5-Lógica Matemática
Representação e Inferência

Inferência em lógica primeira ordem

Regras de inferência para quantificadores

1. Instanciação universal (IU): sentença α, variável v, instância básica g:

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

Exemplo:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$

$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$

$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

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

2. Instanciação existencial(IE): sentença α, variável v, constante k ($\neq$ ctes na KB):

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

Exemplo:

$\exists x\ Crown(x) \wedge OnHead(x, John)$

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

$\text{SUBST}(\{x/C_1\}, \alpha)$

$C_1 \equiv$ constante de Skolem

Redução à inferência proposicional

Ideia

1. IU para obter sentenças correspondentes à todas instâncias básicas
2. IE para obter sentença correspondente à instância básica
3. aplicar algoritmos inferência proposicionais

Exemplo: considerar a *KB*

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King\ (John)$

$Greedy(John)$

$Brother(Richard, John)$

substituições: $\{x/John\}$, $\{x/Richard\}$

KB após instanciação universal

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

símbolos proposicionais: *King* (*John*), *Greedy*(*John*), *Evil*(*John*), etc.

KB está *proposicionalizada*

inferência: resolução

- *KB* em LPO pode ser proposicionalizada preservando a implicação lógica (*entailment*)

- Sentença instanciada é consequência lógica da nova *KB* se e somente se ela é consequência lógica da *KB* original

- Ideia: proposicionalizar *KB* e consulta (*query*), aplicar resolução, retornar resultado

- Problema: para símbolos funcionais existem infinitas instanciações básicas
  Exemplo: *Father*(*Father*(*Father*(*John*)))

Teorema: Herbrand (1930)

Se a sentença α é consequência lógica de uma *KB* em LPO, então ela também é consequência lógica de um subconjunto **finito** de sentenças da *KB* proposicionalizada.

Ideia: for $n = 0$ to $\propto$ do

criar *KB* proposicional instanciando até profundidade-*n*

verificar se α é consequência lógica (*entailed by*) da *KB*.

Problema: pára se α é consequência lógica (*entailed*)

não pára se α não é consequência lógica (*is not entailed*)

Teorema: Turing (1936), Church (1936)

Consequência lógica em LPO é **semi-decidível** (existem algoritmos que terminam com *sim* quando uma sentença é consequência lógica, más não existem algoritmos que terminam com *não* quando a sentença não é consequência lógica).

Problemas com a proposicionalização

- proposicionalização gera muitas sentenças irrelevantes

  $\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
  $King\ (John)$
  $\forall y\ Greedy(y)$
  $Brother(Richard, John)$

- é evidente que $Evil\ (John)$, mas proposicionalização produz $Greedy(Richard)$, o que é irrelevante

- p predicados k-ários e n constantes $\rightarrow$ pn^k instanciações

Regra de inferência LPO: modus ponens generalizado

Exemplo: consulta (*query*) *Evil* (*x*)

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

King (*John*)

Greedy(*John*)

Brother(*Richard*, *John*)

substituição: $\theta = \{x/John\}$

– Ideia: encontrar algum *x* tal que *x* é *King* e *x* é *Greedy* e inferir que *x* é *Evil*

Em geral: se existe uma substituição θ que torna as premissas de uma implicação idêntica à um subconjunto de sentenças da *KB*, então inferimos o consequente (conclusão) da implicação após aplicar θ

Exemplo: consulta *Evil* (*x*)

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King\ (John)$

$\forall y\ Greedy(y)$

$Brother(Richard, John)$

$\theta = \{x/John, y/John\}$

aplicando θ à $King(x)$ e $Greedy(x)$ e à $Greedy\ (y)$ e $King\ (Jonh)$

$King\ (John)$, $Greedy\ (John)$

$King\ (John) \wedge Greedy\ (John) \Rightarrow Evil(John)$

inferimos $Evil(John)$

Modus ponens generalizado (MPG)

p_i', p_i, q: sentenças atômicas

$$\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i), \quad \forall\, i$$

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

– MPG: *KB* na forma de cláusulas definidas (exatamente um literal positivo)

– assume todas variáveis quantificadas universalmente

Teorema: MPG é consistente (*sound*)

Prova:

mostrar que se SUBST(θ, p_i') = SUBST(θ, p_i) , $\forall\ i$ então

$p_1', p_2', \ldots\ldots, p_n', (p_1 \wedge p_2 \wedge \ldots\ldots \wedge p_n \Rightarrow q) \models$ SUBST(θ, q)

lema: para qualquer sentença p e substituição θ, $p \models$ SUBST(θ, p) (prova: IU)

(1) $p_1 \wedge p_2 \wedge \ldots\ldots \wedge p_n \Rightarrow q \models$ SUBST(θ, $p_1 \wedge p_2 \wedge \ldots\ldots \wedge p_n \Rightarrow q$)

(2) SUBST(θ, $p_1 \wedge p_2 \wedge \ldots\ldots \wedge p_n \Rightarrow q$) $\models$ SUBST(θ, p_1) $\wedge \ldots . \wedge$ SUBST(θ, p_n) $\Rightarrow$ SUBST(θ, q)

(3) $p_1', p_2', \ldots\ldots, p_n' \models$ SUBST(θ, p_1') $\wedge \ldots . \wedge$ SUBST(θ, p_n')

(4) como SUBST(θ, p_i') = SUBST(θ, p_i), SUBST(θ, q) segue de (1)–(3) e modus ponens ordinário, pois de (2) SUBST(θ, p_1') $\wedge \ldots . \wedge$ SUBST(θ, p_n') $\Rightarrow$ SUBST(θ, q).

Exemplo:

$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King\ (John)$

$\forall y\ Greedy(y)$

$Brother(Richard, John)$

$p_1' = King\ (John)$ $\qquad$ $p_1 = King\ (x)$

$p_2' = Greedy\ (y)$ $\qquad$ $p_2 = Greedy\ (x)$

$\theta = \{x/John, y/John\}$ $\qquad$ $q = Evil(x)$

$SUBST(\theta, q) = Evil(John)$

Unificação

algoritmo unificação: dadas duas sentenças p e q, retorna unificador

UNIFY $(p, q) = \theta$ onde SUBST(θ, p) = SUBST(θ, q)

Exemplo

p	q	θ
Knows (*John*, *x*)	*Knows* (*John*, *Jane*)	{*x*/*Jane*}
Knows (*John*, *x*)	*Knows* (*y*, *Bill*)	{*x*/*Bill*, *y*/*John*}
Knows (*John*, *x*)	*Knows* (*y*, *Mother*(*y*))	{*x*/*Mother*(*John*), *y*/*John*}
Knows (*John*, *x*)	*Knows* (*x*, *Elizabeth*)	*falha*

Unificação

– padronização

Knows (*x*, *Elizabeth*) todo mundo conhece *Elizabeth* (João inclusive !)
Knows (*John*, *x*)

UNIFY(*Knows* (*John*, *x*), *Knows* (*z*, *Elizabeth*)) = {*x*/*Elizabeth*, *z*/*John*}

– unificador mais geral (UMG)

UNIFY(*Knows* (*John*, *x*), *Knows* (*y*, *z*)) = {*y*/*John*, *x*/*z*} ou
UNIFY(*Knows* (*John*, *x*), *Knows* (*y*, *z*)) = {*y*/*John*, *x*/*John*, *z*/*John*}
{*y*/*John*, *x*/*z*} = UMG (impõe menos restrições no valor da variável)

