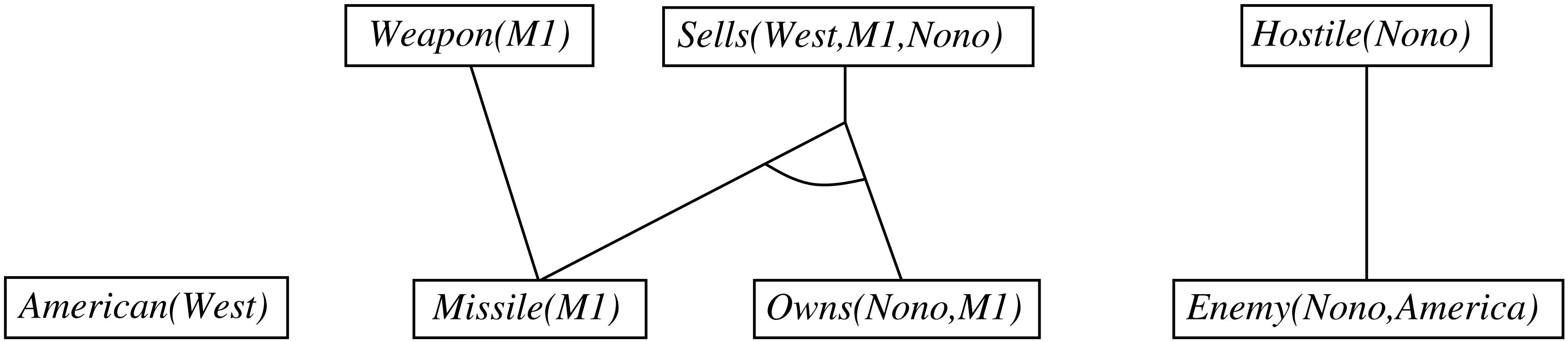
function FOL-FC-ASK(KB, α) **returns** a substitution or *false*

  repeat until *new* is empty
    $new \leftarrow \{\ \}$
    for each sentence r **in** KB **do**
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-APART(r)
      for each θ such that $(p_1 \wedge \ldots \wedge p_n)\theta = (p'_1 \wedge \ldots \wedge p'_n)\theta$
        for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST(θ, q)
        if q' is not a renaming of a sentence already in KB or *new* **then do**
          add q' to *new*
          $\phi \leftarrow$ UNIFY(q', α)
          if ϕ is not *fail* **then return** ϕ
    add *new* to KB
  return *false*

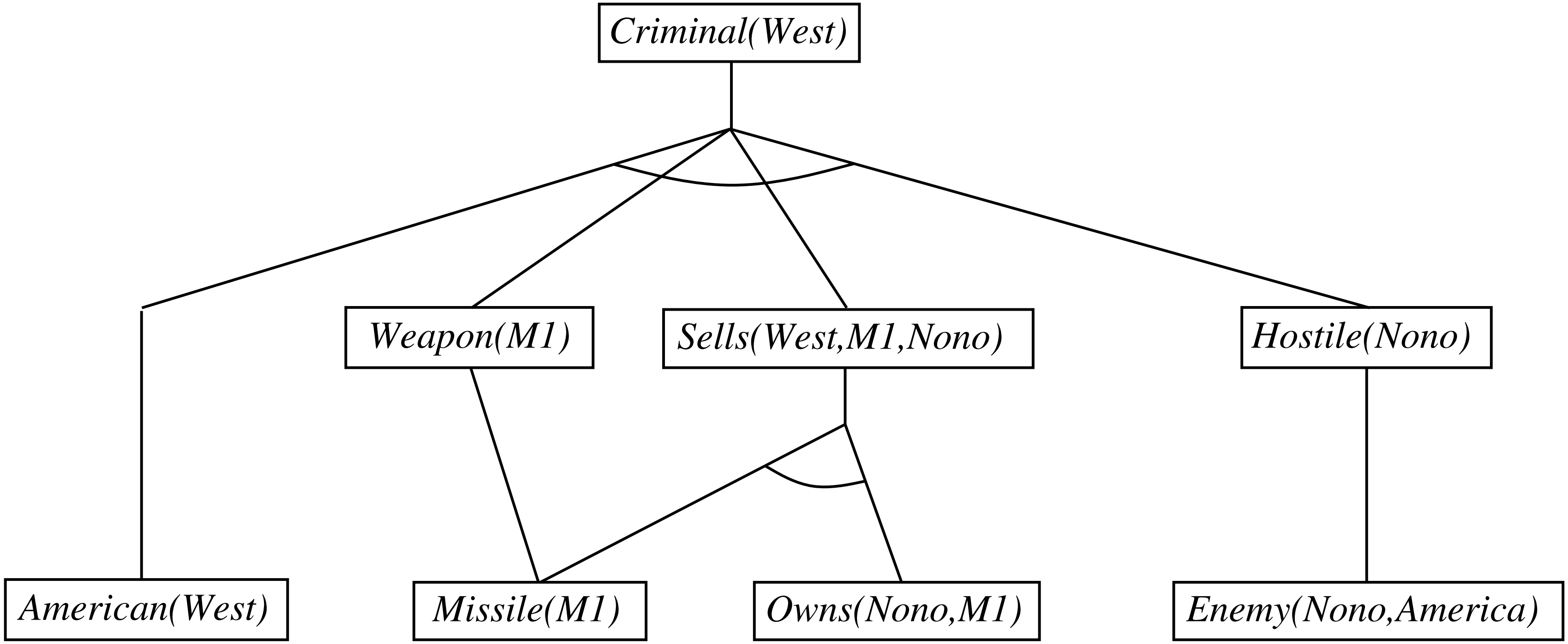
Prova por encadeamento para a frente

$\boxed{American(West)}$ $\boxed{Missile(M1)}$ $\boxed{Owns(Nono,M1)}$ $\boxed{Enemy(Nono,America)}$

