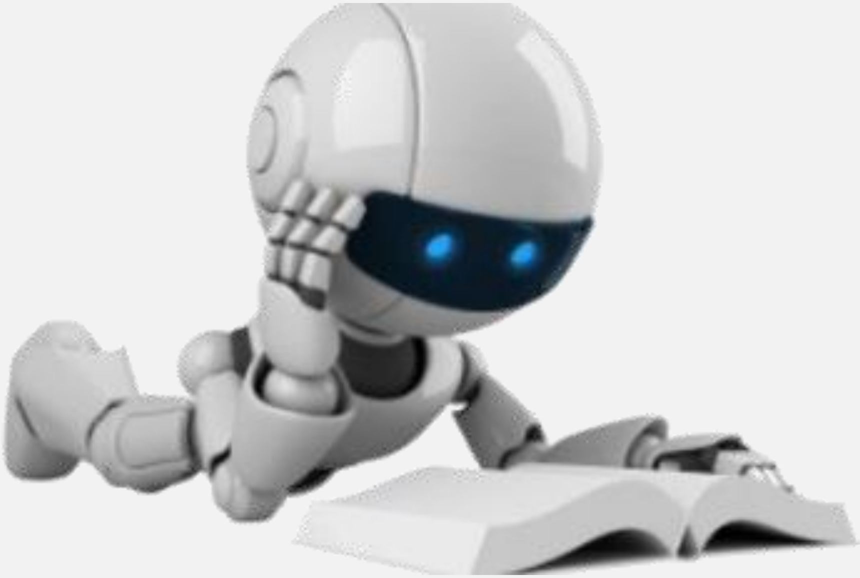
Agentes Lógicos
(Part II)



Sumário

- Agentes baseados em conhecimento ✓
- O mundo do Wumpus ✓
- Lógica em geral ✓
- Lógica proposicional (Booleana) ✓
  - Equivalência, validade, satisfação
- Lógica de 1ª ordem
  - Representação em lógica de 1ª ordem
  - Inferência em lógica de 1ª ordem

TÉCNICO LISBOA

Sumário

- Necessidade da Lógica de Primeira Ordem (LPO)
- Sintaxe e Semântica da LPO
- Uso da LPO
- Mundo do Wumpus em LPO
- Engenharia do Conhecimento em LPO

Relação com o livro

- Capítulo 8 (8.1, 8.2., 8.3, 8.4.1)
- Capítulo 9 (9.1, 9.2, 9.3.1, 9.3.2, 9.4.1)

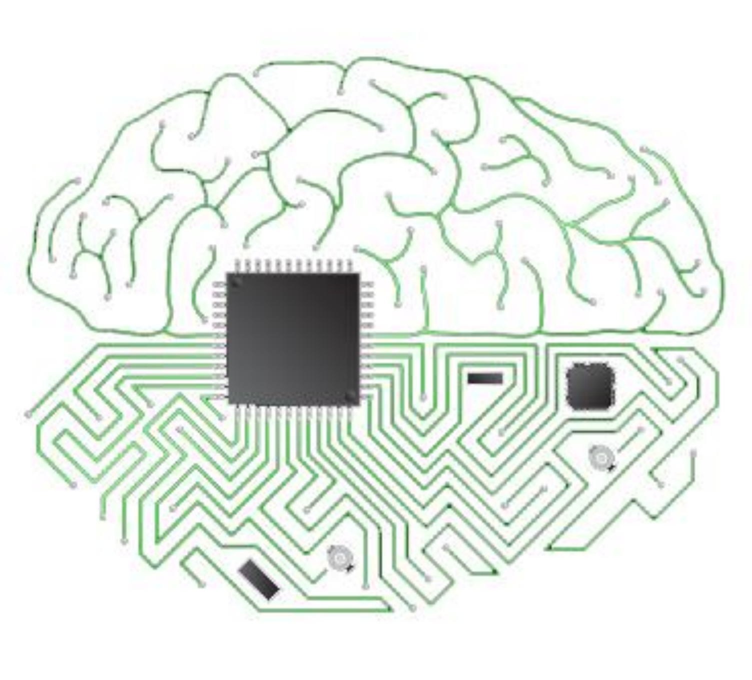
Outras secções assume-se que os alunos já deram na cadeira de Lógica para a Programação.