Algoritmo de unificação

The unification algorithm

```
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))
```

The unification algorithm

Algoritmo de unificação (alternativo)

Determinar o unificador mais geral para: $E = \{P(A,f(x)),P(y,z)\}$

1–Localizar os termos mais à esquerda em que os literias diferem

Considerar cada um dos literais em E como uma cadeia se símbolos

Extrair de cada um dos literais os termos que começam na i-ésima posição formando o *conjunto de discórdia* D dos literais

$D_1 = \{A, y\}$

D_1 contém uma variável (y) e um termo (A) onde y não ocorre, podemos construir a substituição $\theta_1 = \{y/A\}$ que aplicada a E torna literais iguais

$E\theta_1 = \{P(A, f(x)), P(A, z)\}$

2–Repetir o processo sobre $E\theta_1 = \{P(A, f(x)), P(A, z)\}$

O novo *conjunto de discórdia* D_2 dos literais

$D_2 = \{f(x), z\}$

Novamente, D_2 contém uma variável (z) e um termo $f(x)$ onde (z) não ocorre

Podemos construir a substituição $\theta_2 = \{z/f(x)\}$ que aplicada a $E\theta_1$ torna literais idênticos $(E\theta_1)\theta_2 = \{P(A, f(x))\}$

3–O unificador mais geral (UMG) θ = composição de θ_1 e θ_2

$\theta = \text{COMPOSE}(\theta_1, \theta_2) = \{y/A, z/f(x)\}$

function UNIFICA (*E*) **returns** UMG, se possível

input: *E* um conjunto de expressões simples

output: θ unificador mais geral

begin

θ ← ε = { }

W ← *E*

D ← conjunto de discórdia de *W*

while |*W*| > 1 e *D* satisfaz teste de ocorrência **do**

begin

selecionar uma variável *x* e um termo *t* em *D* tal que *x* não ocorre em *t*

θ ← COMPOSE (θ , {*x*/*t*})

W ← SUBST ({*x*/*t*}, *W*)

D ← conjunto de discórdia de *W*

end

if |*W*| = 1 **then return** θ

else return *falha*

end

Encadeamento direto (*forward chaining*)

Cláusulas definidas de primeira ordem

– disjunção de literais, onde no máximo um é positivo
  - sentença atômica
  - implicação

Exemplo:

$King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King\ (John)$

$Greedy(y)$

– literais podem incluir variáveis

– variáveis universalmente quantificadas (omite-se quantificador)

Exemplo

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.

Prove that Colonel West is a criminal

- Base de conhecimento: cláusulas definidas primeira ordem

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

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

Nono … has some missiles, i.e., $\exists x\ Owns(Nono, x) \land Missile(x)$

$$Owns(Nono, M_1) \land Missile(M_1) \equiv Owns(Nono, M_1)\ ,\ \ Missile(M_1)$$

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

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

Missiles are weapons

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

An enemy of America counts as "hostile "