Prova por encadeamento para a frente



Prova por encadeamento para a frente



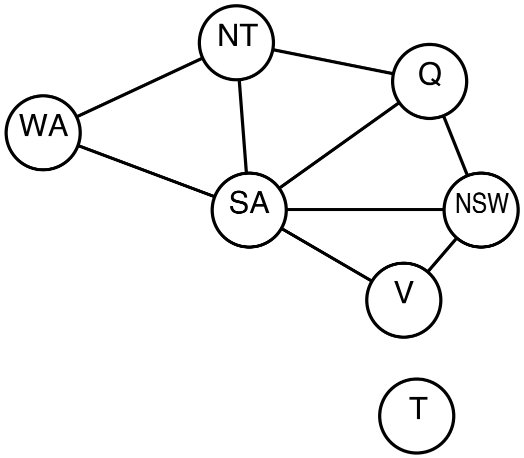
Propriedades do encadeamento para a frente

- Correcto para cláusulas definidas de primeira-ordem
  - Demonstração semelhante à do caso proposicional
- Datalog = cláusulas definidas de primeira-ordem + sem funções (e.g. KB crime)
  - EF termina para Datalog num número polinomial de iterações: limite máximo de $p \cdot n^k$ literais
    - p é o número de predicados, n o número de símbolos de constante e k a aridade máxima dos predicados
- Pode não terminar no caso geral se α não é consequência
  - Inevitável: consequência com cláusulas definidas é semidecidível

Eficiência do encadeamento para a frente

- Observação simples: não é necessário utilizar uma regra na iteração k se a premissa não foi adicionada na iteração *k-1*
  - Estratégia: testar regras cuja premissa contém apenas literais adicionados recentemente
- O mecanismo de concordância pode ser dispendioso
- Indexação nas Bases de Dados permite a obtenção em tempo O(1) de factos conhecidos.
  - E.g., a consulta Missile(x) devolve Missile(M1)
- Concordância de premissas conjuntivas com factos conhecidos é NP-difícil
- Encadeamento para a frente é amplamente utilizado em bases de dados dedutivas

Exemplo de concordância difícil



Diff(wa,nt) $\wedge$ Diff(wa,sa) $\wedge$ Diff(nt,q) $\wedge$
Diff(nt,sa) $\wedge$ Diff(q,nsw) $\wedge$ Diff(q,sa) $\wedge$
Diff(nsw,v) $\wedge$ Diff(nsw,sa) $\wedge$ Diff(v,sa) $\Rightarrow$
Colorable()

Diff(Red,Blue)	Diff (Red,Green)
Diff(Green,Red)	Diff(Green,Blue)
Diff(Blue,Red)	Diff(Blue,Green)

- Colorable() é inferido sse o CSP tem uma solução
- CSPs incluem 3SAT como caso especial, logo a concordância é NP-difícil

Encadeamento para trás

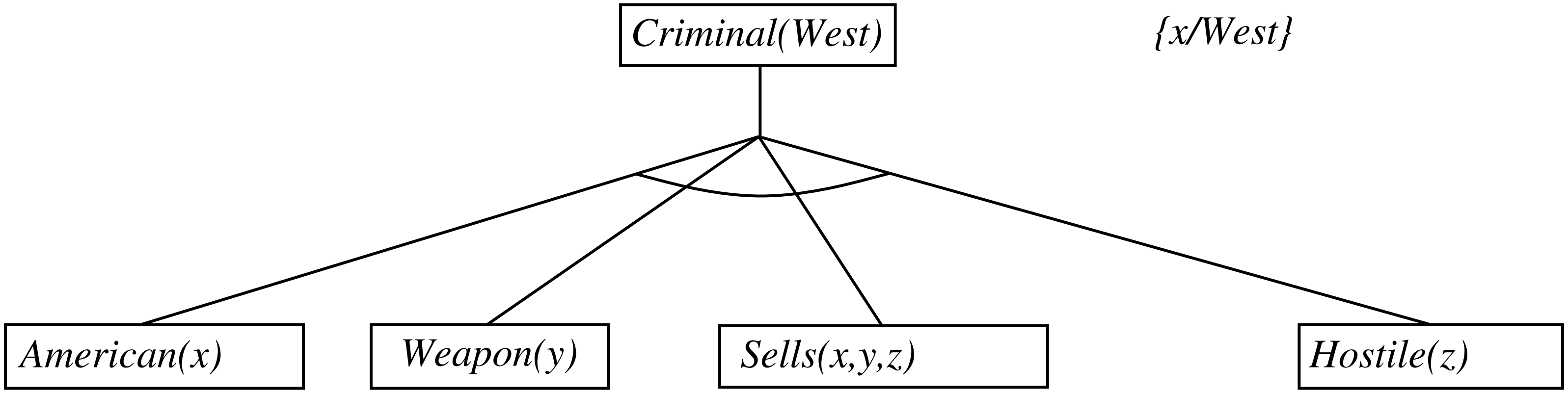
function FOL-BC-ASK(KB, $goals$, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query (θ already applied)
    θ, the current substitution, initially the empty substitution $\{\,\}$
  local variables: $answers$, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each sentence r **in** KB
    where STANDARDIZE-APART(r) $= (p_1 \land \ldots \land p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
   $new\_goals \leftarrow [p_1, \ldots, p_n | \text{REST}(goals)]$
   $answers \leftarrow$ FOL-BC-ASK(KB, new_goals, COMPOSE(θ', θ)) $\cup$ $answers$
  return $answers$