As áreas de IA

Robótica

Representação do Conhecimento e Raciocínio

Visão



Procura

Agentes

Planeamento de acções

Língua Natural

Aprendizagem

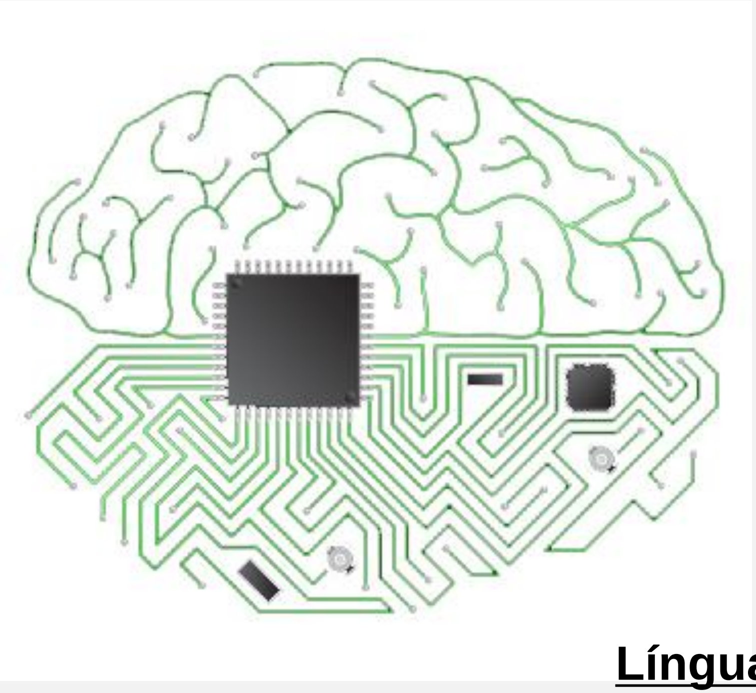
Jogos

As áreas de IA

Representação do Conhecimento e Raciocínio

Robótica

Visão



Procura

Agentes

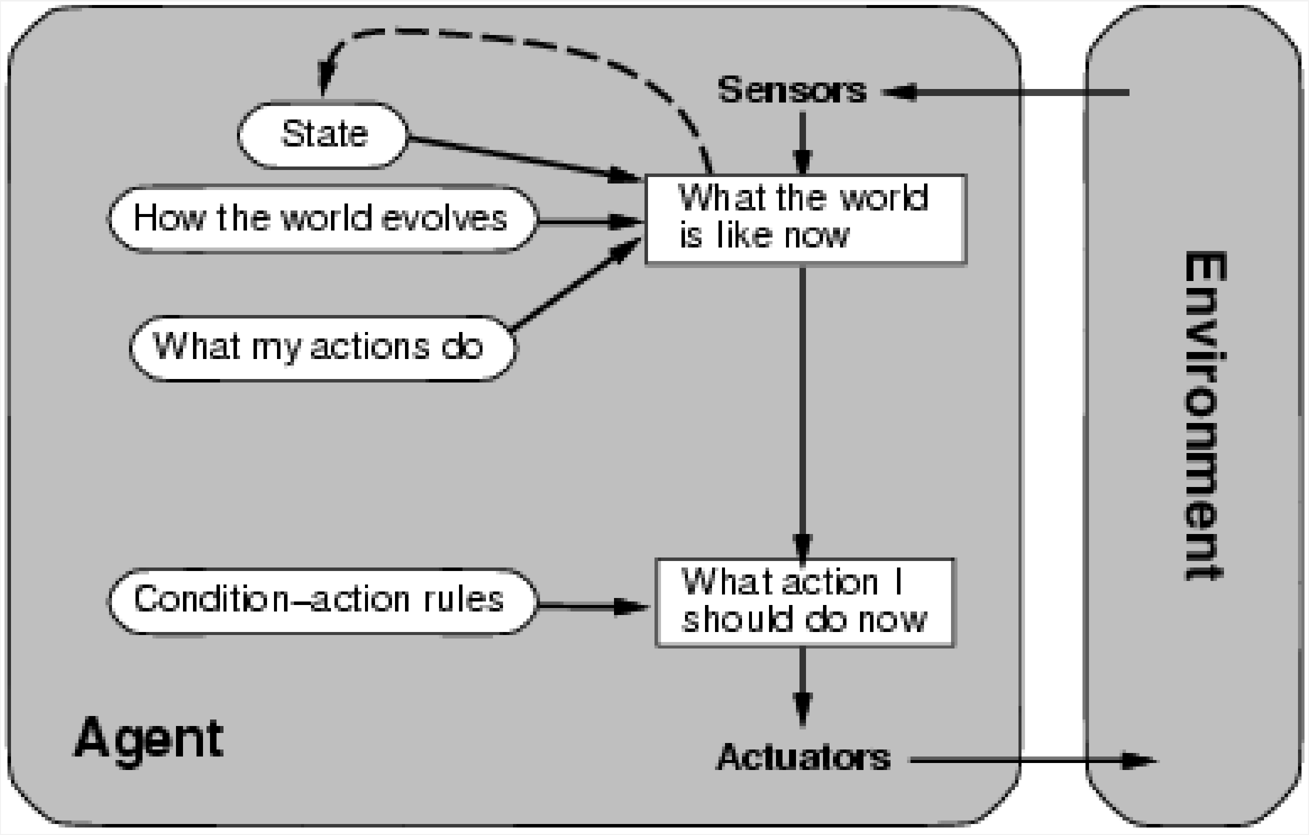
Planeamento de acções

Língua Natural

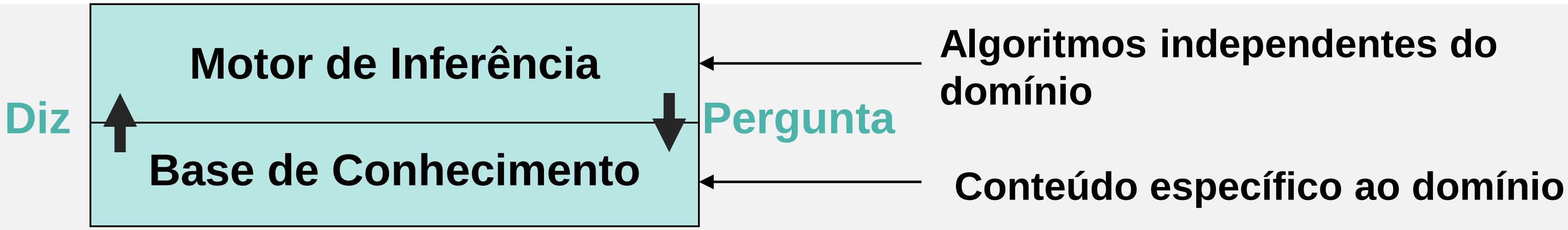
Aprendizagem

Jogos

Agentes baseados no conhecimento



Bases de conhecimento



- Base de Conhecimento (BC ou KB, do Inglês *Knowledge Base*) = conjunto de frases numa linguagem formal

Agente baseado em conhecimento

Função AgenteBC (*percepcao)* **devolve** *accao*
estático: *BC*, uma base de conhecimento
t, um contador, inicialmente a 0, que indica o tempo

`Diz`(*BC*,cria-precepcao-frase*(percepcao,t)*)
accao ← `Pergunta`(*BC*,cria-accao-pergunta*(t)*)
`Diz`(BC,cria-accao-frase*(accao,t)*)
t ← *t* +1
devolve *accao*

- O agente deve ser capaz de:
  - Representar estados, acções, etc.
  - Incorporar novas percepções
  - Actualizar representação interna do mundo
  - Deduzir propriedades implícitas no mundo
  - Deduzir acções mais apropriadas

Prós e Contras da Lógica Proposicional

- ☺ Lógica proposicional é declarativa
- ☺ Lógica proposicional permite informação parcial / disjuntiva / negada
  - Ao contrário de muitas estruturas de dados e bases de dados
- ☺ Lógica proposicional é composta
  - Significado de $P \wedge Q$ é derivado do significado de P e de Q
- ☺ Significado em lógica proposicional é independente do contexto
  - Ao contrário da linguagem natural, onde o significado depende do contexto
- ☹ Lógica proposicional tem poder de expressividade limitado
  - Ao contrário da linguagem natural
  - E.g., não se pode dizer ”*todas as pessoas são simpáticas*“
    - Excepto se escrevermos uma frase para cada pessoa

Lógica de Primeira Ordem

- Enquanto que a lógica proposicional assume que o mundo contém factos
- A lógica de primeira ordem (tal como a linguagem natural) assume que o mundo contém:
  – Objectos: pessoas, casas, números, cores, jogos de baseball, guerras, …
  – Relações: vermelho, redondo, par, irmão de, maior do que, parte de, está entre, …
  – Funções: pai de, melhor amigo, incremento, soma, …