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

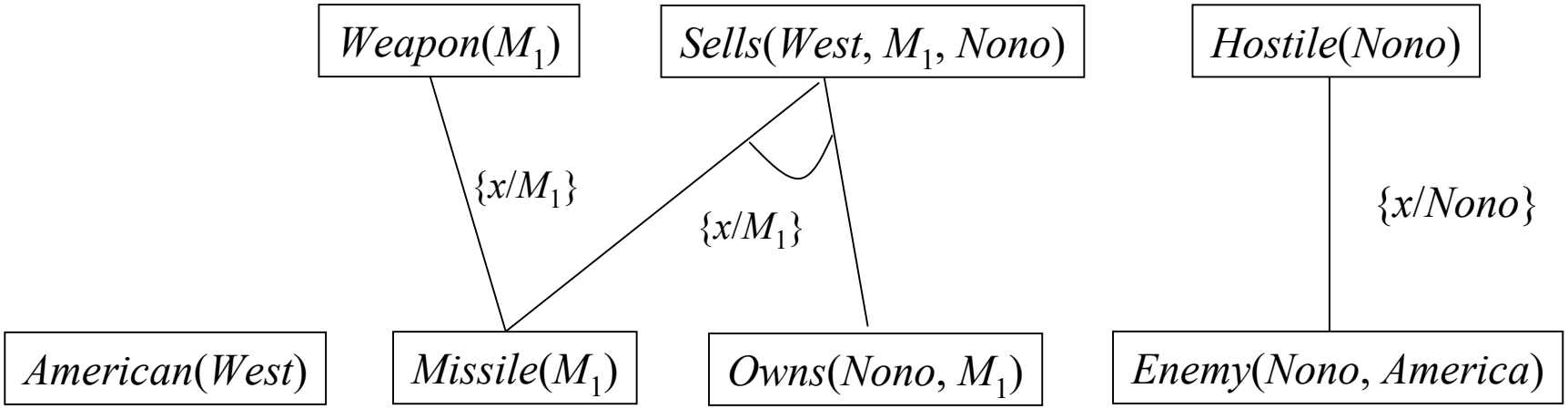
West, who is American …

$$American(West)$$

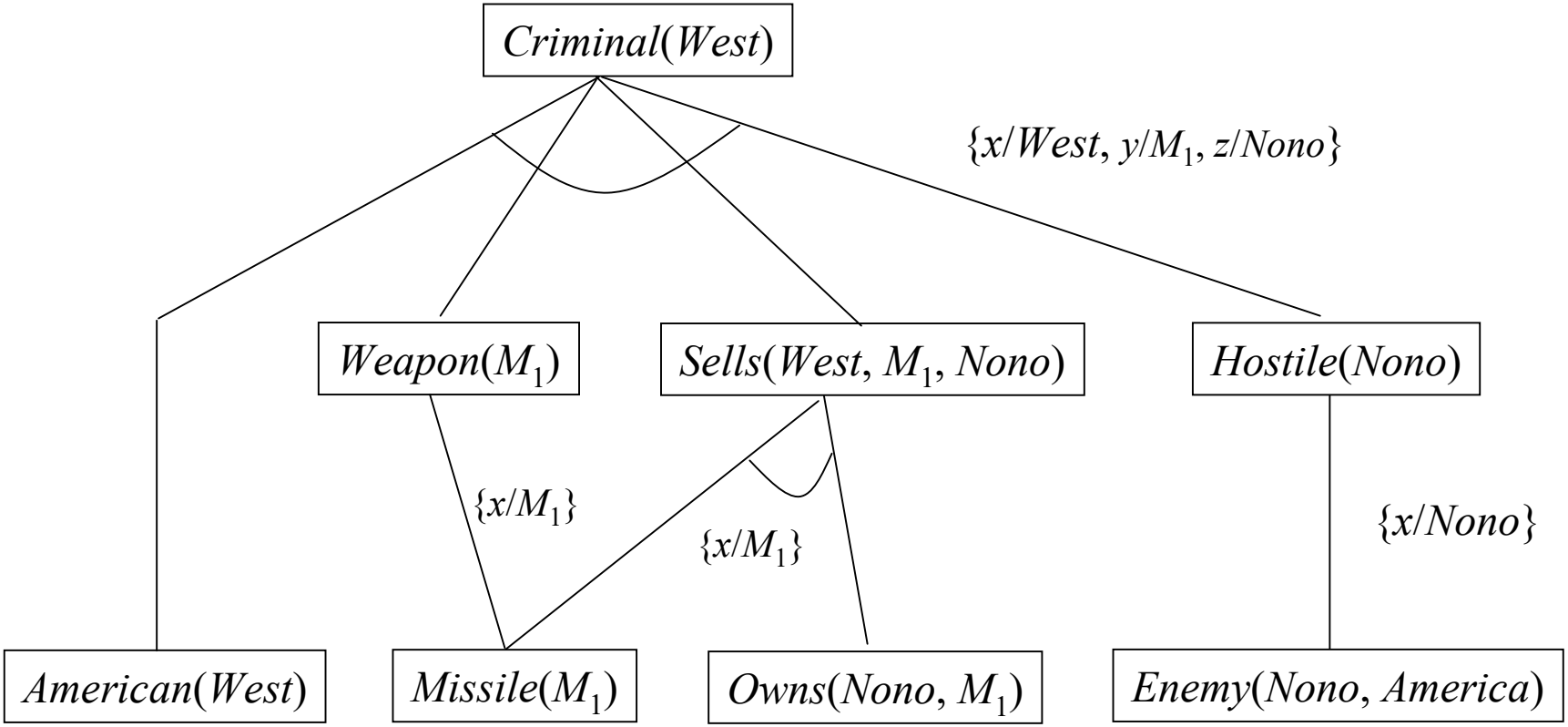
The country Nono, an enemy of America …

$$Enemy(Nono, America)$$

$American(West)$ $Missile(M_1)$ $Owns(Nono, M_1)$ $Enemy(Nono, America)$



Árvore de derivação



Esta *KB* é um ponto fixo do algoritmo de inferência

Algoritmo de encadeamento direto

Forward chaining algorithm

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**

- Características do algoritmo de encadeamento direto

  – consistente e completo
    - para cláusulas definidas
    - problemas com cláusulas definidas com símbolos funcionais
    - pode não terminar (símbolos funcionais, e.g. axiomas de Peano)

      $NatNum\ (0)$

      $\forall n\ NatNum\ (n) \Rightarrow NatNum(S(n))$

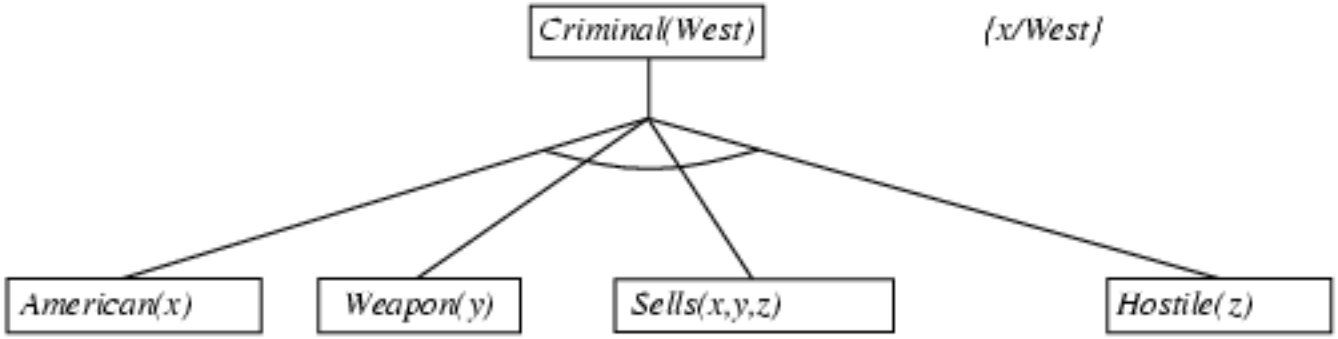
  – eficiência
    - *pattern matching*
    - encadeamento direto incremental (Rete)
    - fatos irrelevantes

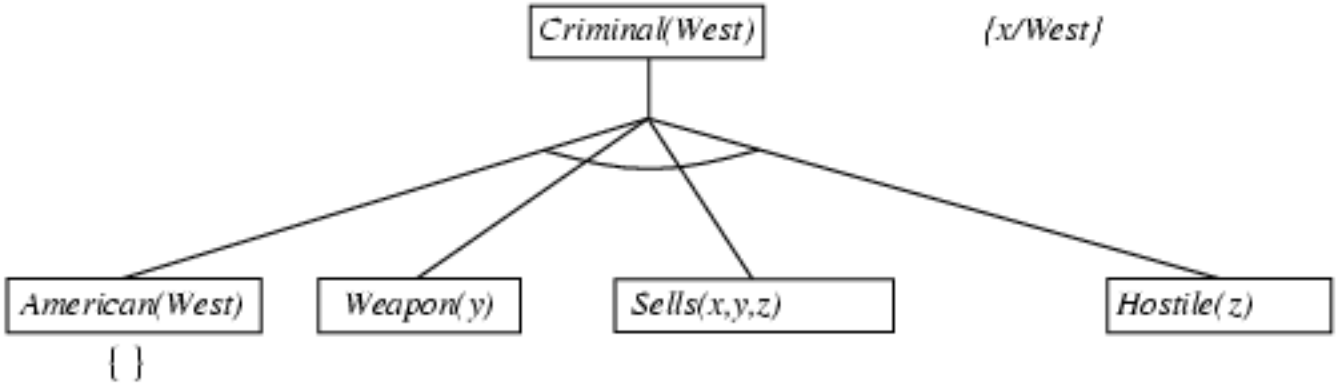
Encadeamento reverso (*backward Cchaining*)

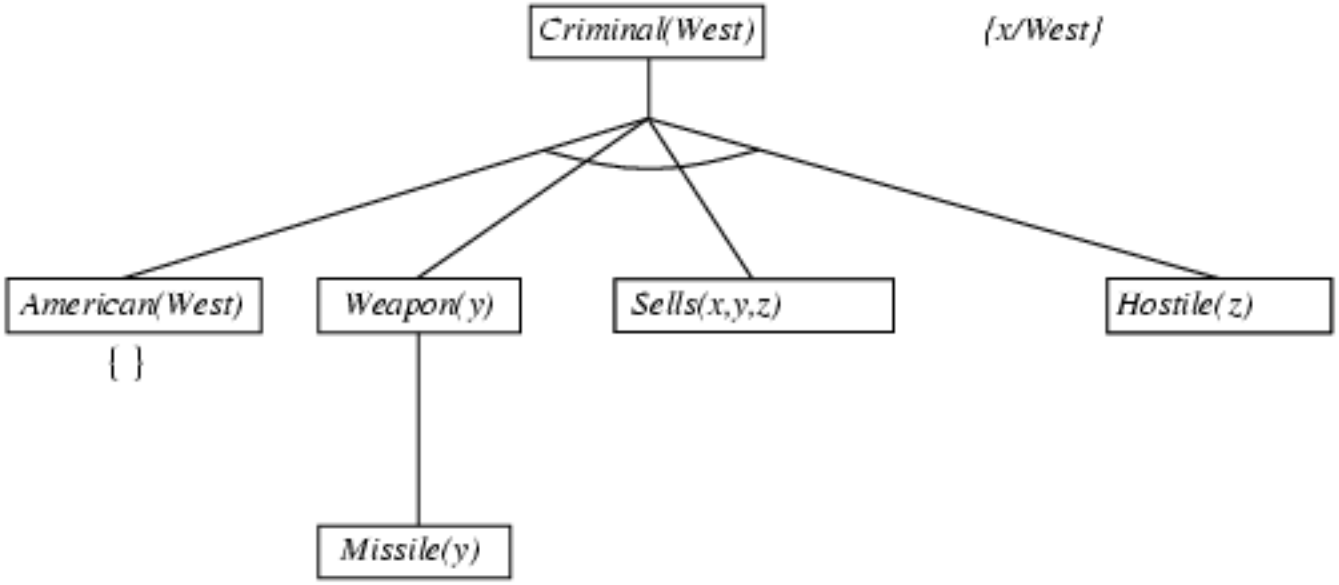
function FOL-BC-ASK(KB, goals, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query
    θ, the current substitution, initially the empty substitution $\{\}$
  local variables: ans, a set of substitutions, initially empty