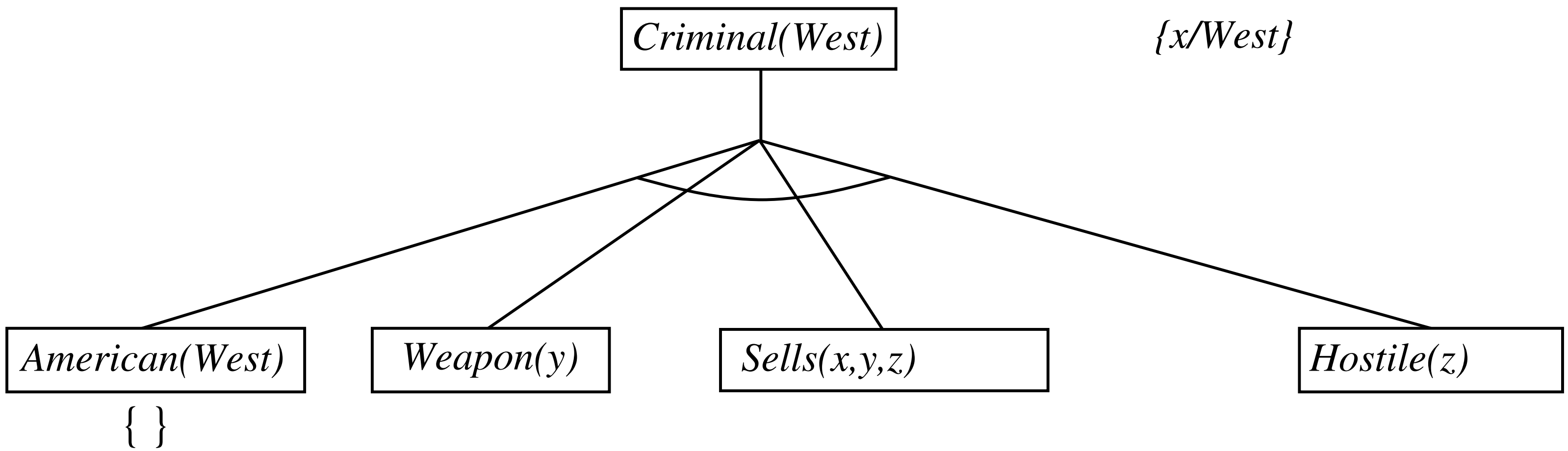
Prova por encadeamento para trás

$\boxed{Criminal(West)}$

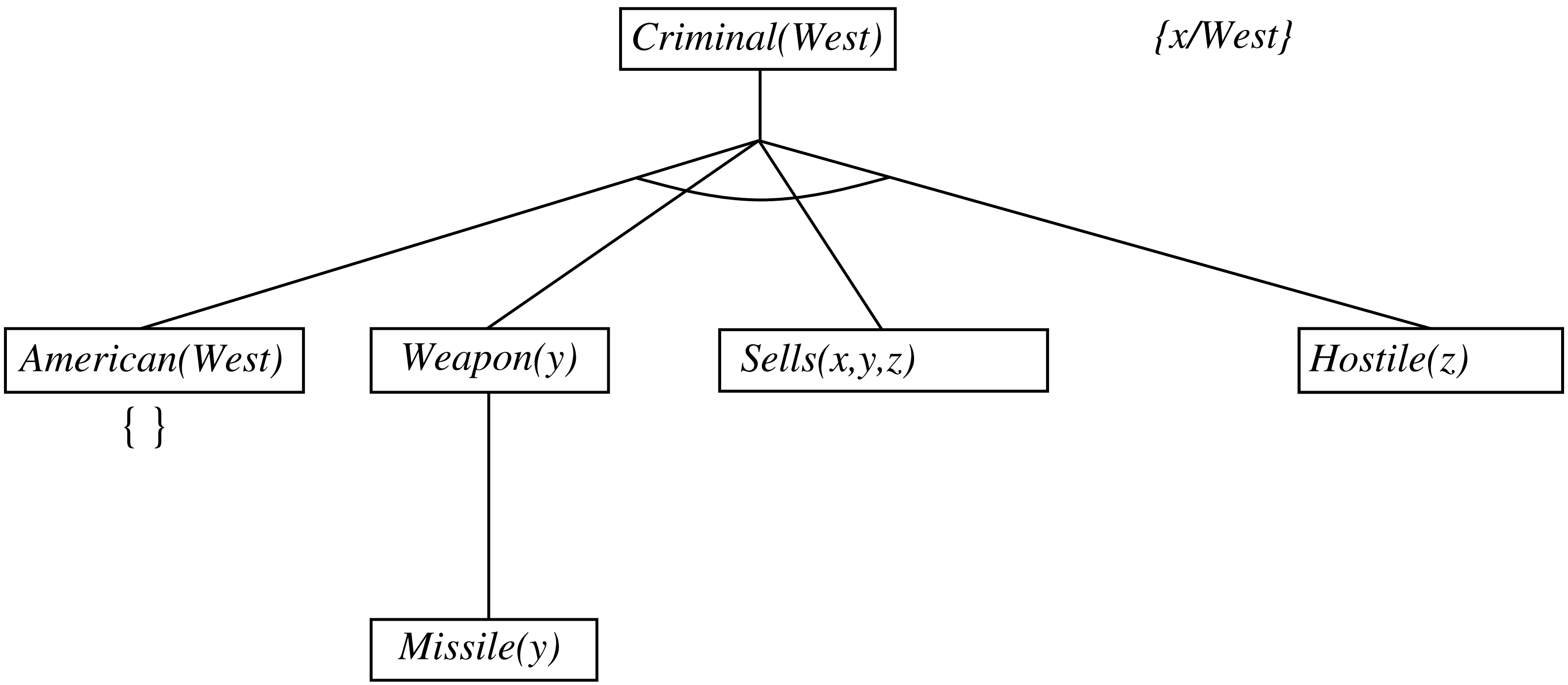
Prova por encadeamento para trás



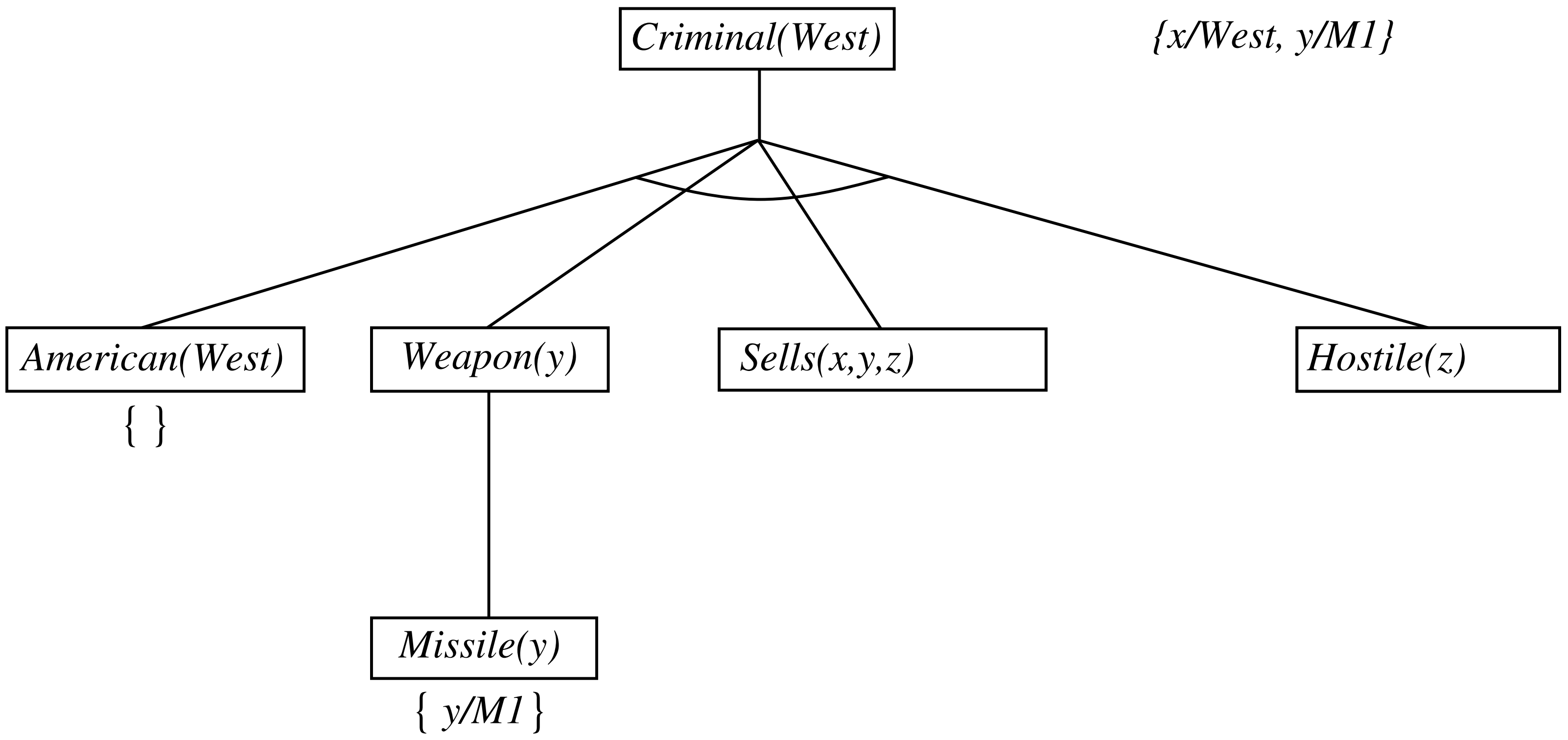
Prova por encadeamento para trás



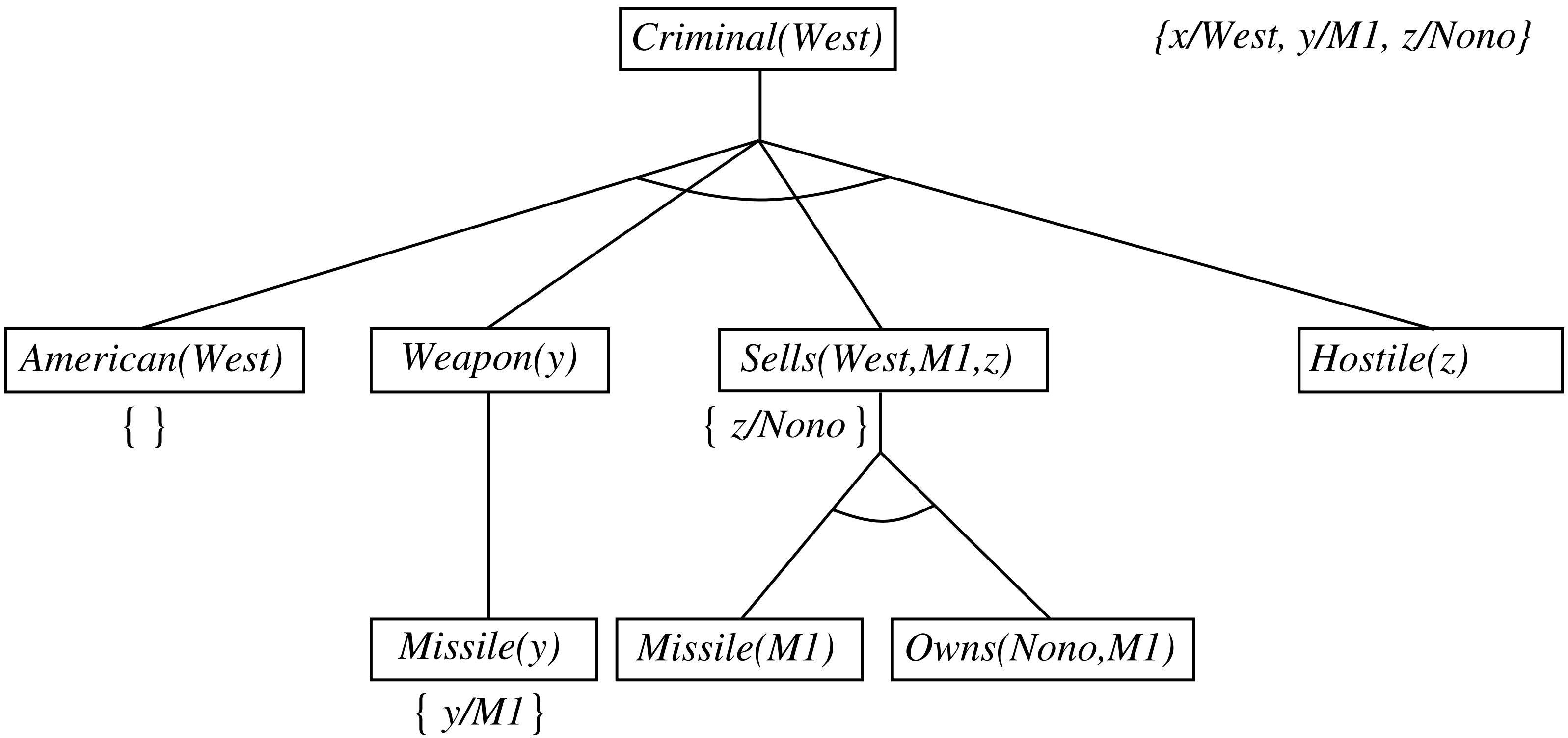
Prova por encadeamento para trás



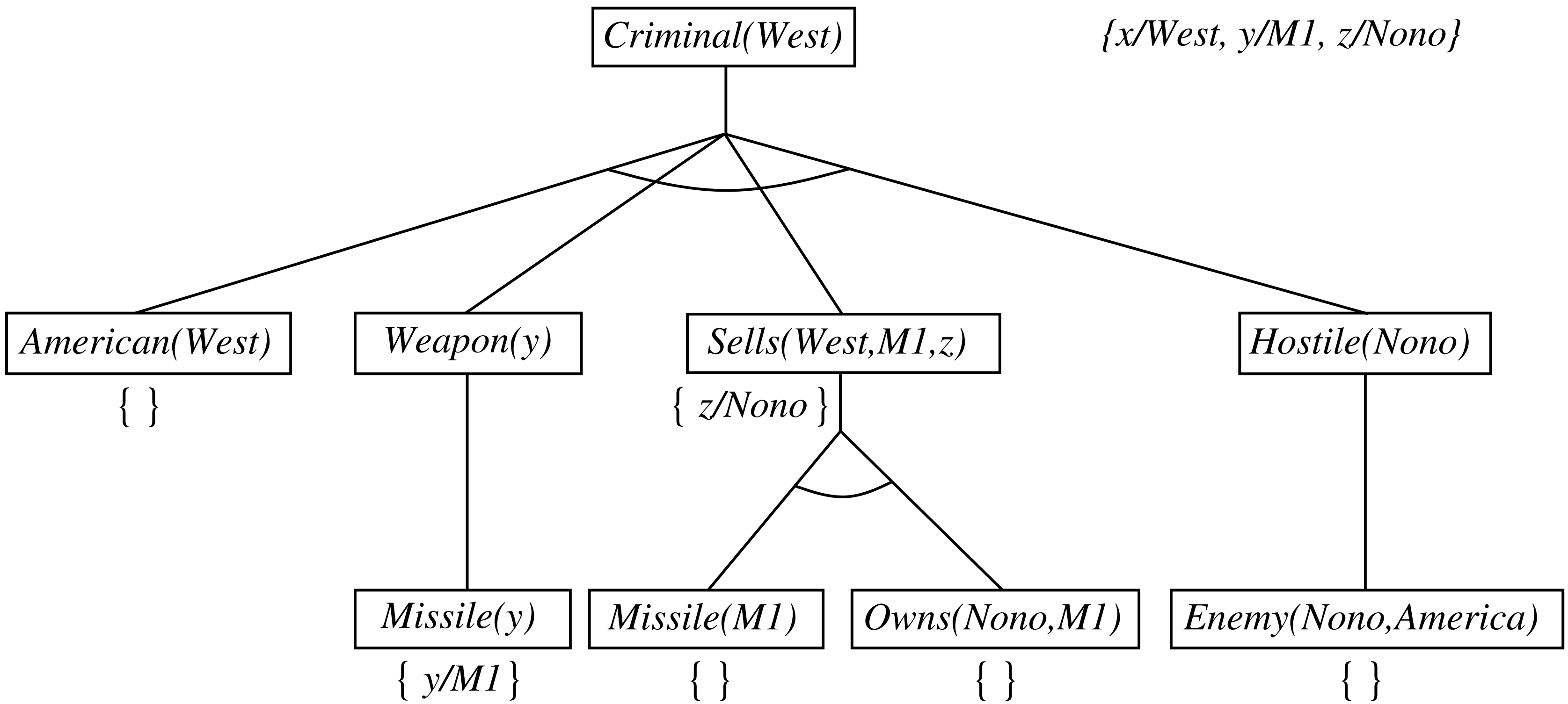
Prova por encadeamento para trás



Prova por encadeamento para trás



Prova por encadeamento para trás



Propriedades do encadeamento para trás

- Pesquisa da prova recursivamente em profundidade primeiro: espaço linear no tamanho da prova
- Incompleto devido a ciclos infnitos
  - Verificação do objectivo corrente com todos os outros na pilha
- Ineficiente devido às subconsultas repetidas (de sucesso e de falha)