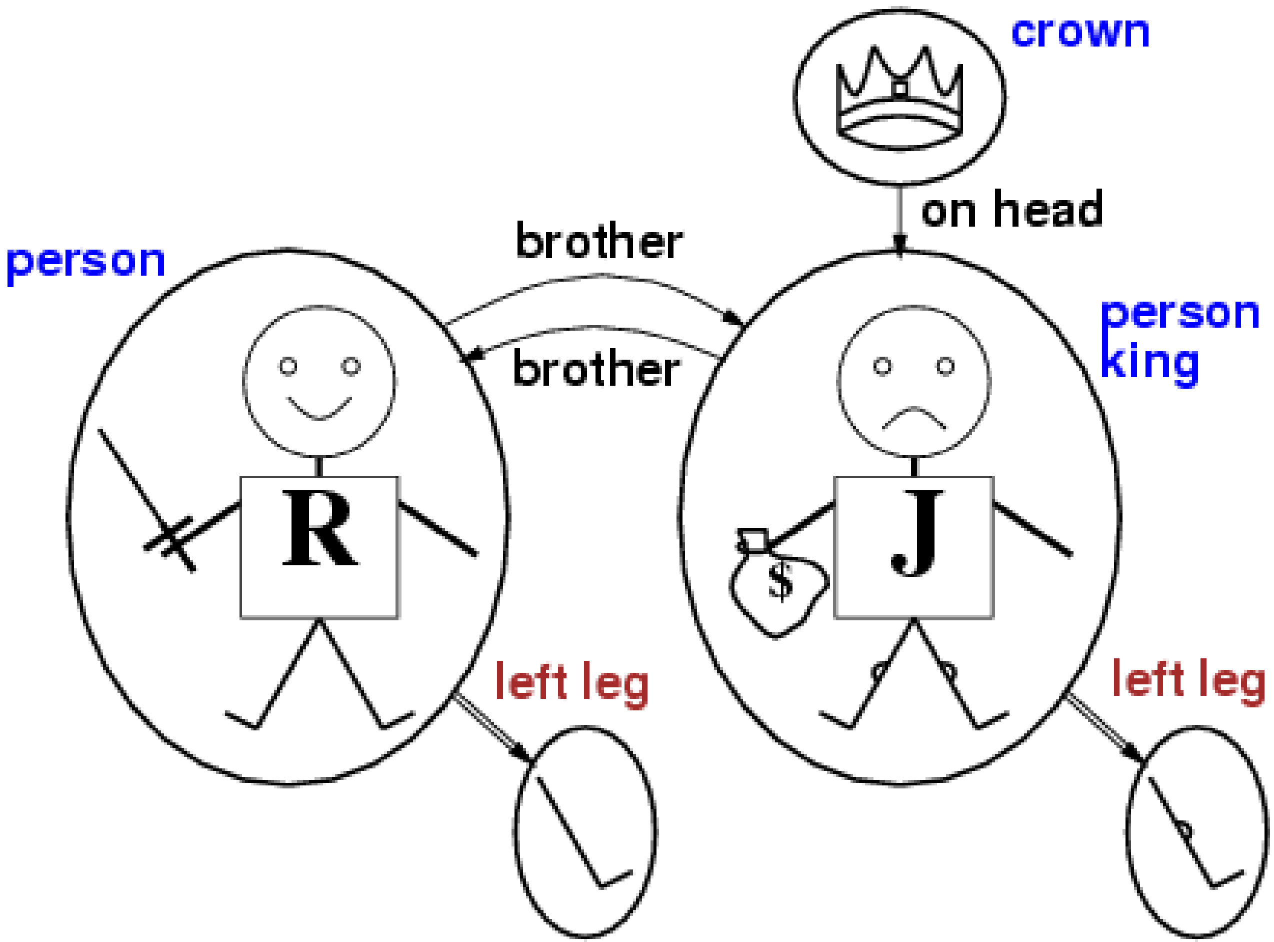
Sintaxe LPO: elementos básicos

- Constantes ReiJoao, 2, ...
- Predicados Irmaos, >,...
- Funções Raiz, PernaEsquerdaDe,...
- Variáveis x, y, a, b,...
- Conectivas $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Igualdade $=$
- Quantificadores $\forall, \exists$

Modelos para LPO

- Modelos para LPO contêm objectos.
- Os objectos num modelo podem estar relacionados...

Modelos para LPO: Exemplo



Modelos são mundos possíveis: modelo contendo 5 objectos, duas relações binárias, três relações unárias e uma função unária (left leg).

Modelos para LPO: Exemplo

- Constantes
  - RicardoCoracaoLeao, ReiJoao, PernaEsqDeRicardoCoracaoLeao, PernaEsqDeReiJoao, Coroa
- Predicados

  Aridade=2
  - Irmãos: (RicardoCoracaoLeao, ReiJoao), (ReiJoao, RicardoCoracaoLeao)
  - NaCabeca: (Coroa, ReiJoao)

  Aridade=1 (propriedades)
  - Pessoa: (RicardoCoracaoLeao),(ReiJoao)
  - Rei: (ReiJoao)
  - ECoroa: (Coroa)
- Funções
  - PernaEsqDe: (RicardoCoracaoLeao,PernaEsqDeRicardoCoracaoLeao), (ReiJoao,PernaEsqDeReiJoao), (PernaEsqDeRicardoCoracaoLeao,INV), (PernaEsqDeReiJoao,INV), (Coroa,INV)

  INV é uma perna “invisível”!

  Funções em LPO são totais, i.e. estão definidas para todos os objectos

Frases Atómicas

Termo: expressão lógica que referência um objecto

Termo $\rightarrow$ *Constante* | *Variável* | *Função(Termo,…)*

FraseAtómica $\rightarrow$

Predicado(Termo,...) | *Termo = Termo*

E.g.

- *PernaEsqDe(ReiJoao)*
- *Irmãos(ReiJoao,RicardoCoracaoLeao)*
- *>(Comprimento(PernaEsqDe(RicardoCoracaoLeao)), Comprimento(PernaEsqDe(ReiJoao)))*
- *Pai(ReiJoao) = Henrique*

Frases Complexas

FraseComplexa $\rightarrow$

FraseAtómica |

(FraseComplexa Conectiva FraseComplexa) |

Quantificador Variável,… FraseComplexa |

$\neg$ FraseComplexa

Conectiva $\rightarrow \wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow$

Quantificador $\rightarrow \forall \mid \exists$

Frases Complexas

Exemplos

- *Irmãos(ReiJoao,RicardoCoracaoLeao)* $\Rightarrow$ *Irmãos(RicardoCoracaoLeao,ReiJoao)*
- $\neg$*Irmãos(PernaEsqDe(RicardoCoracaoLeao), ReiJoao)*
- $\forall$*x,y Irmãos(x,y)* $\Rightarrow$ *Irmãos(y,x)*

Verdade em LPO

- Frases são verdadeiras em relação a um modelo/conceptualização e uma interpretação

- Modelo contém objectos (elementos do domínio) e relações entre eles

- Interpretação especifica referências para

Símbolos de constante	→	objectos
Símbolos de predicado	→	relações
Símbolos de função	→	relações funcionais

- Uma frase atómica com a forma *predicado(termo*$_1$*,...,termo*$_n$*)* é verdadeira *sse* os objectos referidos por *termo*$_1$*,...,termo*$_n$ pertencem à relação referida pelo *predicado*

Quantificador Universal

- $\forall$<*variáveis*> <*frase*>
- $\forall x$ *P* é verdadeiro num modelo *m* sse *P* é verdadeiro para *x* em que x são todos os objectos existentes no modelo *m*

Todos os reis são pessoas:

$\forall x\ Rei(x) \Rightarrow Pessoa(x)$

- Por outras palavras, é equivalente à conjunção de instanciações de *P*

$Rei(ReiJoao) \Rightarrow Pessoa(ReiJoao)$

$\wedge\ Rei(RicardoCoracaoLeao) \Rightarrow Pessoa(RicardoCoracaoLeao)$

$\wedge\ Rei(PernaEsqDeRicardoCoracaoLeao) \Rightarrow Pessoa(PernaEsqDeRicardoCoracaoLeao)$

$\wedge\ \ldots$

Erro comum a evitar

- Tipicamente, $\Rightarrow$ é a principal conectiva usada com $\forall$
- Erro comum: usar $\wedge$ como conectiva com $\forall$:

$\forall x$ Rei(x) $\wedge$ Pessoa(x)

significa “Todos são reis e são pessoas”

Quantificador Existencial

- $\exists$<*variáveis*> <*frase*>
- $\exists x$ *P* é verdadeiro num modelo *m* sse *P* é verdadeiro para *x* em que *x* é um objecto existente no modelo *m*

- O Rei João tem uma coroa na cabeça:
- $\exists x$ ECoroa(x) $\wedge$ NaCabeca(x,ReiJoao)

- Por outras palavras, é equivalente à disjunção das instanciações de *P*

  ECoroa(ReiJoao) $\wedge$ NaCabeca(ReiJoao,ReiJoao)

  $\vee$ ECoroa(RicardoCoracaoLeao) $\wedge$ NaCabeca(RicardoCoracaoLeao,ReiJoao)

  $\vee$ ...

  $\vee$ ECoroa(Coroa) $\wedge$ NaCabeca(Coroa,ReiJoao)

  $\vee$ ...

Outro erro comum a evitar

- Tipicamente, $\wedge$ é a principal conectiva usada com $\exists$

- Erro comum: usar $\Rightarrow$ como conectiva com $\exists$:

  $$\exists x\ \mathrm{ECoroa}(x) \Rightarrow \mathrm{NaCabeca}(x,\mathrm{ReiJoao})$$

  é verdadeiro se não existe nenhuma coroa!