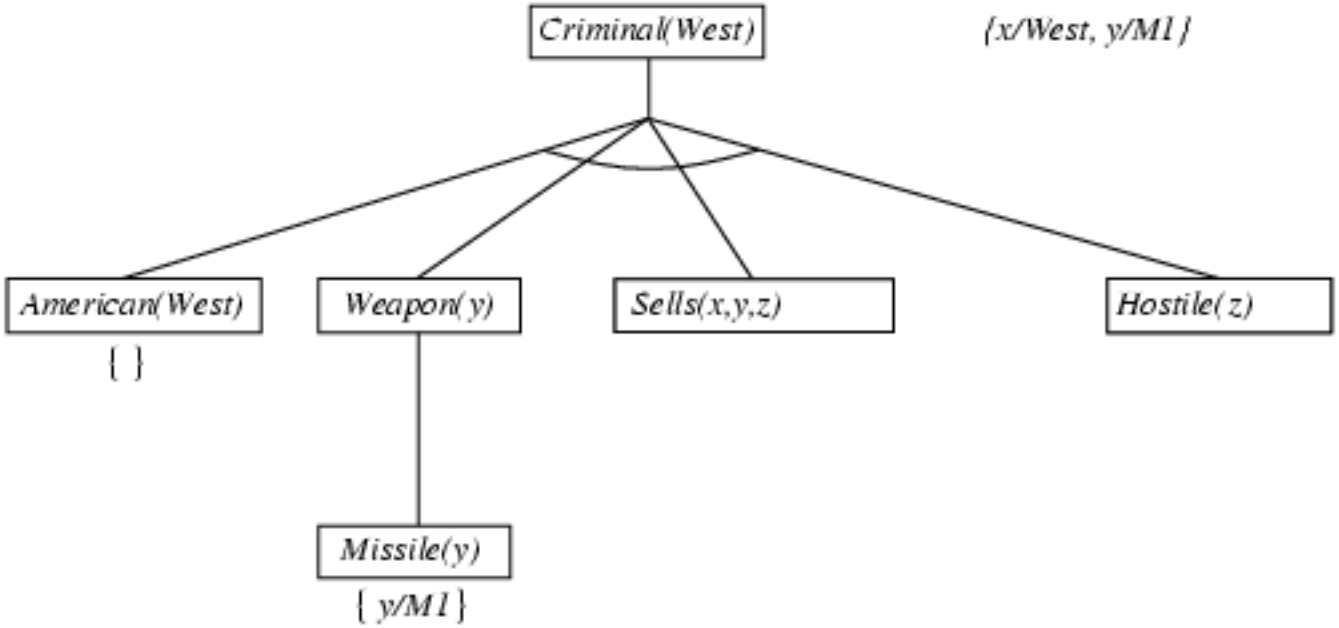
  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow$ SUBST(θ, FIRST($goals$))
  for each r **in** KB where STANDARDIZE-APART(r) $= (p_1 \wedge \ldots \wedge p_n \Rightarrow q)$
    and $\theta' \leftarrow$ UNIFY(q, q') succeeds
    $ans \leftarrow$ FOL-BC-ASK(KB, $[p_1, \ldots, p_n | \text{REST}(goals)]$,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

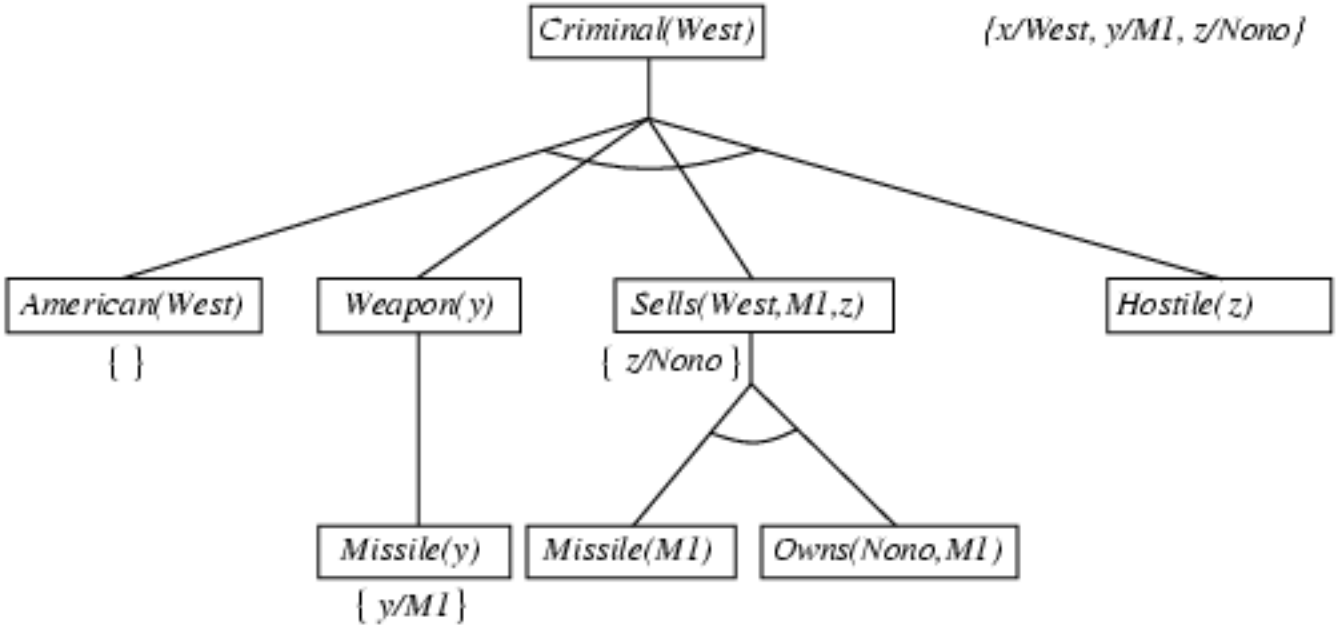
SUBST(COMPOSE(θ_1, θ_2), p) = SUBST(θ_2, SUBST(θ_1, p))