  - Memorização dos resultados anteriores (espaço extra!)
- Amplamente utilizado (sem melhoramentos!) na programação em lógica

Programação em Lógica

Programação em Lógica

- Identificar problema
- Coligir Informação
- <pausa para café>
- Codificar informação na KB
- Representar instância com factos
- Efectuar consultas
- Encontrar factos errados

Programação Procedimental

- Identificar problema
- Coligir informação
- Descobrir solução
- Programar solução
- Representar instância com dados
- Aplicar programa aos dados
- Depurar erros procedimentais

- Deve ser mais facil depurar $Capital(NewYork, US)$ do que $x := x + 2$!

Programação em Lógica: Prolog

- Essência: encadeamento para trás com cláusulas de Horn
- Programa = conjunto de cláusulas = `head:-literal`$_1$`,…,literal`$_n$`.`

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

- Unficação eficiente sem teste de ocorrência
- Obtenção eficiente de cláusulas
- Encadeamento para trás em profundidade primeiro, da esquerda para a direita
- Predicados de sistema para efectuar aritmética etc. e.g., `X is Y*Z+3`
- Pressuposto do Mundo Fechado ("negação por falha")
  - e.g., dado `alive(X):-not dead(X).`
  - `alive(joe)` sucede se `dead(joe)` falha

Exemplo em Prolog

- Programa

```
criminal(X):-american(X),weapon(Y),sells(X,Y,Z),hostile(Z).
sells(west,X,nono):-missile(X),owns(nono,X).
owns(nono,m1).
missile(m1).
weapon(X):-missile(Y).
hostile(X):-enemy(X,america).
american(west).
enemy(nono,america).
```