Propriedades dos quantificadores

- $\forall x \; \forall y$ é o mesmo que $\forall y \; \forall x$
- $\exists x \; \exists y$ é o mesmo que $\exists y \; \exists x$

- $\exists x \; \forall y$ não é o mesmo que $\forall y \; \exists x$
- Exº para o domínio das pessoas:

  $\exists x \; \forall y$ Gosta(x,y)
  - "Existe alguém que gosta de todas as pessoas"

  $\forall y \; \exists x$ Gosta(x,y)
  - "Todas as pessoas têm alguém que gosta delas"

- Dualidade dos quantificadores: cada quantificador pode ser expresso usando o outro quantificador

  $\forall x$ Gosta(x,Gelado) $\quad\quad \neg\exists x \; \neg$Gosta(x,Gelado)

  $\exists x$ Gosta(x,Brócululos) $\quad\quad \neg\forall x \; \neg$Gosta(x,Brócululos)

TÉCNICO LISBOA

Igualdade

- $termo_1 = termo_2$ é verdadeiro para uma dada interpretação se e só se $termo_1$ e $termo_2$ se referem ao mesmo objecto
  - Pai(ReiJoao) = Henrique

- E.g. para uma frase complexa

"Ricardo Coração de Leão tem pelo menos dois irmãos"

$\exists$x,y Irmao(x,RicardoCoracaoLeao) $\wedge$
Irmao(y,RicardoCoracaoLeao) $\wedge$ $\neg$(x = y)

Uso da LPO

Domínio do Reino:

- Irmãos são parentes

  $\forall x,y\ \mathit{Irmãos}(x,y) \Rightarrow \mathit{Parentes}(x,y)$

- A mãe é o elemento feminino dos progenitores

  $\forall m,c\ \mathit{Mãe}(c) = m \Leftrightarrow (\mathit{Feminino}(m) \wedge \mathit{Progenitor}(m,c))$

- Parentesco é uma relação simétrica

  $\forall x,y\ \mathit{Parentes}(x,y) \Leftrightarrow \mathit{Parentes}(y,x)$

Exemplos

- Todos os As são Bs:

$$\forall_x A(x) \Rightarrow B(x)$$

- Nenhum A é B:

$$\neg\exists_x A(x) \wedge B(x)$$

- Alguns As são Bs:

$$\exists_x A(x) \wedge B(x)$$

- Alguns As não são Bs:

$$\exists_x A(x) \wedge \neg B(x)$$

Exemplos

- Somente os As são Bs:

$$\forall_x B(x) \Rightarrow A(x)$$

- Nem todos os As são Bs

  Alguns As não são Bs:

$$\exists_x A(x) \wedge \neg B(x)$$

- Todos os As não são Bs
  - Nenhum A é B:

$$\neg\exists_x A(x) \wedge B(x)$$

Exercícios

- Todas as pessoas gostam de outra pessoa
  - $\forall_x$ Pessoa(x) $\Rightarrow \exists_y$ Pessoa(y) $\wedge$ Gosta(x,y) $\wedge \neg$(x=y)
- Existe uma pessoa de quem todas as outras pessoas gostam
  - $\exists_x$ Pessoa(x) $\wedge \forall_y$ Pessoa(y) $\wedge \neg$(x=y) $\Rightarrow$ Gosta(y,x)
- O João frequenta a cadeira de IA ou PE (pode frequentar as duas)
  - Frequenta(João,IA) $\vee$ Frequenta(João,PE)
- O Rui frequenta ou a cadeira de IA ou a cadeira de PE (somente uma das duas)
  - Frequenta(Rui,IA) $\Leftrightarrow \neg$Frequenta(Rui,PE)

Inferência

- Inferência em lógica proposicional vs. inferência em lógica de primeira ordem
- Unificação
- Modus Ponens Generalizado
- Encadeamento para a frente
- Encadeamento para trás

Instanciação Universal

- A partir de uma frase com um quantificador universal, podemos inferir uma frase que resulta da substituição da variável por um termo sem variáveis

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

- E.g., $\forall x\ Rei(x) \wedge Ambicioso(x) \Rightarrow Malvado(x)$ permite inferir:

  $Rei(João) \wedge Ambicioso(João) \Rightarrow Malvado(João)$

  $Rei(Ricardo) \wedge Ambicioso(Ricardo) \Rightarrow Malvado(Ricardo)$

  $Rei(Pai(João)) \wedge Ambicioso(Pai(João)) \Rightarrow Malvado(Pai(João))$

  .

  .

  .

Instanciação Existencial

- Para uma frase α, uma variável *v*, e uma constante *k* que não aparece em nenhuma frase da base de conhecimento

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

- E.g., A partir de $\exists x\ ECoroa(x) \wedge NaCabeca(x, João)$ podemos inferir $ECoroa(C_1) \wedge NaCabeca(C_1, João)$

  desde que C_1 seja um símbolo de constante novo, chamado constante de Skolem

- A frase que contém o quantificador existencial pode ser eliminada → equivalência por inferência

Unificação

$$\forall x\ Rei(x) \wedge Ambicioso(x) \Rightarrow Malvado(x)$$

- Se conseguimos encontrar uma substituição θ para x para a qual se verifica *Rei(x) e Ambicioso(x)* então podemos inferir *Malvado(x)*
- Genericamente: se conseguirmos encontrar uma substituição θ que converta a premissa de uma implicação numa frase já existente na BC, então podemos derivar a conclusão da implicação após efectuada a substituição θ

Exº

- $\forall x$ Rei(x) $\wedge$ Ambicioso(x) $\Rightarrow$ Malvado(x)
- Rei(João)
- $\forall y$ Ambicioso(y)

θ = {x/João,y/João} permite inferir Malvado(João)

- Unificação = identificação de uma substituição que permita que duas frases sejam logicamente equivalentes

Unificação: exemplo

- Unificação(α,β) = θ se αθ = βθ

α	β	θ
Conhece(João,x)	Conhece(João,Rita)	
Conhece(João,x)	Conhece(y,Isabel)	
Conhece(João,x)	Conhece(y,Mãe(y))	
Conhece(João,x)	Conhece(x,Isabel)	

Unificação: exemplo

- Unificação(α,β) = θ se αθ = βθ

α	β	θ
Conhece(João,x)	Conhece(João,Rita)	{x/Rita}
Conhece(João,x)	Conhece(y,Isabel)	
Conhece(João,x)	Conhece(y,Mãe(y))	
Conhece(João,x)	Conhece(x,Isabel)	

Unificação: exemplo

- Unificação(α,β) = θ se αθ = βθ

α	β	θ
Conhece(João,x)	Conhece(João,Rita)	{x/Rita}
Conhece(João,x)	Conhece(y,Isabel)	{x/Isabel,y/João}
Conhece(João,x)	Conhece(y,Mãe(y))	
Conhece(João,x)	Conhece(x,Isabel)	

Unificação: exemplo

- Unificação(α,β) = θ se αθ = βθ

α	β	θ
Conhece(João,x)	Conhece(João,Rita)	{x/Rita}
Conhece(João,x)	Conhece(y,Isabel)	{x/Isabel,y/João}
Conhece(João,x)	Conhece(y,Mãe(y))	{y/João,x/Mãe(João)}
Conhece(João,x)	Conhece(x,Isabel)	

Unificação: exemplo

- Unificação(α,β) = θ se αθ = βθ

α	β	θ
Conhece(João,x)	Conhece(João,Rita)	{x/Rita}
Conhece(João,x)	Conhece(y,Isabel)	{x/Isabel,y/João}
Conhece(João,x)	Conhece(y,Mãe(y))	{y/João,x/Mãe(João)}
Conhece(João,x)	Conhece(x,Isabel)	{*falha*}

Estandardização

- Conhece(João,x) e Conhece(x,Isabel) poderão ser unificados se substituirmos x por outra variável
- Esta unificação faz sentido
  - Conhece(João,x) significa que o João conhece toda a gente
  - Conhece(x,Isabel) significa que a Isabel é conhecida por toda a gente
  - Logo, o João conhece a Isabel
- Estandardização = renomeação de variáveis numa das duas frases a serem unificadas para evitar conflitos nos nomes das variáveis
- Conhece(João,x) e Conhece(y,Isabel) pode ser unificado com {x/Isabel,y/João}

Unificação: UMG

- Para unificar *Conhece(João,x)* e *Conhece(y,z)*,
  θ = {y/João, x/z } ou θ = {y/João, x/João, z/João}

- A primeira substituição é mais genérica do que a segunda.

- Existe um único unificador mais geral (UMG): efectua o menor número de substituições para unificar dois termos
  UMG = { y/João, x/z }

Modus Ponens Generalizado (MPG)

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

com $p_i'\theta = p_i\,\theta$ para todo o *i*

p_1' é *Rei*(*João*)	p_1 é *Rei*(*x*)
p_2' é *Ambicioso*(*y*)	p_2 é *Ambicioso*(*x*)
θ é {x/João,y/João}	q é *Malvado*(*x*)
q θ é *Malvado*(*João*)	

- MPG usado com BC com cláusulas que têm exactamente um literal positivo: $(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)$ equivale a $(\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee q)$

- Todas as variáveis estão quantificadas universalmente

Exemplo: base de conhecimento

- Do ponto de vista legal, um Americano é um criminoso por vender armas a nações hostis. O país Nono, um inimigo da América, possui alguns mísseis, e todos estes mísseis foram-lhe vendidos pelo Coronel West, que é Americano.

