

LÓGICA DE PRIMEIRA ORDEM CAP 8

Parcialmente adaptado de
<http://aima.eecs.berkeley.edu>

Resumo

- Motivação para a Lógica de Primeira Ordem
- Sintaxe e semântica da LPO
- Representação de conhecimento em LPO
- Exemplos de representação em LPO

Prós e Contras da Lógica Proposicional

- 😊 A lógica proposicional é **declarativa**: a sintaxe permite expressar factos
- 😊 A lógica proposicional permite representar informação parcial, disjuntiva e negativa
 - contrariamente à maioria das estruturas de dados e bases de dados
- 😊 A lógica proposicional é **composicional**
 - o significado de $B_{1,1} \wedge P_{1,2}$ é obtido compondo o significado de $B_{1,1}$ e de $P_{1,2}$
- 😊 O significado da lógica proposicional é **independente do contexto**
 - contrariamente à lingua natural, em que o significado depende do contexto
- 😞 A lógica proposicional tem um poder expressivo muito limitado
 - contrariamente à lingua natural.
 - E.g., não se consegue dizer “buracos provocam brisa em casas adjacentes” a não ser que se escreva uma proposição para cada casa do mundo

Lógica de Primeira Ordem

- Enquanto que a lógica proposicional assume que o mundo contém factos,
- A Lógica de Primeira Ordem (tal como a lingua natural) assume que o mundo pode conter:
 - **Objectos**: pessoas, casas, números, cores, jogos de futebol, guerras,...
 - **Relações**: vermelho, redondo, errado, primo, irmão de, maior do que, dentro de, parte de, tem cor, ocorreu após, tem, vende,...
 - **Funções**: pai de, melhor amigo, um a mais do que, princípio de,...

Lógica de Primeira Ordem

- A Lógica de Primeira Ordem (LPO) é também conhecida através de outras designações:
 - Cálculo de Predicados de Primeira Ordem (first-order predicate calculus)
 - Cálculo de Predicados de Ordem Inferior (lower predicate calculus)
 - Lógica de Predicados (predicate logic)
 - Linguagem de Lógica de Primeira Ordem (language of first-order logic)

Lógicas em Geral

- **Compromisso Ontológico:** O que existe no mundo – **Verdade**
- **Compromisso Epistemológico:** Aquilo em que um agente acredita sobre factos – **Crença**

Linguagem	Compromisso Ontológico	Compromisso Epistemológico
Lógica Proposicional	Factos	Verdadeiro/Falso/Desconhecido
Lógica de Primeira Ordem	Factos, Objectos, Relações	Verdadeiro/Falso/Desconhecido
Lógica Temporal	Factos, Objectos, Relações, Tempo	Verdadeiro/Falso/Desconhecido
Teoria da Probabilidade	Factos	Grau de Crença $\in [0,1]$
Lógica Vaga/Difusa	Graus de verdade $\in [0,1]$	Intervalo Conhecido de Valores

- Existem inúmeras lógicas, variando com os seus compromissos ontológicos e epistemológicos, logo com os seus domínios de aplicação:
 - Lógicas terminológicas, Lógica de primeira ordem tipificada, Lógica de segunda ordem, Lógicas de ordem superior, Lógica de primeira ordem intuicionista

Sintaxe da LPO: Elementos Básicos

O vocabulário da LPO é constituído pelos seguintes elementos:

- **Constantes:** *KingJohn, 2, UNL, Benfica, Reitor ...*
 - **Predicados:** *Brother, >, Irmão, Gato, ...*
 - **Funções:** *Sqrt, LeftLegOf, ...*
 - **Variáveis:** *x, y, a, b, ...*
 - **Conectivos:** $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
 - **Igualdade** $=$
 - **Quantificadores** $\forall \exists$
 - **Pontuação** $() ,$
- A lógica de primeira ordem não atribui qualquer interpretação pré-definida aos seus **símbolos não lógicos**: **constantes**, **predicados**, e **funções**