- Interrogação

```
|?- criminal(Who).
Who = west;
no
```

Prova que West é criminoso em Prolog

```
                    ?- criminal(Who).
                              |
?- american(Who), weapon(Y1), sells(Who,Y1,Z1), hostile(Z1).
                              |                              Who = west
      ?- weapon(Y1), sells(west,Y1,Z1), hostile(Z1).
                              |
     ?- missile(Y1), sells(west,Y1,Z1), hostile(Z1).
                              |                              Y1 = m1
            ?- sells(west,m1,Z1), hostile(Z1).
                              |                              Z1 = nono
        ?- missile(m1), owns(nono,m1), hostile(nono).
                              |
             ?- owns(nono,m1), hostile(nono).
                              |
                    ?- hostile(nono).
                              |
                 ?- enemy(nono,america).
                              |
                             ?-
```

Resolução binária

- Forma Normal Conjuntiva (FNC – universal)
  - Conjunção de disjunções de literais com variáveis quantificadas universalmente
- Regra de inferência Resolução:

$$\frac{l_1 \vee \cdots \vee l_k, \qquad m_1 \vee \cdots \vee m_n}{\left(l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n\right)\theta}$$

em que $Unificar(l_i, \neg m_j) = \theta$. E.g.,

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

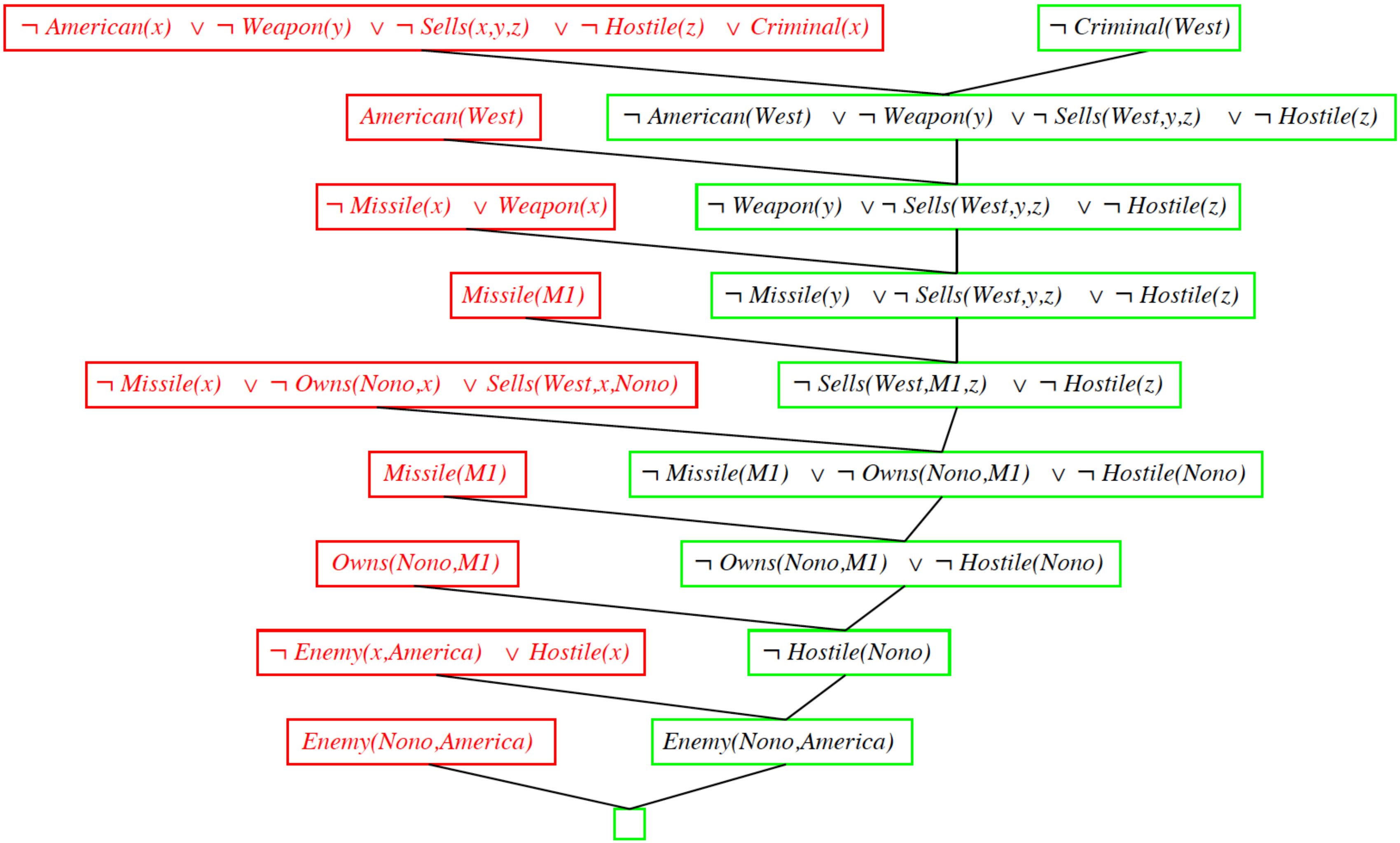
em que $\theta = \{x/Ken\}$.

- A regra da resolução binária, por sí só, não é completa.

Factorização

- Seja C' um subconjunto de literais com o mesmo sinal de uma cláusula C e unificável com unificador mais geral θ. A cláusula $C\theta$ é um factor de C.
- A regra de factorização autoriza a adição de qualquer factor de uma cláusula ao conjunto de cláusulas.
- Exemplo:
  1. $P(x, f(x), z) \vee P(u,w,w)$ axioma
  2. $\neg P(x, y, z) \vee \neg P(A, z, z)$ axioma
  3. $P(x1, f(x1), f(x1))$ factor de 1.
  4. $\neg P(A, z1, z1))$ factor de 2:
  5. $\square$ resolvente de 3. e 4.
- Aplicar resolução a CNF(KB$\wedge\neg\alpha$)
- Resolução Binária + Factorização $\Rightarrow$ completo para LPO

Prova por resolução: cláusulas definidas



Conversão para a FNC

- Toda a pessoa que ama todos os animais é amada por alguém

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

1. Eliminar bicondicionais e implicações

$$\forall x \left[\neg \forall y \neg Animal(y) \lor Loves(x,y)\right] \lor \left[\exists y\ Loves(y,x)\right]$$