- Objectivo: provar que o Coronel West é um criminoso.

Exemplo: base de conhecimento (cont.)

... um Americano é um criminoso por vender armas a nações hostis:

Americano(x) $\wedge$ Arma(y) $\wedge$ Vende(x,y,z) $\wedge$ Hostil(z) $\Rightarrow$ Criminoso(x)

Nono … possui alguns mísseis, i.e., $\exists$x Possui(Nono,x) $\wedge$ Míssil(x):

Possui(Nono,M_1) e Missil(M_1) *[instanciação existencial]*

… todos os mísseis foram-lhe vendidos pelo Coronel West

Missil(x) $\wedge$ Possui(Nono,x) $\Rightarrow$ Vende(West,x,Nono)

Mísseis são armas:

Missil(x) $\Rightarrow$ Arma(x)

Um inimigo da América é considerado “hostil”:

Inimigo(x,America) $\Rightarrow$ Hostil(x)

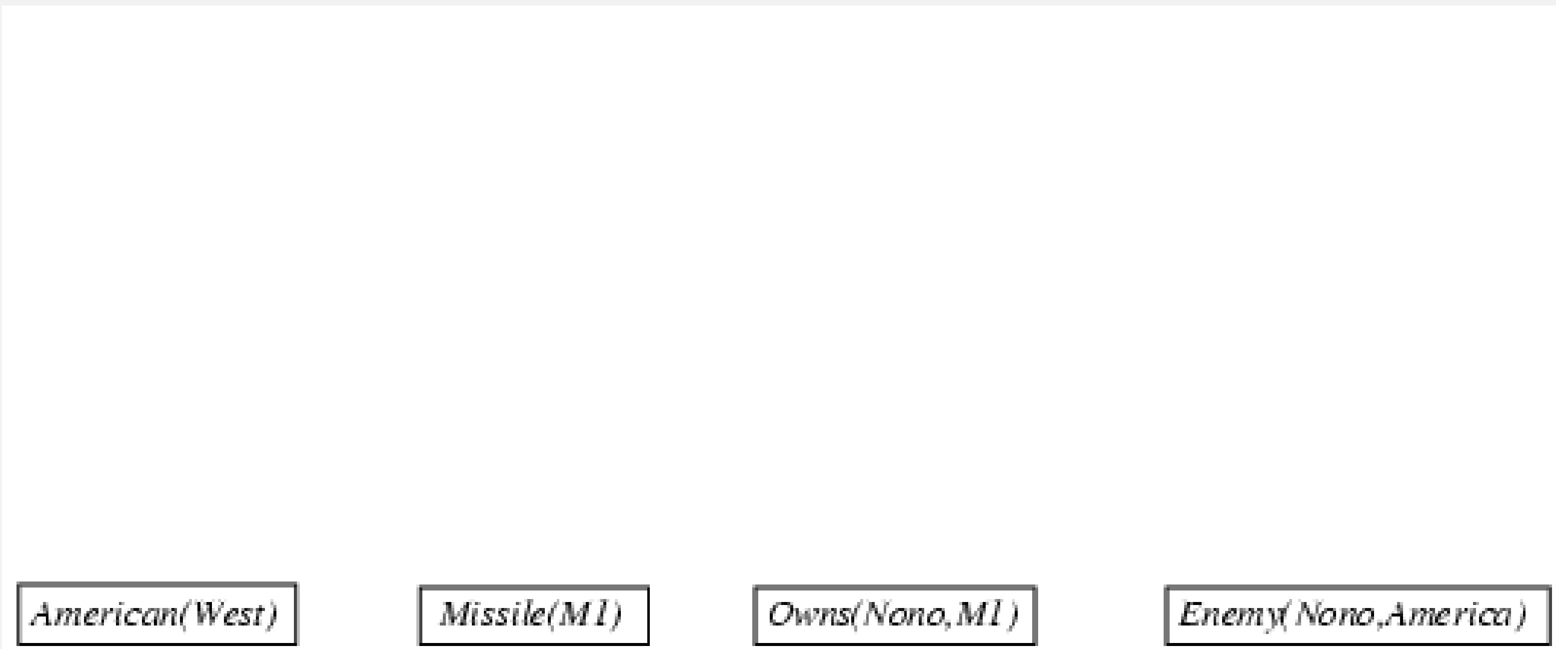
West é Americano …

Americano(West)

O país Nono é um inimigo da América …

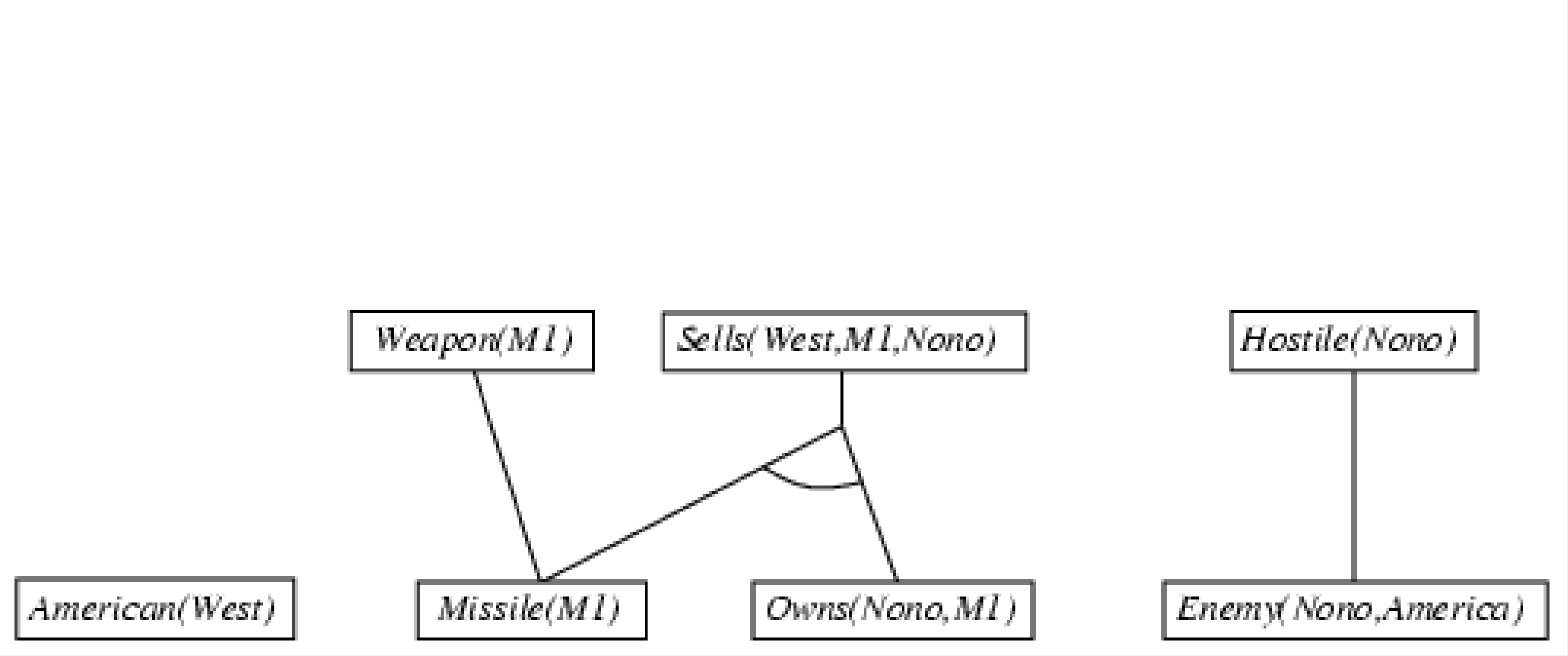
Inimigo(Nono,America)