Fórmulas (ou Frases) Atômicas

- O poder expressivo adicional da lógica de primeira ordem advém da sua possibilidade de referir objectos no domínio de discurso. Sintacticamente, os termos da lógica de primeira ordem denotam esses objectos
- Um **termo** é uma **constante**, ou uma **variável**, ou $f(t_1, \dots, t_n)$ onde f é uma função e t_1, \dots, t_n são termos.
 - Exemplos: 1 , $12e40$, π , -3 , *Portugal*, *UNL*, *Diabo*, *Bem*, *Unicórnio*, *Abc*, *Xpto123*, *Key123*, *'uma cadeia de caracteres muito longa'*, $\text{Exp}(1.0)$, $\text{Exp}(\text{Mult}(I, \text{Pi}))$, $\text{Peso}(\text{Unicórnio})$, $\text{Mãe}(\text{Árbitro}(\text{Jogo}(\text{Académica}, \text{Belenenses}, 2006)))$, ...
- Uma **fórmula atômica** é $t_1 = t_2$, ou $p(t_1, \dots, t_n)$ onde p é um predicado e t_1, \dots, t_n são termos.
 - Exemplos: $\text{Brother}(\text{KingJohn}, \text{RichardTheLionheart})$, $\text{Exp}(I * \text{Pi}) + 1 = 0$, $0 + x = x$, $\text{Arco}(a1, a2)$, $> (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn})))$, $\text{Matriculado}(s123, \text{inf}, \text{ciclo1})$, ...

Fórmulas (ou Frases) complexas

- As fórmulas complexas – aka **fórmulas bem formadas (fbf)** – são contruídas recursivamente a partir das fórmulas atômicas usando as conectivas e os quantificadores, através das seguintes regras:
 - Qualquer **frase atômica** é uma fórmula bem formada (fbf)
 - Se φ é uma fórmula bem formada, então $\neg\varphi$ é uma fbf.
 - Se φ e ψ são fbfs, então $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$ e $(\varphi \Leftrightarrow \psi)$ também são fbfs.
 - Se φ é uma fbf e x é uma variável, então $\forall x \varphi$ e $\exists x \varphi$ são fbfs.
 - Nada mais é uma fbf.

Exemplos de fórmulas bem formadas

- $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
- $>(1,2) \vee \leq(1,2)$
- $>(1,2) \wedge \neg >(1,2)$
- $\forall x \forall y (x + y = y + x)$
- $\forall x \forall y \exists z (x < y) \Rightarrow (x < z \wedge z < y)$
- $\forall y \exists x Progenitor(x, y)$
- $\forall x (Humano(x) \Leftrightarrow (Mulher(x) \vee Homem(x)))$
- $\exists x (Humano(x) \wedge \neg \exists y Progenitor(x, y))$

Variáveis livres e ligadas

- x é **livre**:
 - numa fórmula atômica φ sse x ocorre em φ .
 - em $\neg\varphi$ sse x é livre em φ .
 - em $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$ e $(\varphi \Leftrightarrow \psi)$ sse x é livre em φ **ou** ψ .
 - em $\forall y \varphi$ e $\exists y \varphi$ sse $x \neq y$ e x é livre em φ .
- x é **ligada** se ocorre no âmbito de algum quantificador.
- Uma variável pode estar livre e ligada na mesma fórmula:

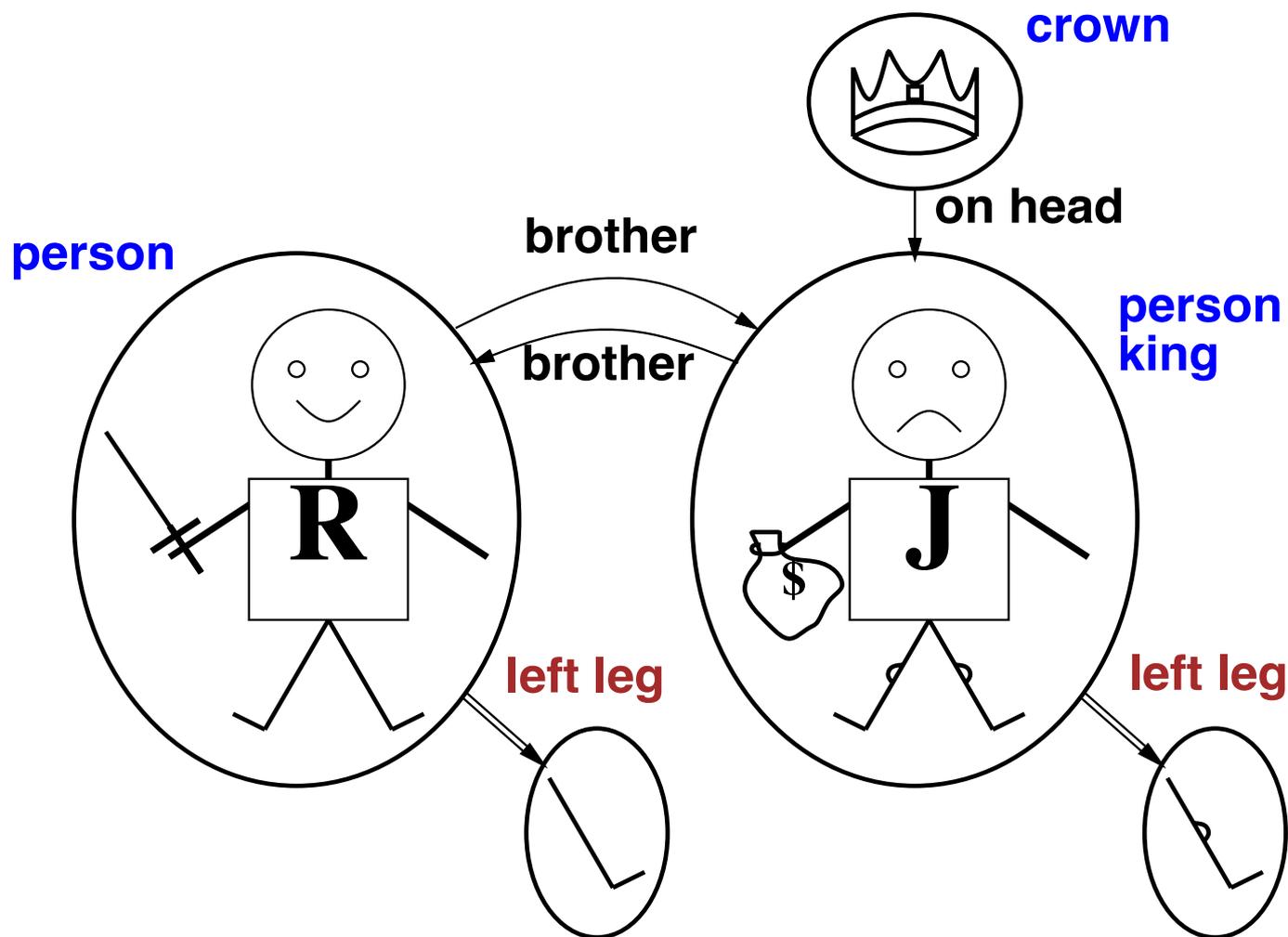
$$\left(\forall x \left(R(x, y) \Rightarrow P(x) \right) \wedge \forall y \left(\neg R(x, y) \vee \forall x P(x) \right) \right)$$

- Nota: Qualquer fórmula pode ser reescrita numa fórmula equivalente em que as variáveis livres e ligadas são disjuntas.

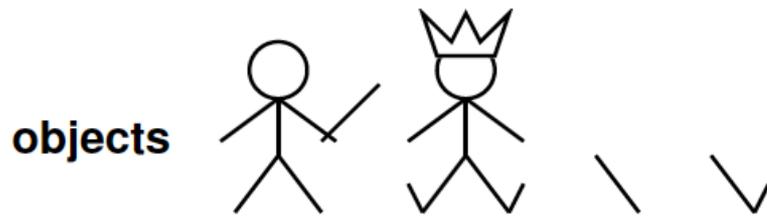
Verdade em Lógica de Primeira Ordem

- As frases bem formadas são avaliadas em **modelos (Estruturas)** $M = \langle D, I \rangle$ em que D é um conjunto não vazio de objectos (elementos do domínio) e I uma função de interpretação que especifica referentes para:
 - **Símbolos de constante** \rightarrow objectos $I(c) \in D$;
 - **Símbolos de predicado** \rightarrow relações $I(P) \subseteq D^n$ para pred P/n
 - **Símbolos de função** \rightarrow relações funcionais $I(f): D^n \rightarrow D$
- Uma frase atômica $\text{predicado}(termo_1, \dots, termo_n)$ é verdade sse os objectos referidos por $termo_1, \dots, termo_n$ se encontram na relação referida por predicado .
- Quando temos fórmulas com variáveis livres é necessário considerar atribuições de variáveis que as mapeiam em elementos do domínio de discurso