2. Deslocar ¬ para dentro: $\neg \forall x\ p \equiv \exists x \neg p,\ \neg \exists x\ p \equiv \forall x \neg p$

$$\forall x \left[\exists y \neg\left(\neg Animal(y) \lor Loves(x,y)\right)\right] \lor \left[\exists y\ Loves(y,x)\right]$$

$$\forall x \left[\exists y \neg\neg Animal(y) \land \neg Loves(x,y)\right] \lor \left[\exists y\ Loves(y,x)\right]$$

$$\forall x \left[\exists y\ Animal(y) \land \neg Loves(x,y)\right] \lor \left[\exists y\ Loves(y,x)\right]$$

Conversão para a FNC

3. Standardizar variáveis: cada quantificador deve usar uma diferente.

$$\forall x \left[\exists y\ Animal(y) \land \neg Loves(x,y) \right] \lor \left[\exists z\ Loves(z,x) \right]$$

4. Skolemizar: uma forma mais geral de instanciação existencial. Cada variável existencial á substituída por uma função de Skolem das variáveis quantificadas universalmente que a incluem.

$$\forall x \left[Animal(F(x)) \land \neg Loves(x,F(x)) \right] \lor Loves(G(x),x)$$

5. Remover quantificadores universais.

$$\left[Animal(F(x)) \land \neg Loves(x,F(x)) \right] \lor Loves(G(x),x)$$

6. Distribuir $\land$ por $\lor$.

$$\left[Animal(F(x)) \lor Loves(G(x),x) \right] \land \left[\neg Loves(x,F(x)) \lor Loves(G(x),x) \right]$$

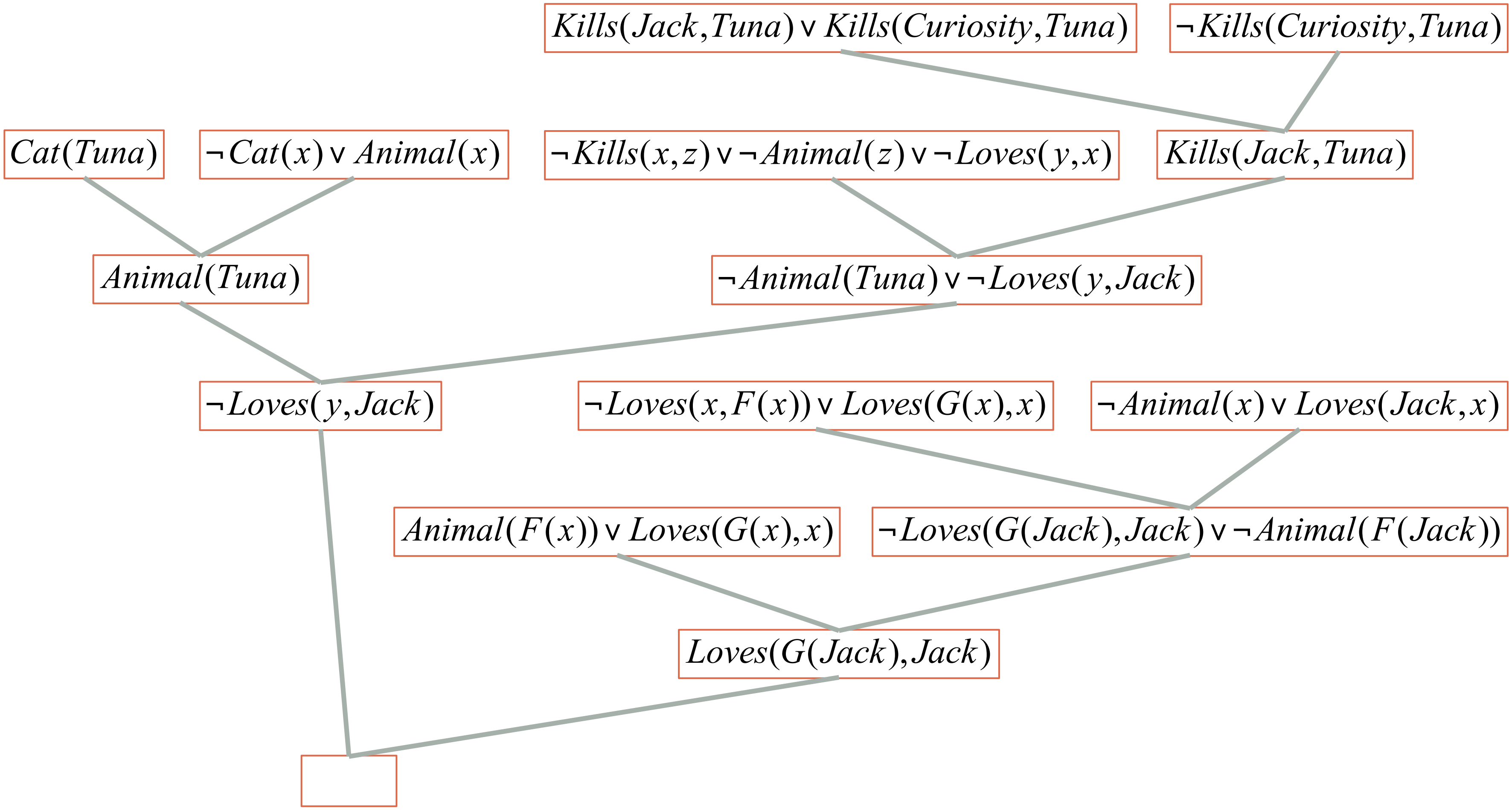
A Curiosidade matou o Gato?