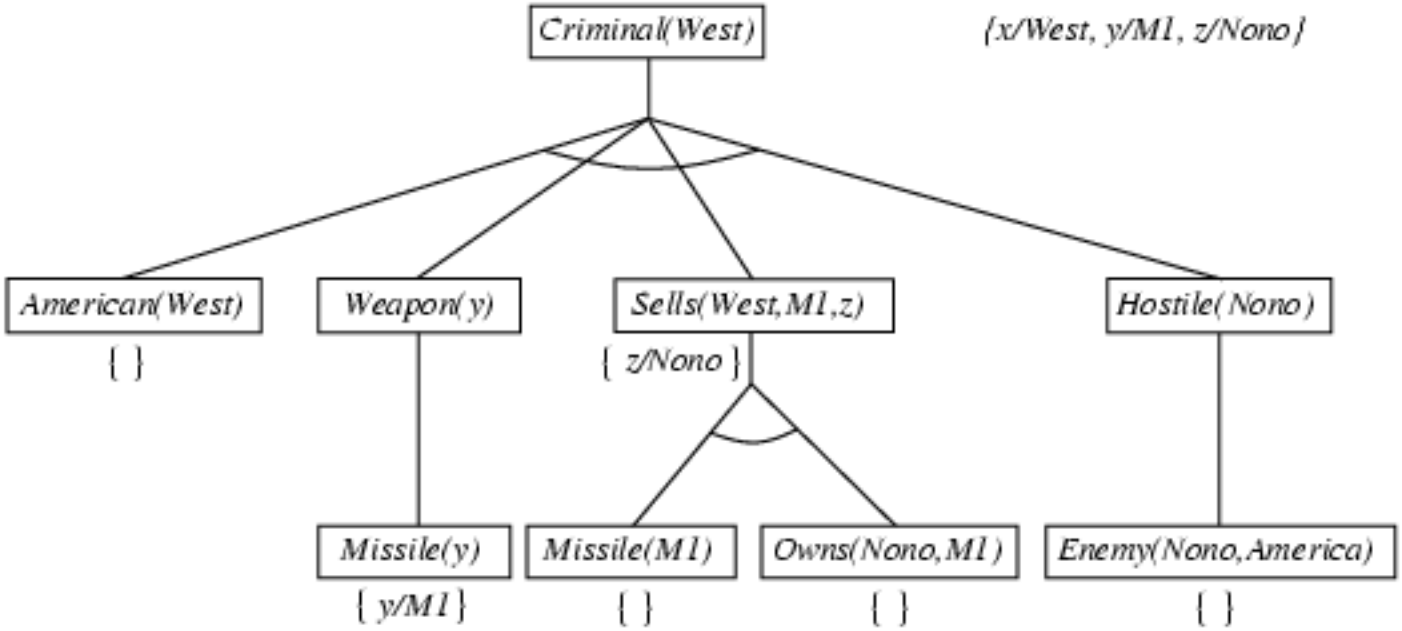




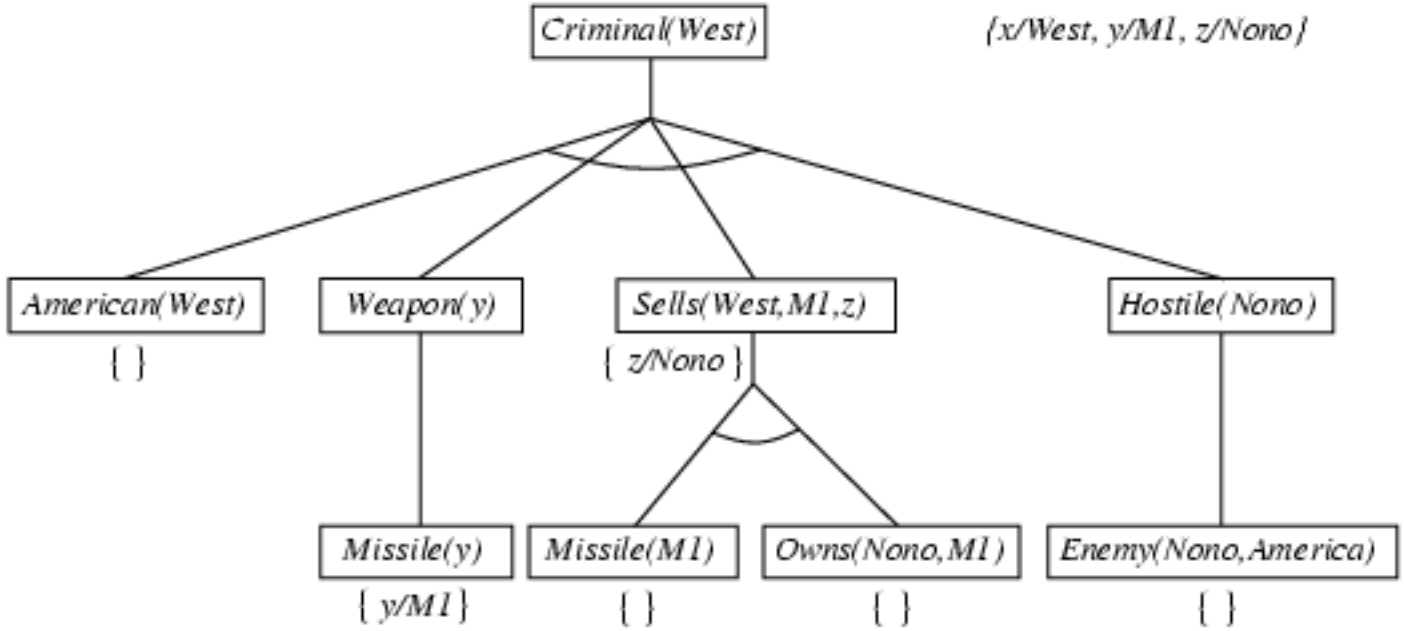








Árvore de derivação



- Características do algoritmo de encadeamento reverso

  – algoritmo de busca em profundidade
    - complexidade espacial linear

  – incompleto devido a possibilidade de ciclos (*loops*) infinitos
    verificar se meta corrente está na lista de metas (*goals*)

  – ineficiente devido à repetição de submetas (ambos, sucesso e falha)
    cache de resultados anteriores (memória extra)

  – aplicações: programação em lógica, sistemas especialistas (diagnóstico)

Resolução

Forma normal conjuntiva (FNC) para LPO

– resolução requer sentenças na FNC

– FNC é uma conjunção de cláusulas
  • cláusula: disjunção de literais
  • literais podem conter variáveis
  • literais com variáveis: supomos quantificados universalmente

Exemplo:

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

$$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$$

Algoritmo: converter sentenças LPO para FNC

1–Eliminar implicações: $(p \Rightarrow q) \equiv (\neg p \vee q)$

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

2–Mover $\neg$ para o interior:

$$\neg\, (p \vee q) \equiv \neg\, p \wedge \neg\, q$$
$$\neg\, (p \wedge q) \equiv \neg\, p \vee \neg\, q$$
$$\neg\, \forall x\, p \equiv \exists x\, \neg\, p$$
$$\neg\, \exists x\, p \equiv \forall x\, \neg\, p$$

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

3-Padronizar variáveis: $(\forall x\ P(x)) \vee (\exists x\ Q(x)) \equiv (\forall x\ P(x)) \vee (\exists y\ Q(y))$

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

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

4-Skolemizar: substituir variáveis quantificadas $\exists$ por funções das variáveis universalmente quantificadas em cujo escopo o quantificador $\exists$ está (funções de Skolem)