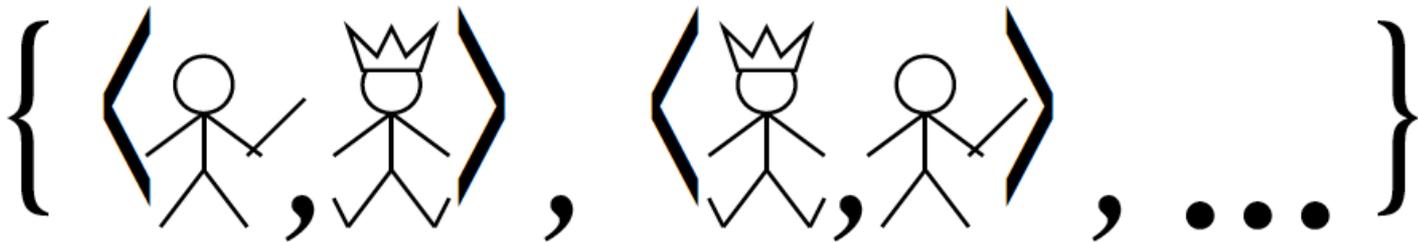
Modelos de LPO: Exemplo



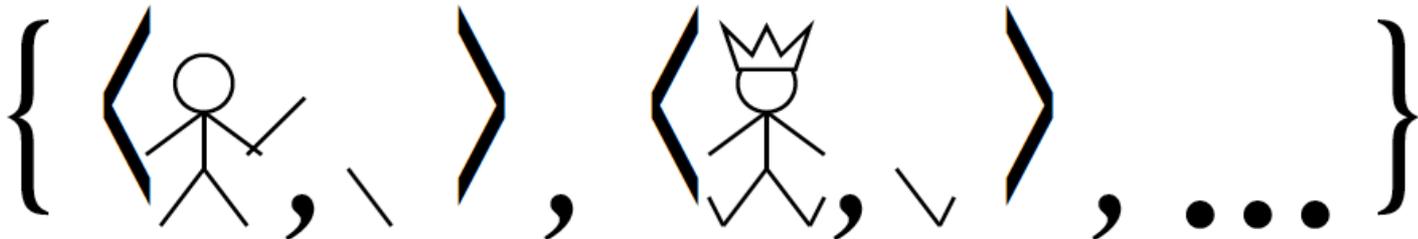
Modelos de LPO: Exemplo



relations: sets of tuples of objects



functional relations: all tuples of objects + "value" object



Modelos para LPO: Imensos!

- Podemos tentar enumerar os modelos para um dado vocabulário de uma KB:
 - Para cada número de elementos no domínio n de 1 até ∞
 - Para cada predicado k -ário P_k no vocabulário
 - Para cada relação k -ária possível sobre n objectos
 - Para cada símbolo de constante C no vocabulário
 - Para cada escolha de referente para C de entre n objectos ...
- A obtenção das conclusões lógicas por enumeração não vai ser nada fácil!

Denotação de um termo

- Seja t um termo e s uma atribuição de variáveis numa estrutura M
- A denotação $t_M[s]$ de t em M é definida recursivamente:
 - $x_M[s] = s(x)$ para uma variável x .
 - $c_M[s] = I[c]$ para uma constante c .
 - $f(t_1, \dots, t_n)_M[s] = I(f)((t_1)_M[s], \dots, (t_n)_M[s])$ para uma função $f(t_1, \dots, t_n)$.

LPO: Relação de satisfação

- A noção de verdade (relativa) em LPO é capturada através da relação de satisfação.
- Seja M uma estrutura e s uma atribuição de variáveis:
 - $M, s \models t_1 = t_2$ sse $(t_1)_M = (t_2)_M$
 - $M, s \models P(t_1, \dots, t_n)$ sse $((t_1)_M, \dots, (t_n)_M) \in I(P)$
 - $M, s \models \neg \varphi$ sse $M, s \not\models \varphi$
 - $M, s \models (\varphi \wedge \psi)$ sse $M, s \models \varphi$ e $M, s \models \psi$
 - $M, s \models (\varphi \vee \psi)$ sse $M, s \models \varphi$ ou $M, s \models \psi$
 - $M, s \models (\varphi \Rightarrow \psi)$ sse $M, s \models \psi$ ou $M, s \not\models \varphi$
 - $M, s \models (\exists x \varphi)$ sse $M, s' \models \varphi$ para alguma atribuição de variáveis s' idêntica a s , excepto possivelmente na variável x .
 - $M, s \models (\forall x \varphi)$ sse $M, s' \models \varphi$ para toda a atribuição de variáveis s' idêntica a s , excepto possivelmente na variável x .

Consequência Lógica

- Uma fórmula φ é **satisfazível** se existir uma estrutura M e uma atribuição de variáveis s tal que $M, s \models \varphi$.
- Um conjunto de fbfs Γ é **satisfazível** se existir uma estrutura M e uma atribuição de variáveis s tal que $M, s \models \varphi$ para toda a fórmula φ de Γ . Se Γ for um conjunto fechado de fórmulas, diz-se que M é um modelo de Γ .
- Uma fórmula φ é **lógicamente verdadeira** ou **válida** se $M, s \models \varphi$ para toda a estrutura M e atribuição de variáveis s (representado por $\models \varphi$).
- Se Γ é um conjunto de fbfs e φ uma fbf. Diz-se que φ é uma **consequência** de Γ sse para toda a interpretação M e atribuição de variáveis s , se $M, s \models \psi$ para toda a fórmula ψ de Γ então $M, s \models \varphi$. Representa-se este facto por $\Gamma \models \varphi$.