Encadeamento progressivo: prova

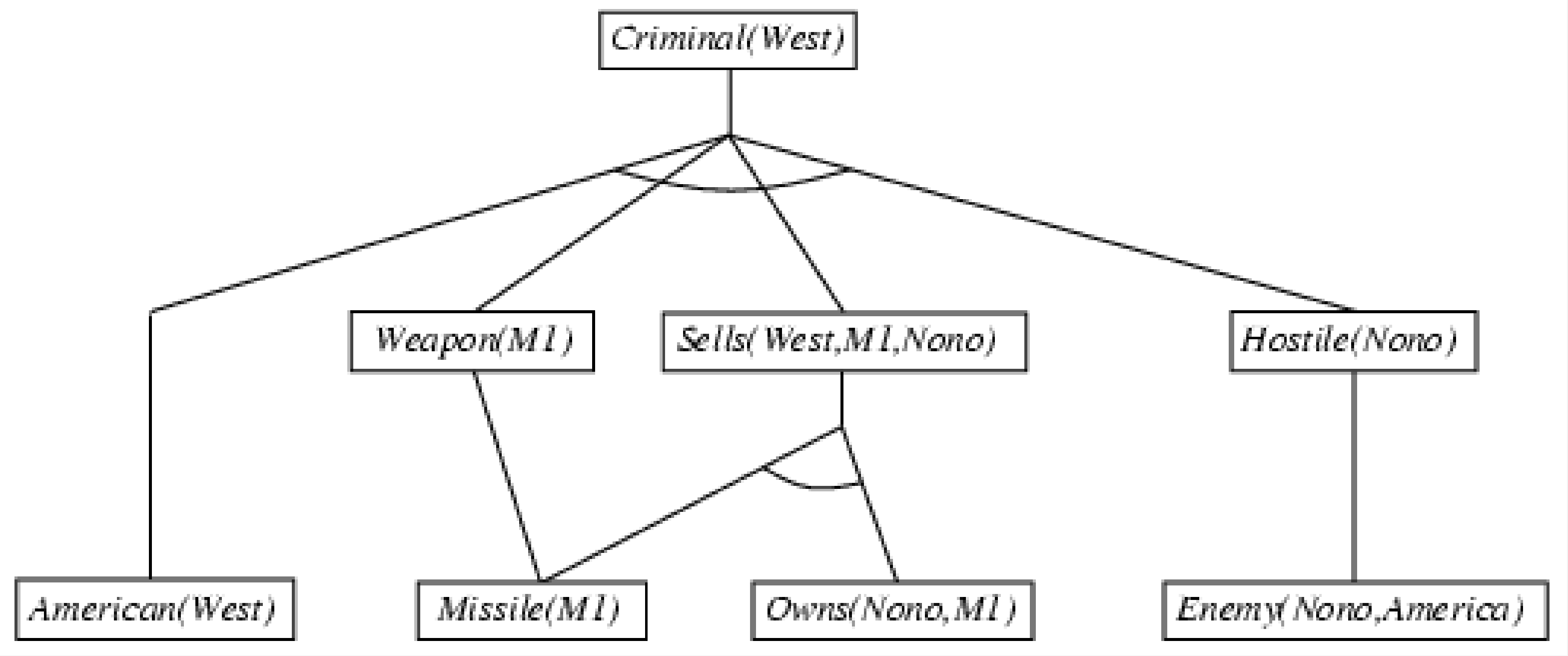


Inicialmente: frases que não têm variáveis

Encadeamento progressivo: prova



Encadeamento progressivo: prova



Propriedades do encadeamento progressivo

- Sólido e completo para cláusulas na forma $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$ em lógica de primeira ordem

- Datalog = cláusulas na forma $(p_1 \wedge \ldots \wedge p_n \Rightarrow q)$ em lógica de primeira ordem + não há funções
- EP termina para Datalog num número finito de iterações

- Não termina se α não é consequência lógica

- Não podemos resolver este problema: encadeamento progressivo é semi-decidível

Eficiência do encadeamento progressivo

Encadeamento progressivo incremental: só é necessário fazer um emparelhamento de uma frase na iteração *k* se uma premissa tiver sido adicionada na iteração *k-1*

$\Rightarrow$ Emparelhar cada frase cuja premissa contém um novo literal positivo

Emparelhamento pode ser dispendioso:

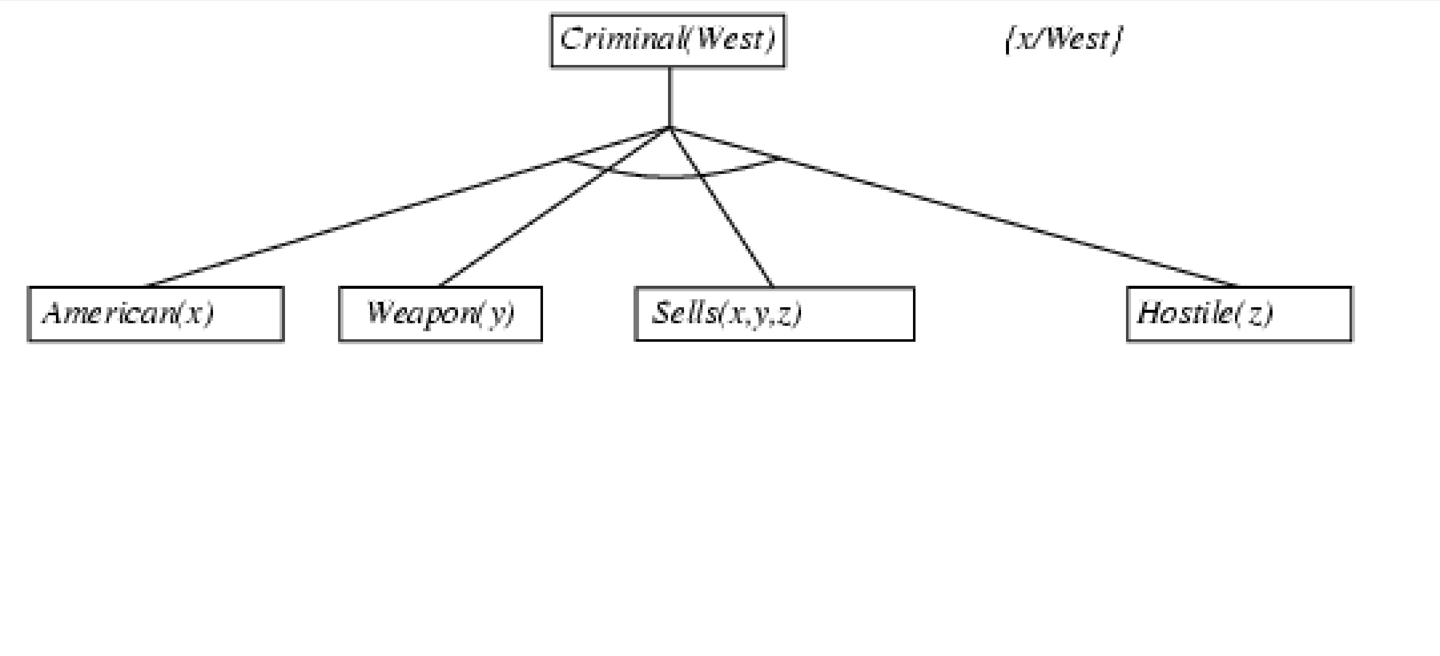
Bases de dados indexadas permitem encontrar factos conhecidos em tempo constante (O(1))