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

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

5-Remover quantificadores $\forall$: $p \vee \forall x\, q \equiv \forall x\, p \vee q$ se x não ocorre em p

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

6-Distribuir $\wedge$ sobre $\vee$: $(p \wedge q) \vee r \equiv (p \vee r) \wedge (q \vee r)$

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

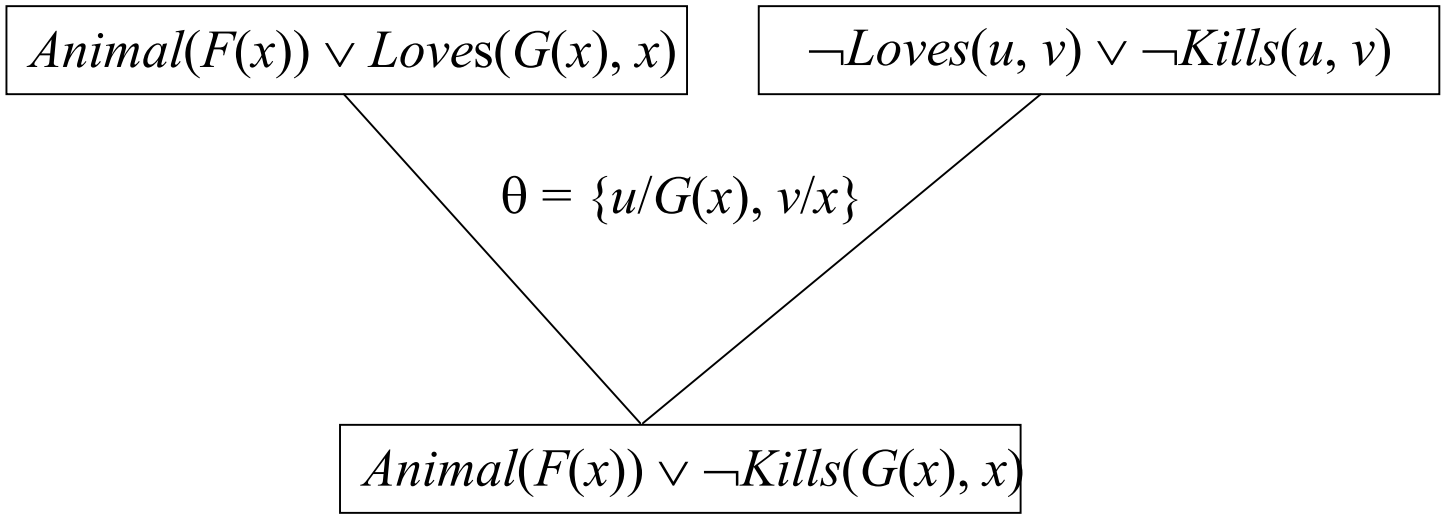
Regra da resolução

l_i, m_j : literais

UNIFY(l_i, $\neg m_j$) = θ

$$\frac{\ell_1 \vee \ldots \vee \ell_k, \quad m_1 \vee \ldots \vee m_n}{\text{SUBST}(\theta, \ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n)}$$

Exemplo



Algoritmo de resolução LPO

Algoritmo de resolução LPO

1. iniciar convertendo $(KB \wedge \neg\alpha)$ para FNC
2. aplicar regra resolução PMO às cláusulas resultantes
3. resolver literais para produzir novas cláusulas
4. continuar processo até que
   4.1 se novas cláusulas não são obtidas, então $KB \not\models \alpha$
   4.2 se resolução produz cláusula vazia , então $KB \models \alpha$

Exemplo

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

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

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

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

$Owns(Nono, M_1)$

$Missile(M_1)$

$American(West)$

$Enemy(Nono, America)$

$\neg Criminal(West)$

FNC e cláusulas

$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x, y, z) \vee \neg Hostile(z) \vee Criminal(x)$

$\neg Missile(x) \vee \neg Owns(Nono, x) \vee Sells(West, x, Nono)$

$\neg Enemy(x, America) \vee Hostile(x)$

$\neg Missile(x) \vee Weapon(x)$

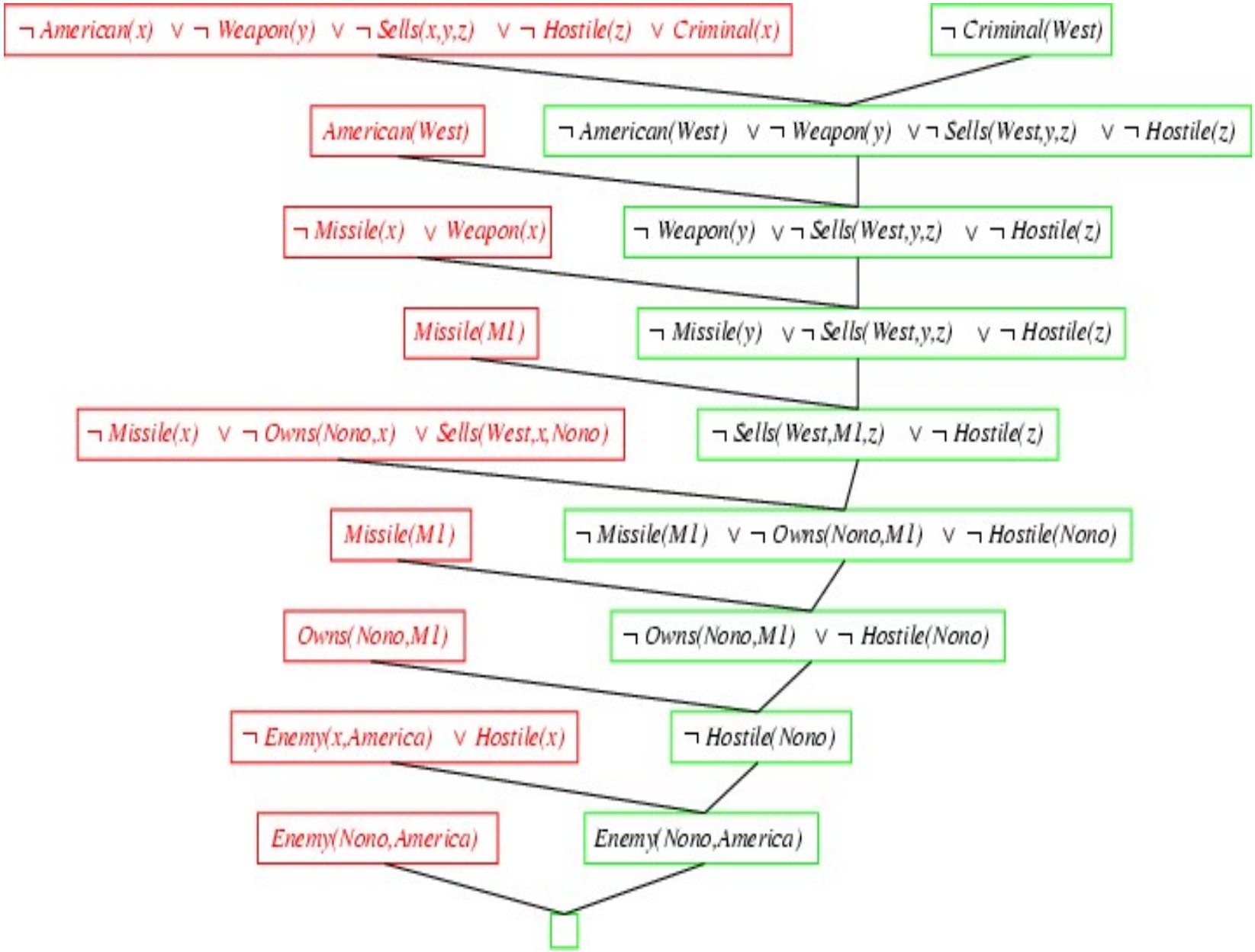
$Owns(Nono, M_1)$

$Missile(M_1)$

$American(West)$

$Enemy(Nono, America)$

$\neg Criminal(West)$



Seleção de cláusulas para algoritmo de resolução

1. Busca em amplitude (*Breadth-first*)

   determinar todos resolventes de 1° nível

   (resolventes entre cláusulas do conjunto base)

   ……..

   determinar todos resolventes de *i*–ésimo nível, etc.