- Toda a gente que ama todos os animais é amada por alguém.
  $\forall x \left[\forall y \; Animal(y) \Rightarrow Loves(x,y) \right] \Rightarrow \left[\exists y \; Loves(y,x) \right]$
- Qualquer pessoa que mata um animal não é amada por ninguém.
  $\forall x \left[\exists z \left(Animal(z) \wedge Kills(x,z) \right) \right] \Rightarrow \left[\neg \exists y \; Loves(y,x) \right]$
- Jack ama todos os animais.
  $\forall x \; Animal(x) \Rightarrow Loves(Jack,x)$
- O Jack ou a Curiosidade mataram o gato, que se chama Tuna.
  $Kills(Jack,Tuna) \vee Kills(Curiosity,Tuna)$
- Todos os gatos são animais
  $\forall x \; Cat(x) \Rightarrow Animal(x)$
- A curiosidade matou o gato?
  $\neg Kills(Curiosity,Tuna)$

Conversão para a FNC

- Toda a gente que ama todos os animais é amada por alguém.
  $Animal(F(x)) \lor Loves(G(x),x)$ $\neg Loves(x,F(x)) \lor Loves(G(x),x)$
- Qualquer pessoa que mata um animal não é amada por ninguém.
  $\neg Kills(x,z) \lor \neg Animal(z) \lor \neg Loves(y,x)$
- Jack ama todos os animais.
  $\neg Animal(x) \lor Loves(Jack,x)$
- O Jack ou a Curiosidade mataram o gato, que se chama Tuna.
  $Kills(Jack,Tuna) \lor Kills(Curiosity,Tuna)$
- Todos os gatos são animais
  $\neg Cat(x) \lor Animal(x)$
- A curiosidade matou o gato?
  $\neg Kills(Curiosity,Tuna)$

Prova que a Curiosidade matou o Gato



Tratamento da igualdade

- A igualdade introduz problemas adicionais no algoritmo de inferência. Existem duas grandes classes de abordagens para lidar com o predicado de igualdade:
- 1. Através da inclusão dos axiomas para a igualdade.
- 2. Recorrendo a regras de inferência adicionais.

Axiomas para a igualdade

- Axiomas básicos:

  $\forall x\ x=x$

  $\forall x\ \forall y\ \ x=y \Rightarrow y=x$

  $\forall x\ \forall y\ \forall z\ x=y \wedge y=z \Rightarrow x=z$

- Para cada predicado P/n e para cada $1 \leq i \leq n$

  $\forall x_1, \ldots, x_i, \ldots, x_n\ \forall y\ x_i=y \Rightarrow (P(x_1,\ldots,x_i,\ldots,x_n) \equiv P(x_1,\ldots,y,\ldots,x_n))$

- Para cada símbolo de função f/n e para cada $1 \leq i \leq n$

  $\forall x_1, \ldots, x_i, \ldots, x_n\ \forall y\ x_i=y \Rightarrow (f(x_1,\ldots,x_i,\ldots,x_n) \equiv f(x_1,\ldots,y,\ldots,x_n))$

- Recorre-se depois ao método de resolução binária com factorização.

Demodulação

- Para quaisquer termos x,y e z tal que m_i é um literal contendo z e UNIFY(x,z)=θ:

$$\frac{x = y, \qquad m_1 \vee \cdots \vee m_n}{SUB\big(SUBST(\theta, x), SUBST(\theta, y)\big) m_1 \vee \cdots \vee m_n}$$

  - Onde SUBST é a substituição normal e
  - SUB(x,y,m) substitui todas as ocorrências de x em m por y.
- Exemplo:

$$\frac{0 + z = z, \qquad P(0 + (0 + 2)) \vee Q(3)}{P(0 + 2) \vee Q(3)}$$

- A regra da Demodulação é incompleta.

Paramodulação

- Para quaisquer termos x,y e z tal que m_i é um literal contendo z e UNIFY(x,z)=θ:

$$\frac{l_1 \vee \cdots \vee l_k \vee x = y, \qquad m_1 \vee \cdots \vee m_n}{SUB\left(SUBST(\theta,x), SUBST(\theta,y), SUBST(\theta, l_1 \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_n\right)}$$

- Exemplo:

$$\frac{P(x_1) \vee f(x_1,h(y_1)) = g(x_1,y_1), \qquad Q(h(f(h(x_2),h(a))))}{P(h(x_2)) \vee Q(h(g(h(x_2),a)))}$$

com

$$x = f(x_1,h(y_1))$$

$$y = g(x_1,y_1)$$

$$z = f(h(x_2),h(a))$$

- A regra da Paramodulação é completa quando combinada com factorização, resolução binária e axiomas de reflexividade para variáveis e funções.

Estratégias de resolução

- Preferência pelas cláusulas unitárias – prefere resoluções envolvendo pelo menos uma cláusula contendo só um literal (cláusula unitária)
- Resolução Unitária – só efectua resoluções em que pelo menos uma das cláusulas é unitária. Método incompleto.
  - Para o caso de cláusulas de Horn, o método é completo. Assemelha-se ao encadeamento para a frente.
- Conjunto de suporte – identifica-se inicialmente um conjunto de cláusulas (o conjunto de suporte – set of support). Qualquer resolução combina uma cláusula do conjunto de suporte com outra cláusula, juntando a resolvente ao conjunto de suporte.
  - Se não houver cuidado, o método pode ser incompleto. Escolhe-se normalmente como conjunto de suporte inicial a negação da fórmula que se pretende demonstrar.

Estratégias de resolução

- Resolução de entrada (input resolution) – combina sempre uma das cláusulas de entrada (na base de conhecimento ou interrogação) com outra cláusula.
  - Completa para cláusulas de Horn.
- Resolução linear – Método completo em que se permite resolver P com Q desde que P esteja na base de conhecimento ou P seja uma antecessor de Q na árvore de prova.
  - Método completo.

Sumário

- Raciocínio em lógica de primeira ordem é semidecidível
- Regras de Instanciação Universal e Instanciação Existencial permitem reduzir inferência em LPO à inferência em lógica proposicional.
- Algoritmo de unificação permite encontrar o unificador mais geral entre 1 ou mais termos/atomos.
- A regra de Modus Ponens Generalizado é completa para cláusulas de Horn, mas semidecidível. Para o caso restrito Datalog, o problema da consequência lógica é decidível.
- Encadeamento para a frente pode ser utilizado em bases de dados dedutivas, sendo completo para programas Datalog.
- Encadeamento para trás é utilizado em sistemas de programação em lógica, tal como o Prolog, sofrendo de problemas de inferências redundantes e possibilidade de entrar em ciclo. Tabulação evita estes problemas.

Sumário

- Regra da resolução binária com factorização é completa para a refutação em LPO.
- Igualdade requer introdução de axiomas extra ou utilização de regras de inferência adicionais (e.g. paramodulação).
- Existem diversas estratégias para reduzir o espaço de procura em sistemas de resolução, sem sacrificar completude. Estes sistemas podem ser utilizados para demonstrar teoremas e para verificar e sintetizar software e hardware.