– e.g., pergunta *Missil(x)* responde $Missil(M_1)$

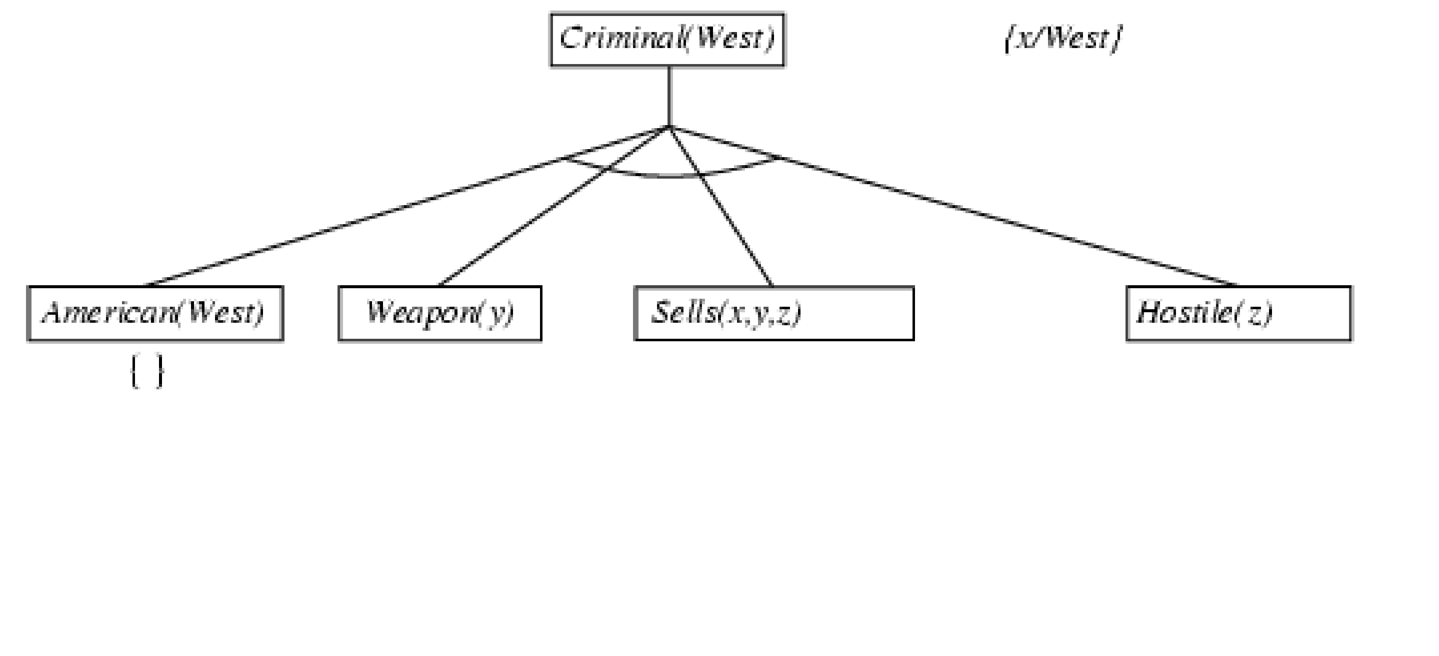
Encadeamento progressivo é muito usado em bases de dados dedutivas