   (resolventes cujos pais mais profundos são resolventes do nível (*i*–1))

2. Conjunto suporte

   pelo menos um pai de cada resolvente é selecionado entre as cláusulas

   resultantes da negação do que sequer provar ou de seus descendentes

3. Preferência unitária

caso particular do método conjunto suporte onde ao invés de se prosseguir como *breadth-first*, seleciona-se uma claúsula com um único literal (ou mínimo)

4. Linear

cada resolvente possui um pai pertencente ao conjunto base

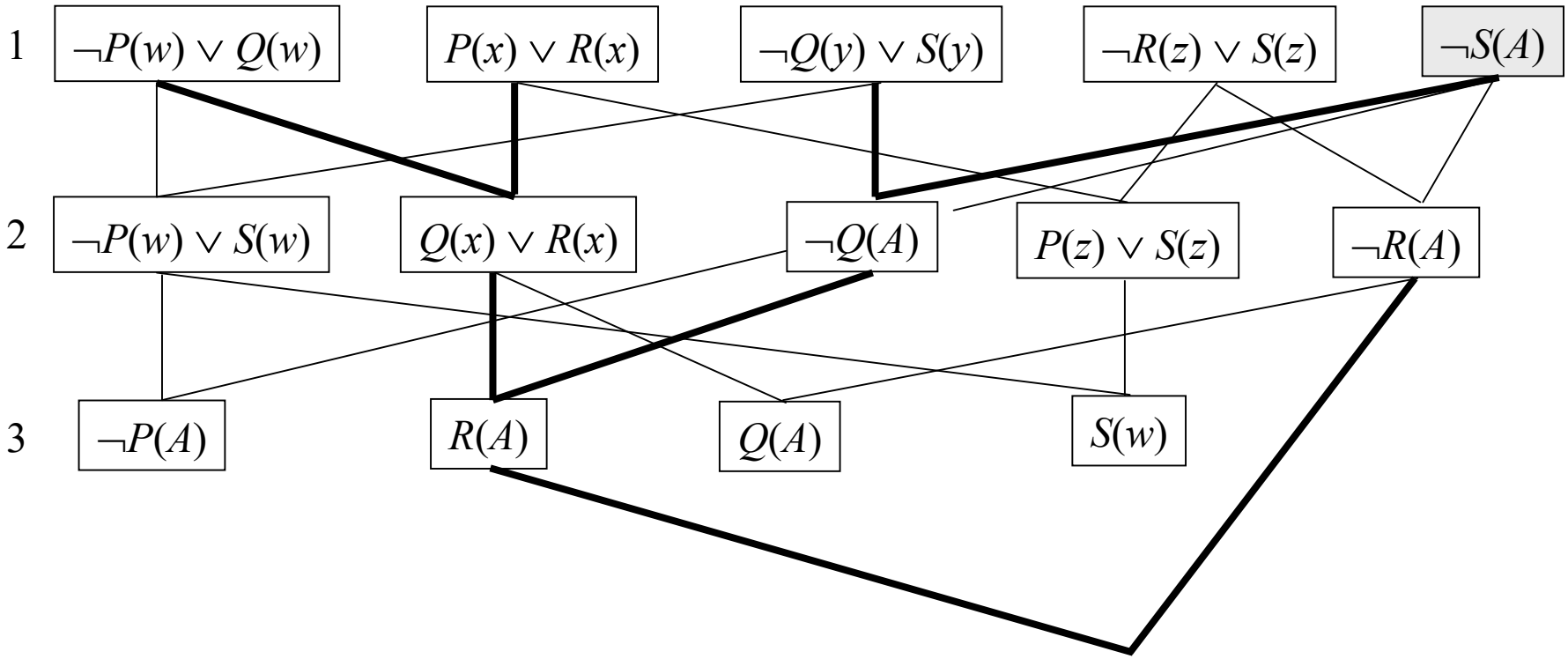
Exemplo

Sentença	FNC
$\forall x\ P(x) \Rightarrow Q(x)$	$\neg P(w) \vee Q(w)$
$\forall x\ \neg P(x) \Rightarrow R(x)$	$P(x) \vee R(x)$
$\forall x\ Q(x) \Rightarrow S(x)$	$\neg Q(y) \vee S(y)$
$\forall x\ R(x) \Rightarrow S(x)$	$\neg R(z) \vee S(z)$

Provar: $S(A)$

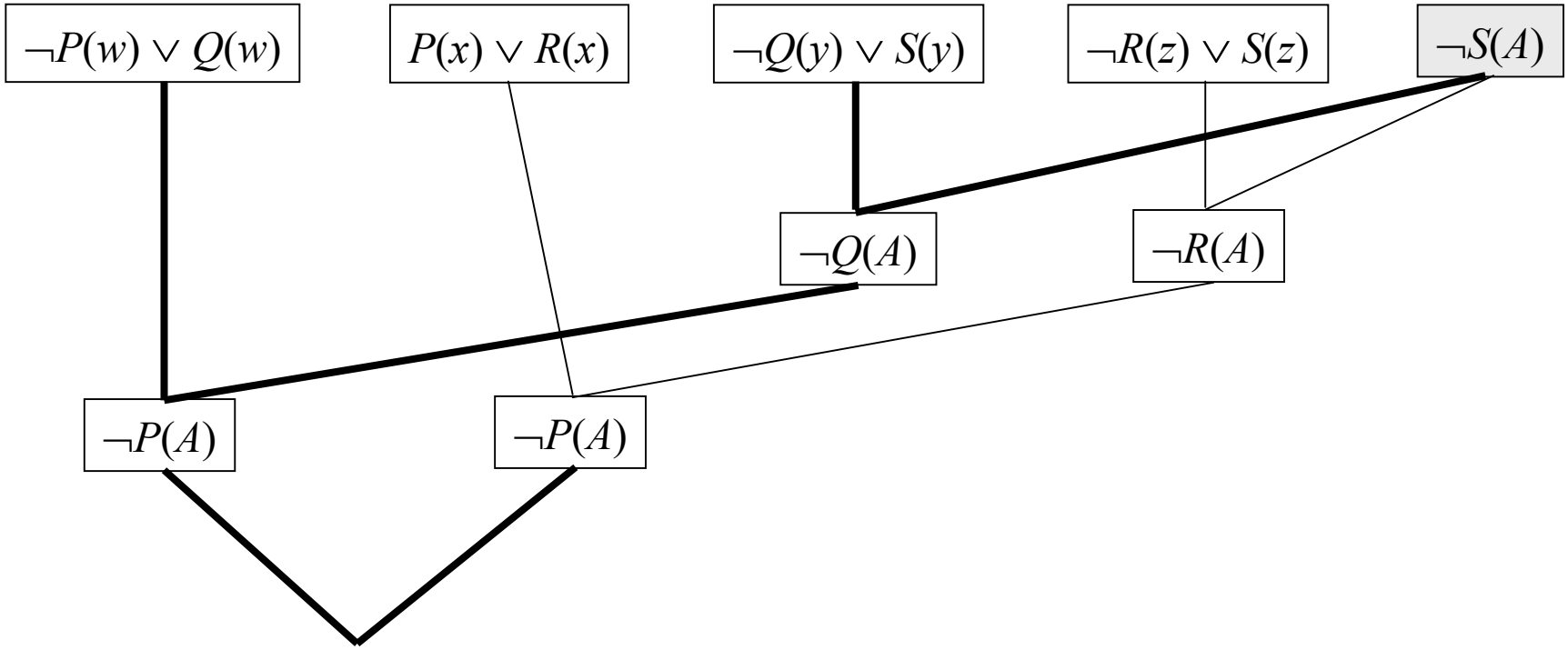
Conjunto base = {cláusulas} $\cup$ {negação da cláusula que se quer provar}