Quantificadores

- Permitem expressar propriedades de colecções de objectos, em vez de os enumerar por nome.
- Universal: “para todo” \forall
- Existencial: “existe” \exists

Quantificação Universal

- $\forall \langle \text{variáveis} \rangle \langle \text{frase} \rangle$
- Toda a gente na UNL é inteligente
 - $\forall x \text{Em}(x, \text{UNL}) \Rightarrow \text{Inteligente}(x)$
- $\forall x P$ é verdade num dado modelo M sse P é verdade para todo o objecto x do modelo.
- Pode ser entendido como a conjunção das instanciações de P :
 - $(\text{Em}(\text{ReiArtur}, \text{UNL}) \Rightarrow \text{Inteligente}(\text{ReiArtur}))$
 - $\wedge (\text{Em}(\text{Ana}, \text{UNL}) \Rightarrow \text{Inteligente}(\text{Ana}))$
 - $\wedge (\text{Em}(\text{UNL}, \text{UNL}) \Rightarrow \text{Inteligente}(\text{UNL}))$
 - $\wedge \dots$
- Normalmente \Rightarrow é o conectivo principal de \forall .
- Erro comum: utilizar \wedge como conectivo principal de \forall :
 - $\forall x \text{Em}(x, \text{UNL}) \wedge \text{Inteligente}(x)$
 - significa “Toda a gente está na UNL e toda a gente é inteligente”

Quantificação Existencial

- $\exists \langle \text{variáveis} \rangle \langle \text{frase} \rangle$
- Algém na UNL é inteligente
 - $\exists x Em(x, UNL) \wedge Inteligente(x)$
- $\exists x P$ é verdade num dado modelo M sse P é verdade para algum objecto x do modelo.
- Pode ser entendido como a disjunção das instanciações de P :
 - $(Em(ReiArtur, UNL) \wedge Inteligente(ReiArtur))$
 - $\vee (Em(Ana, UNL) \wedge Inteligente(Ana))$
 - $\vee (Em(UNL, UNL) \wedge Inteligente(UNL))$
 - $\vee \dots$
- Normalmente \wedge é o conectivo principal de \exists .
- Erro comum: utilizar \Rightarrow como conectivo principal de \exists :
 - $\exists x Em(x, UNL) \Rightarrow Inteligente(x)$
 - é verdade se houver alguém que não está na UNL!

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 $\forall y \exists x$.
 - $\exists x \forall y \text{ Ama}(x,y)$ significa “Existe alguém que ama toda a gente no mundo”
 - $\forall y \exists x \text{ Ama}(x,y)$ significa “Toda a gente no mundo é amada por alguém”
- **Dualidade dos quantificadores:**
 - $\forall x P$ é equivalente a $\neg \exists x \neg P$
 - $\forall x \text{ Gosta}(x, \text{Gelado}) \quad \neg \exists x \neg \text{Gosta}(x, \text{Gelado})$
 - $\exists x P$ é equivalente a $\neg \forall x \neg P$
 - $\exists x \text{ Gosta}(x, \text{Bróculos}) \quad \neg \forall x \neg \text{Gosta}(x, \text{Bróculos})$

Equivalências importantes da LPO

- $\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$
- $\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$
- $\forall x \forall y P(x,y) \Leftrightarrow \forall y \forall x P(x,y)$
- $\exists x \exists y P(x,y) \Leftrightarrow \exists y \exists x P(x,y)$
- $\forall x P(x) \wedge \forall x Q(x) \Leftrightarrow \forall x (P(x) \wedge Q(x))$
- $\exists x P(x) \vee \exists x Q(x) \Leftrightarrow \exists x (P(x) \vee Q(x))$

Alguns exemplos

- Irmãos são amigos
 - $\forall x,y \text{ Irmão}(x,y) \Rightarrow \text{Amigo}(x,y)$
- A relação entre irmãos é simétrica
 - $\forall x,y \text{ Irmão}(x,y) \Leftrightarrow \text{Irmão}(y,x)$
- A mãe de alguém é o seu progenitor feminino
 - $\forall x,y \text{ Mãe}(x,y) \Leftrightarrow (\text{Feminino}(x) \wedge \text{Progenitor}(x,y))$
- Um primo direito é um filho de um dos irmãos dos pais
 - $\forall x,y \text{ PrimoDireito}(x,y) \Leftrightarrow \exists p,ps (\text{Progenitor}(p,x) \wedge \text{Irmão}(ps,p) \wedge \text{Progenitor}(ps,y))$