Encadeamento regressivo: exemplo

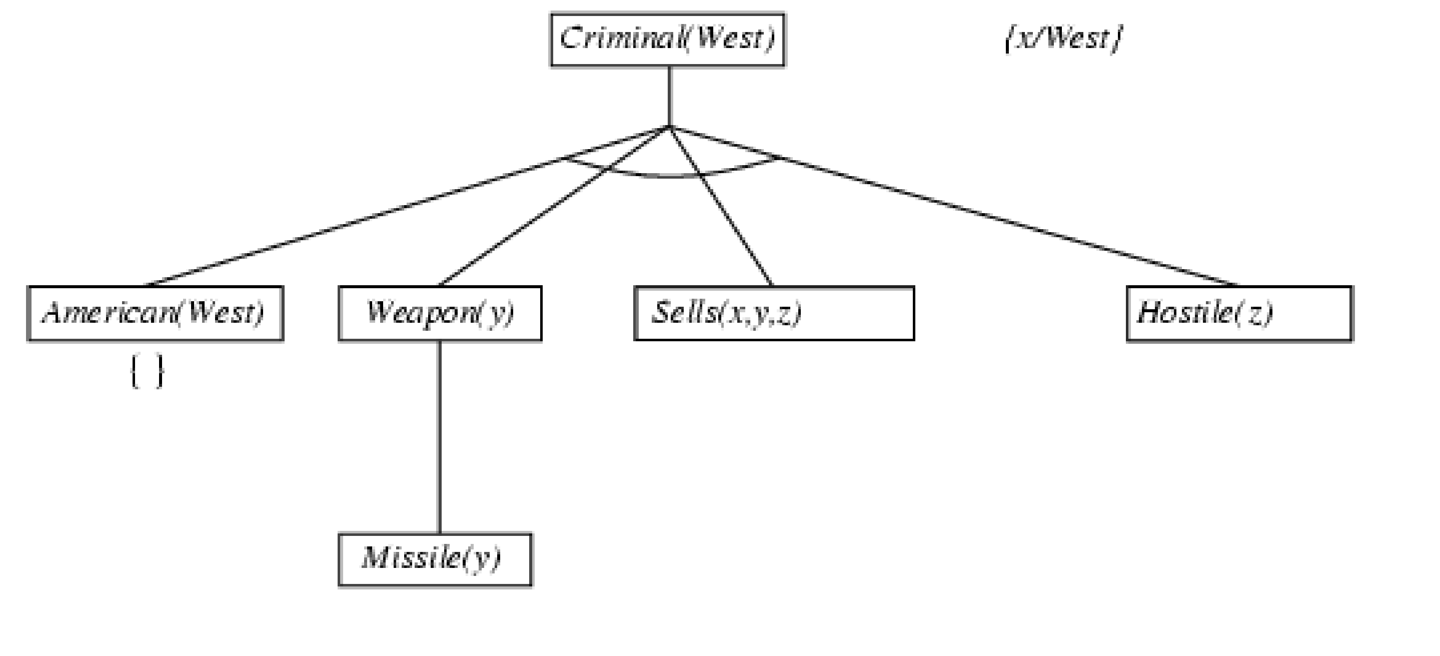
Encadeamento regressivo: exemplo



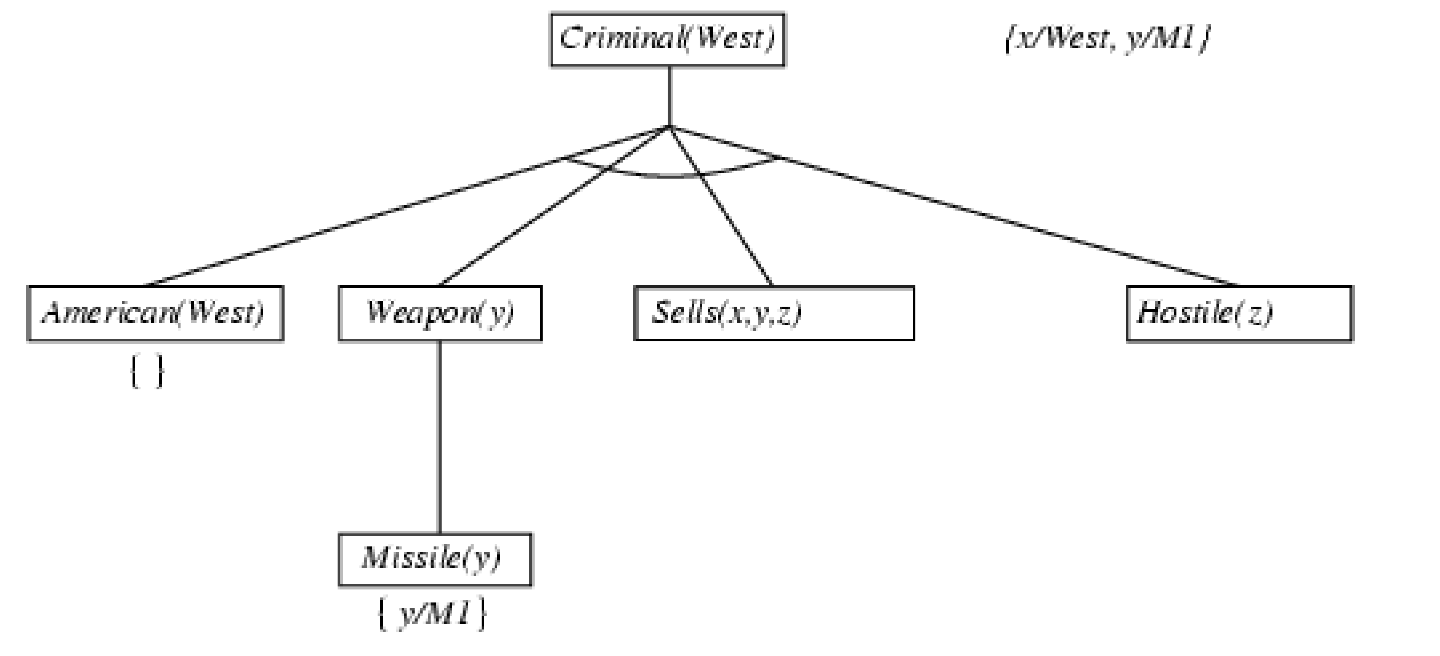
Encadeamento regressivo: exemplo



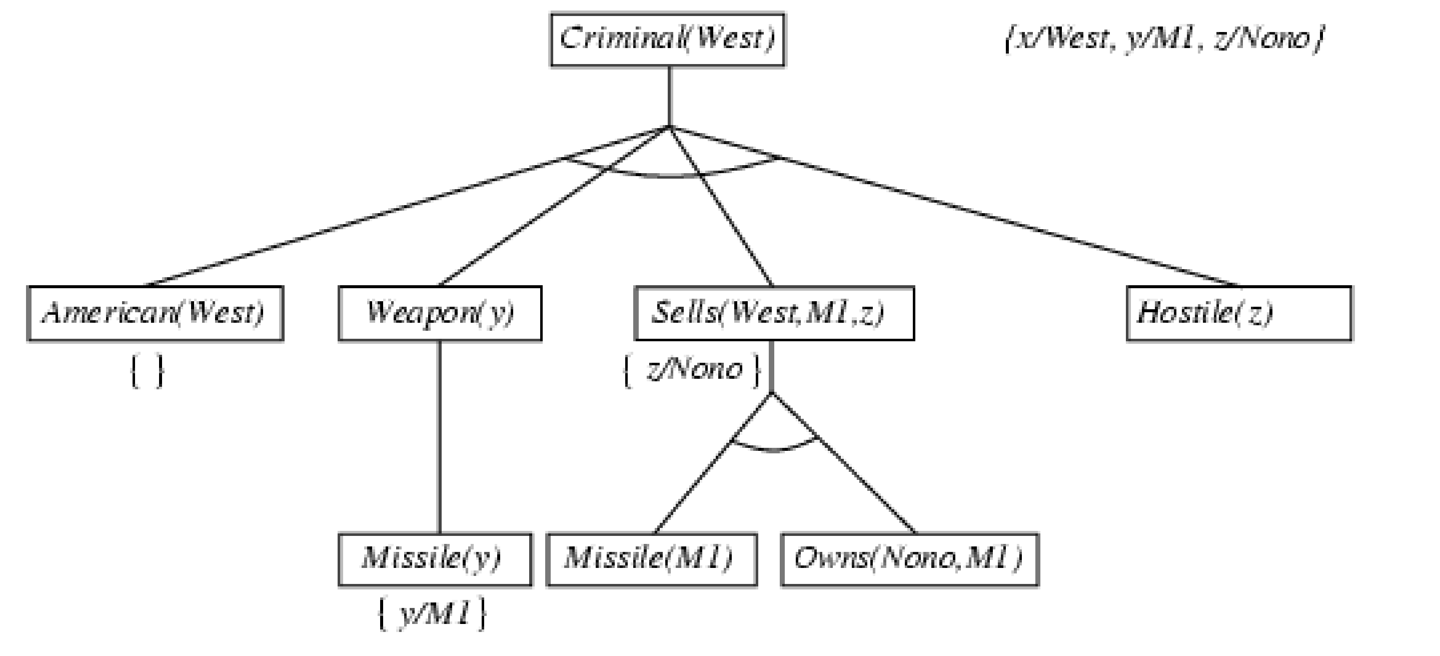
Encadeamento regressivo: exemplo



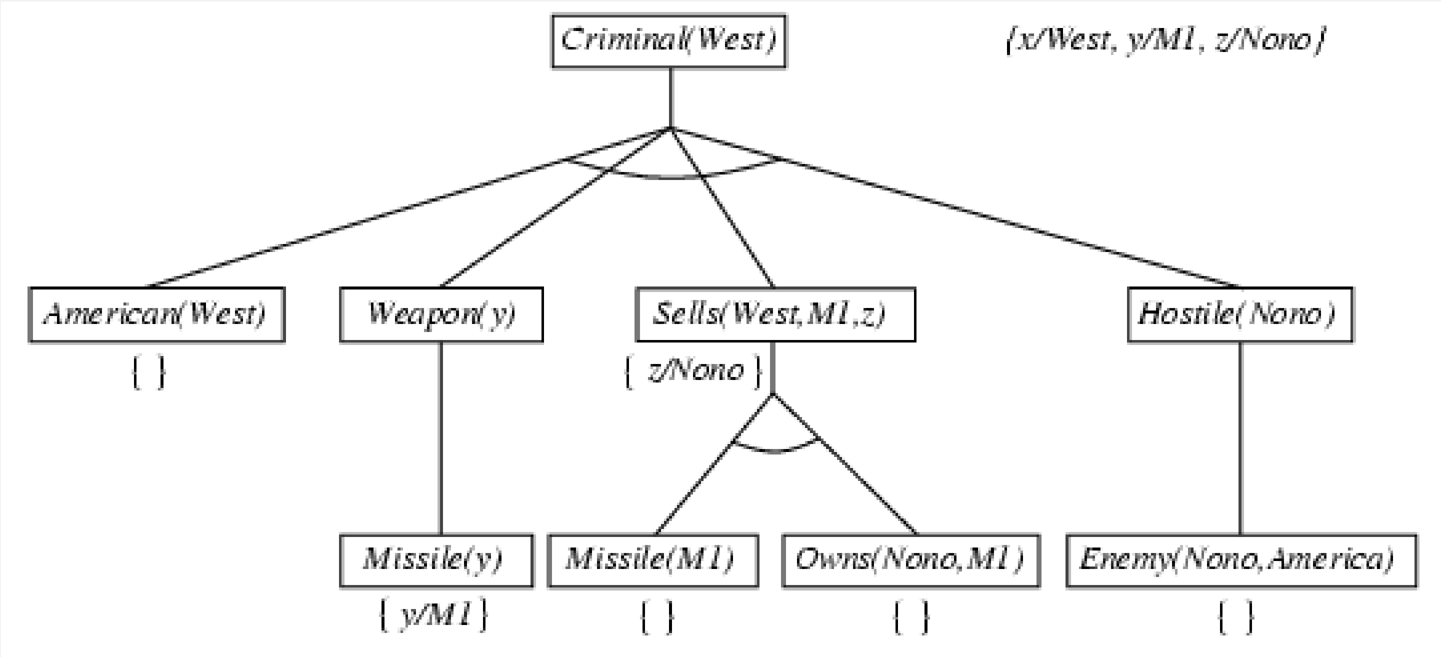
Encadeamento regressivo: exemplo



Encadeamento regressivo: exemplo



Encadeamento regressivo: exemplo



Propriedades do encadeamento regressivo

- Prova com procura em profundidade com recursão: espaço é linear no tamanho da prova
- Incompletude devido a ciclos infinitos
  - Podem ser evitados comparando o objectivo actual com os objectivos na pilha
- Ineficiente devido à existência de sub-objectivos repetidos (tanto com sucesso como falha)
  - Guardar em memória resultados obtidos anteriormente $\Rightarrow$ memória adicional
- Muito usado em programação em lógica

Sumário

- Uso de lógica de 1ª ordem para representação
- Emparelhamento para a frente usado principalmente em bases de dados dedutivas
- Emparelhamento regressivo usado em sistemas com programação em lógica.