Busca em amplitude



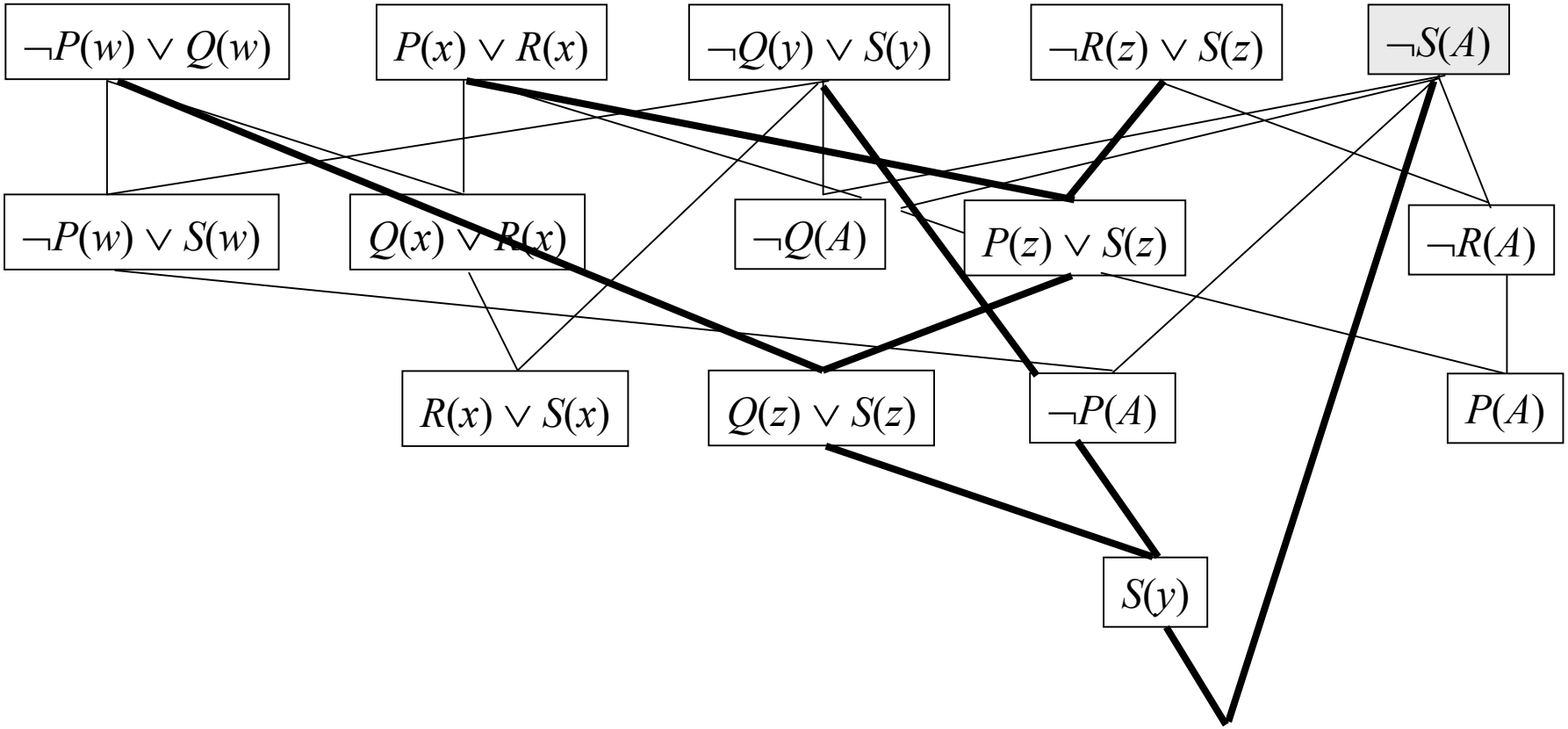
Completo

Conjunto suporte



Completo se explorarmos (e.g. *breadth-first*) todos conjuntos suporte

Linear



Não é completo

Completude da resolução

Qualquer conjunto de sentenças *S* de primeira ordem é representável na forma clausal

Assumir que *S* é insatisfatível, *S* na forma clausal

Teorema de Herbrand

Existe um conjunto *S'* de instâncias básicas de *S* que é insatisfatível

Teorema da instanciação

Resolução encontra uma contradição em *S'*

Lema da elevação

Existe uma prova por resolução para a contradição em *S'*

Resultados importantes da LPO

1 – Correção: para toda linguagem de primeira ordem L, para todo conjunto de sentenças KB de L, para toda sentença Q de L:

se $KB \vdash Q$ então $KB \models Q$

2 – Completude: para toda linguagem de primeira ordem L, para todo conjunto de sentenças KB de L, para toda sentença Q de L:

se $KB \models Q$ então $KB \vdash Q$

3 – Compacidade: um conjunto KB de sentenças é satisfatível se e somente se *todo* subconjunto finito de KB for satisfatível.

4 – Seja A um alfabeto de primeira ordem, Q uma sentença de $L(A)$ e KB um conjunto de fórmulas de $L(A)$:

a) $\models Q$ se e somente se $\neg Q$ é insatisfatível

b) $KB \models Q$ se e somente se $KB \cup \{\neg Q\}$ é insatisfatível

5 – O problema da implicação lógica para linguagens de primeira ordem é parcialmente decidível.

6 – Teorema de Church

o problema da validade para linguagens de primeira ordem é indecidível

Observação

Este material refere-se às notas de aula do curso EA 072 Inteligência Artificial em Aplicações Industriais da Faculdade de Engenharia Elétrica e de Computação da Unicamp. Não substitui o livro texto, as referências recomendadas e nem as aulas expositivas. Este material não pode ser reproduzido sem autorização prévia dos autores. Quando autorizado, seu uso é exclusivo para atividades de ensino e pesquisa em instituições sem fins lucrativos.