Igualdade

- $termo_1 = termo_2$ é verdade numa dada interpretação sse $termo_1$ e $termo_2$ se referem ao mesmo objecto.
- A definição de *Irmão* em termos de *Progenitor*:
 - $\forall x, y \text{ Irmão}(x, y) \Leftrightarrow [\neg(x=y) \wedge \exists m, f (\neg(m=f) \wedge \text{Progenitor}(m, x) \wedge \text{Progenitor}(f, x) \wedge \text{Progenitor}(m, y) \wedge \text{Progenitor}(f, y)))]$

Interacção com KBs em LPO

- Suponhamos que um agente do mundo do Wumpus recorre a uma KB em LPO e percepção um cheiro e uma brisa (mas não um brilho) em $t = 5$:
 - $Tell(KB, Percept([Stench, Breeze, None], 5))$
 - $Ask(KB, \exists a BestAction(a, 5))$
 - I.e., Será que a KB conclui alguma melhor acção concreta para $t = 5$?
- Resposta: *Yes*; $\{a/Shoot\} \leftarrow$ **substituição** (binding list)
- Dada uma frase S e a substituição σ ,
 - $S\sigma$ denota o resultado aplicar σ a S ; e.g.,
 - $S = MaisInteligente(x, y)$
 - $\sigma = \{x=Hillary, y=Bill\}$
 - $S\sigma = MaisInteligente(Hillary, Bill)$
- $Ask(KB, S)$ devolve alguns/todos os σ tal que $KB \models S\sigma$

Versão LPO do Mundo do Wumpus

- Frase típica de percepção:
 - $Percept([Stench, Breeze, Glitter, None, None], 5)$
- Ações:
 - $Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb$
- Para determinar a melhor acção, construir a consulta:
 - $\forall a \text{ BestAction}(a, 5)$
- ASK resolve e retorna $\{a/Grab\}$

Base de Conhecimento para o Mundo de Wumpus

- **Percepção**

- $\forall b,g,m,c,t \text{ Percep}([Smell,b,g,m,c],t) \Rightarrow Smelt(t)$
- $\forall s,b,m,c,t \text{ Percep}([s,b,Glitter,m,c],t) \Rightarrow AtGold(t)$

- **Reflexo**

- $\forall t \text{ AtGold}(t) \Rightarrow BestAction(Grab,t)$

- **Reflexo com estado interno (já temos o ouro?)**

- $\forall t \text{ AtGold}(t) \wedge \neg Holding(Gold,t) \Rightarrow BestAction(Grab,t)$
- $Holding(Gold,t)$ não pode ser observado \Rightarrow manter registo das alterações é essencial!

Dedução de propriedades invisíveis

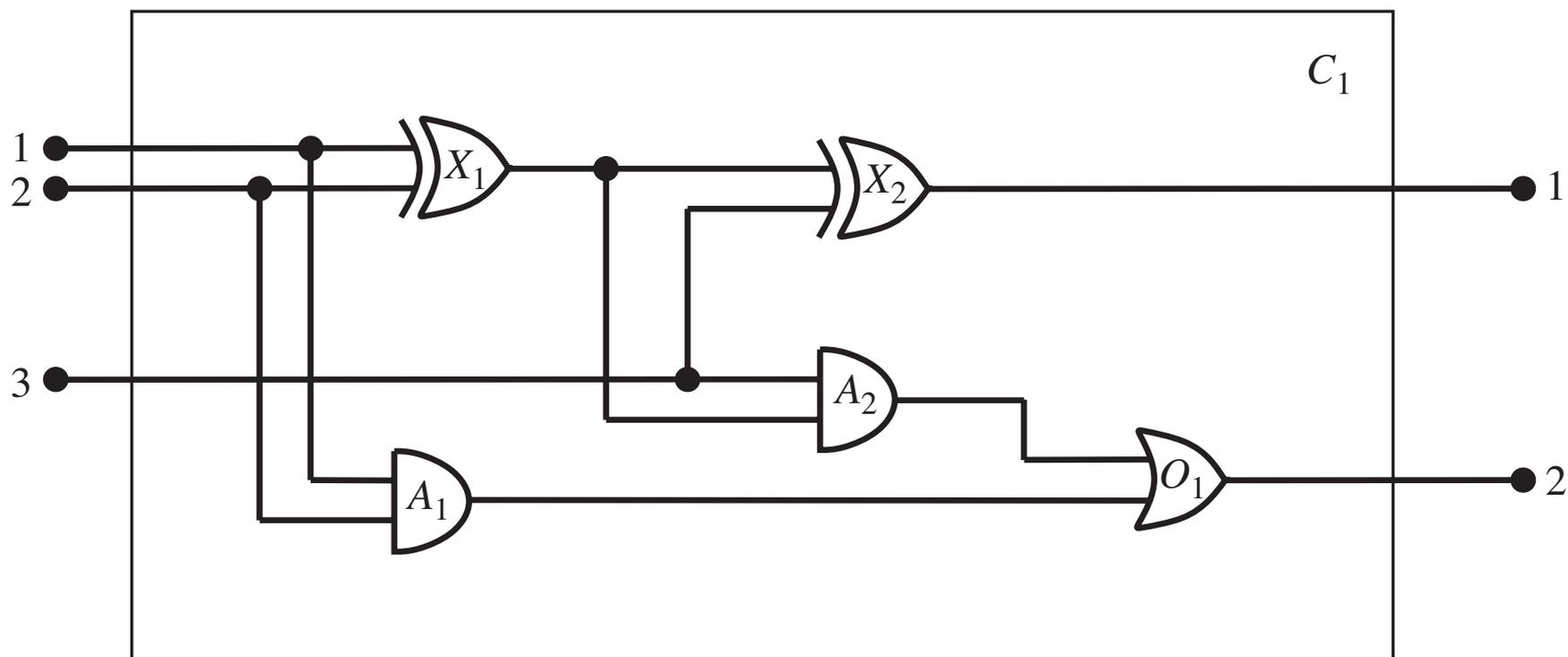
- $\forall x,y,a,b \text{ Adjacent}([x,y],[a,b]) \Leftrightarrow [a,b] \in \{[x+1,y],[x-1,y],[x,y+1],[x,y-1]\}$
- Propriedades das posições
 - $\forall x,t \text{ At}(\text{Agent},x,t) \wedge \text{Smelt}(t) \Rightarrow \text{Smelly}(x)$
 - $\forall x,t \text{ At}(\text{Agent},x,t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(x)$
- Casas são ventosas ao pé de um poço:
 - Definição para o predicado Breezy:
 - $\forall y \text{ Breezy}(y) \Leftrightarrow [\exists x \text{ Pitt}(x) \wedge \text{Adjacent}(x,y)]$
- Axiomas de sucessão de estado (um para cada predicado)
 - $\forall t \text{ HaveArrow}(t+1) \Leftrightarrow (\text{HaveArrow}(t) \wedge \neg \text{Action}(\text{Shoot},t))$

Engenharia de Conhecimento em LPO

1. Identificar a tarefa
2. Obter o conhecimento relevante
3. Decidir qual o vocabulário: predicados, funções e constantes
4. Codificar conhecimento genérico acerca do domínio
5. Codificar uma instância concreta
6. Interrogar a teoria utilizando um motor de inferência e obter respostas
7. Depurar (debug) a base de conhecimento

Circuitos digitais

- Somador de um bit



Circuitos digitais

1. Identificar a tarefa
 - Verificação: será que o circuito funciona como esperado?
2. Obter o conhecimento relevante
 - Composto por portas e fios
 - Tipos de portas (AND,OR,XOR,NOT)
 - Conexões entre terminais
 - Detalhes irrelevantes: cor, forma, tamanho, custo das portas,...
3. Decidir qual o vocabulário: predicados, funções e constantes
 - Alternativas:
 - $Type(X_1)=XOR$
 - $Type(X_1,XOR)$
 - $XOR(X_1)$
 - Escolha:
 - **Objectos**: $A_1, A_2, X_1, X_2, O_1, C_1, 0, 1, 2, 3, OR, AND, XOR, NOT$
 - **Funções**: $Signal/1, Type/1, In/2, Out/2$
 - **Predicados**: $Connected/2$

Circuitos digitais

4. Codificar conhecimento genérico acerca do domínio: ligações

- Se dois terminais estão ligados, então têm o mesmo sinal
 - $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$
- O sinal para todo o terminal é 0 ou 1 (mas não ambos)
 - $\forall t \text{ Signal}(t) = 0 \vee \text{Signal}(t) = 1$
 - $1 \neq 0$
- O predicado *Connected* é comutativo
 - $\forall t_1, t_2 \text{ Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$

Circuitos digitais

4. Codificar conhecimento genérico acerca do domínio: portas
- A saída de uma porta OR é 1 sse pelo menos uma das suas entradas é 1
 - $\forall g \text{ Type}(g) = \text{OR} \Rightarrow (\text{Signal}(\text{Out}(1,g))=1 \equiv \exists n \text{ Signal}(\text{In}(n,g))=1)$
 - A saída de uma porta AND é 0 sse pelo menos uma das suas entradas é 0
 - $\forall g \text{ Type}(g) = \text{AND} \Rightarrow (\text{Signal}(\text{Out}(1,g))=0 \equiv \exists n \text{ Signal}(\text{In}(n,g))=0)$
 - A saída de uma porta XOR é 1 sse as suas duas entradas são diferentes
 - $\forall g \text{ Type}(g) = \text{XOR} \Rightarrow (\text{Signal}(\text{Out}(1,g))=1 \equiv \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g)))$
 - A saída de uma porta NOT é diferente da sua entrada
 - $\forall g \text{ Type}(g) = \text{NOT} \Rightarrow (\text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g)))$

Circuitos digitais

5. Codificar uma instância concreta

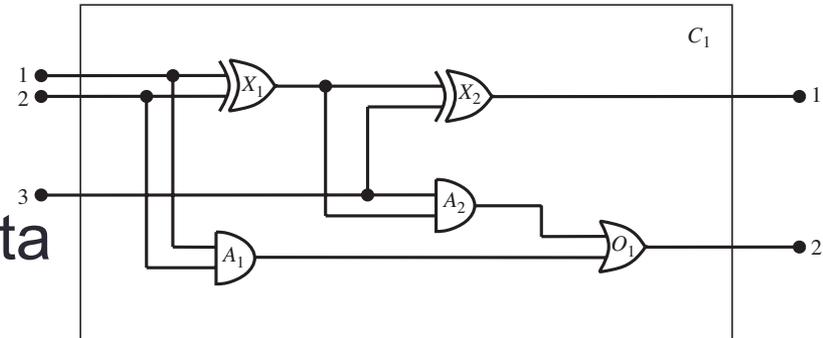
- Portas existentes

• $Type(X_1)=XOR$ $Type(X_2)=XOR$

• $Type(A_1)=AND$ $Type(A_2)=AND$

- Ligações entre os componentes:

- $Connected(Out(1,X_1),In(1,X_2))$
- $Connected(Out(1,X_1),In(2,A_2))$
- $Connected(Out(1,A_1),In(1,O_1))$
- $Connected(Out(1,A_2),In(2,O_1))$
- $Connected(Out(1,X_2),Out(1,C_1))$
- $Connected(Out(1,O_1),Out(2,C_1))$



$Type(O_1)=OR$

- $Connected(In(1,C_1),In(1,X_1))$
- $Connected(In(1,C_1),In(1,A_1))$
- $Connected(In(2,C_1),In(2,X_1))$
- $Connected(In(2,C_1),In(2,A_1))$
- $Connected(In(3,C_1),In(2,X_2))$
- $Connected(In(3,C_1),In(1,A_2))$

Circuitos digitais

6. Interrogar a teoria utilizando um motor de inferência e obter respostas
 - Saber quais os valores de input necessários para se ter a primeira saída a 0 e a segunda saída a 1?
 - $\exists i_1, i_2, i_3 \text{ Signal(In}(1, C_1))=i_1 \wedge \text{Signal(In}(2, C_1))=i_2 \wedge \text{Signal(In}(3, C_1))=i_3 \wedge \text{Signal(Out}(1, C_1))=0 \wedge \text{Signal(Out}(2, C_2))=1$
 - Quais os possíveis valores de todos os terminais (entrada/saída) do circuito?
 - $\exists i_1, i_2, i_3, o_1, o_2 \text{ Signal(In}(1, C_1))=i_1 \wedge \text{Signal(In}(2, C_1))=i_2 \wedge \text{Signal(In}(3, C_1))=i_3 \wedge \text{Signal(Out}(1, C_1))=o_1 \wedge \text{Signal(Out}(2, C_2))=o_2$
7. Depurar (debug) a base de conhecimento
 - Poderiam faltar asserções como $1 \neq 0$, etc...

Sumário

- Lógica de Primeira Ordem:
 - objectos e relações são primitivas semânticas
 - sintaxe: constantes, funções, predicados, igualdade, quantificadores
- Maior poder expressivo: suficiente para definir o mundo do Wumpus e circuitos electrónicos.