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 $B_{1,1} \land P_{1,2}$ 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 Ontologico	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 compromisos 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) Atómicas

- 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 $f(t_1, ..., t_n)$ onde f é uma função e $t_1, ..., t_n$ são termos.
  - Exemplos: *1, 12e40, pi, -3, Portugal, UNL, Diabo, Bem, Unicórnio, Abc, Xpto123, Key123, 'uma cadeia de caracteres muito longa', Exp(1.0), Exp(Mult(I,Pi)), Peso(Unicórnio) Mãe(Árbitro(Jogo(Académica,Belenenses,2006))), ...*
- Uma fórmula atómica é $t_1 = t_2$, ou $p(t_1, ..., t_n)$ onde p é um predicado e $t_1, ..., t_n$ são termos.
  - Exemplos: *Brother(KingJohn,RichardTheLionheart), Exp(I*Pi) + 1 = 0, 0 + x = x, Arco(a1,a2), > (Length(LeftLegOf(Richard)),Length(LeftLegOf(KingJohn))), Matriculado(s123,inf,ciclo1), ...*

Fórmulas (ou Frases) complexas

- As fórmulas complexas – aka fórmulas bem formadas (fbf) – são contruídas recursivamente a partir das fórmuals atómicas usando as conectivas e os quantificadores, através das seguintes regras:
  - Qualquer frase atómica 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

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

Variáveis livres e ligadas

- x é livre:
  - numa fórmula atómica 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:

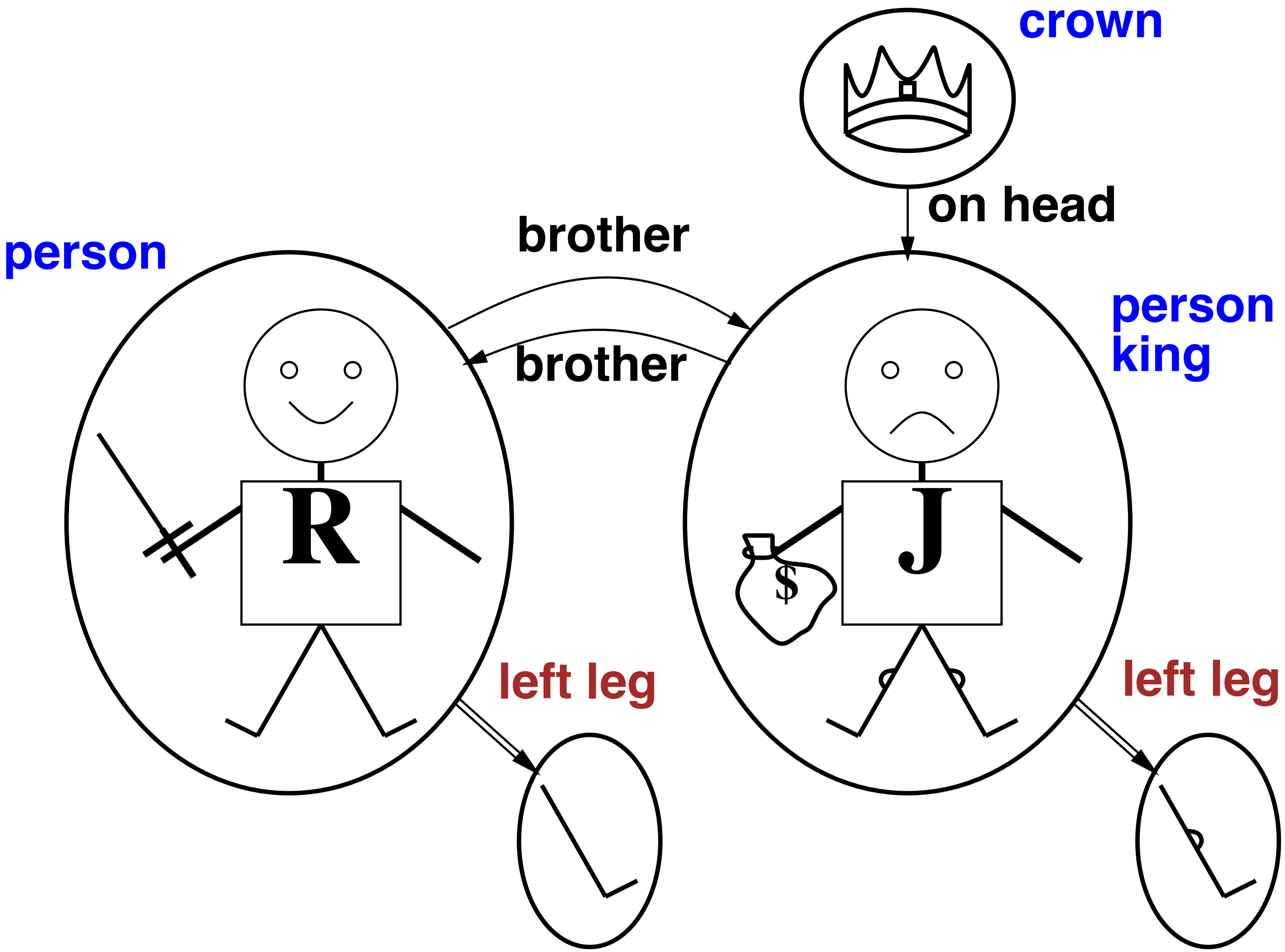
$$\Big(\forall x\big(R(x,y) \Rightarrow P(x)\big) \wedge \forall y\big(\neg R(x,y) \vee \forall x\, P(x)\big)\Big)$$

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

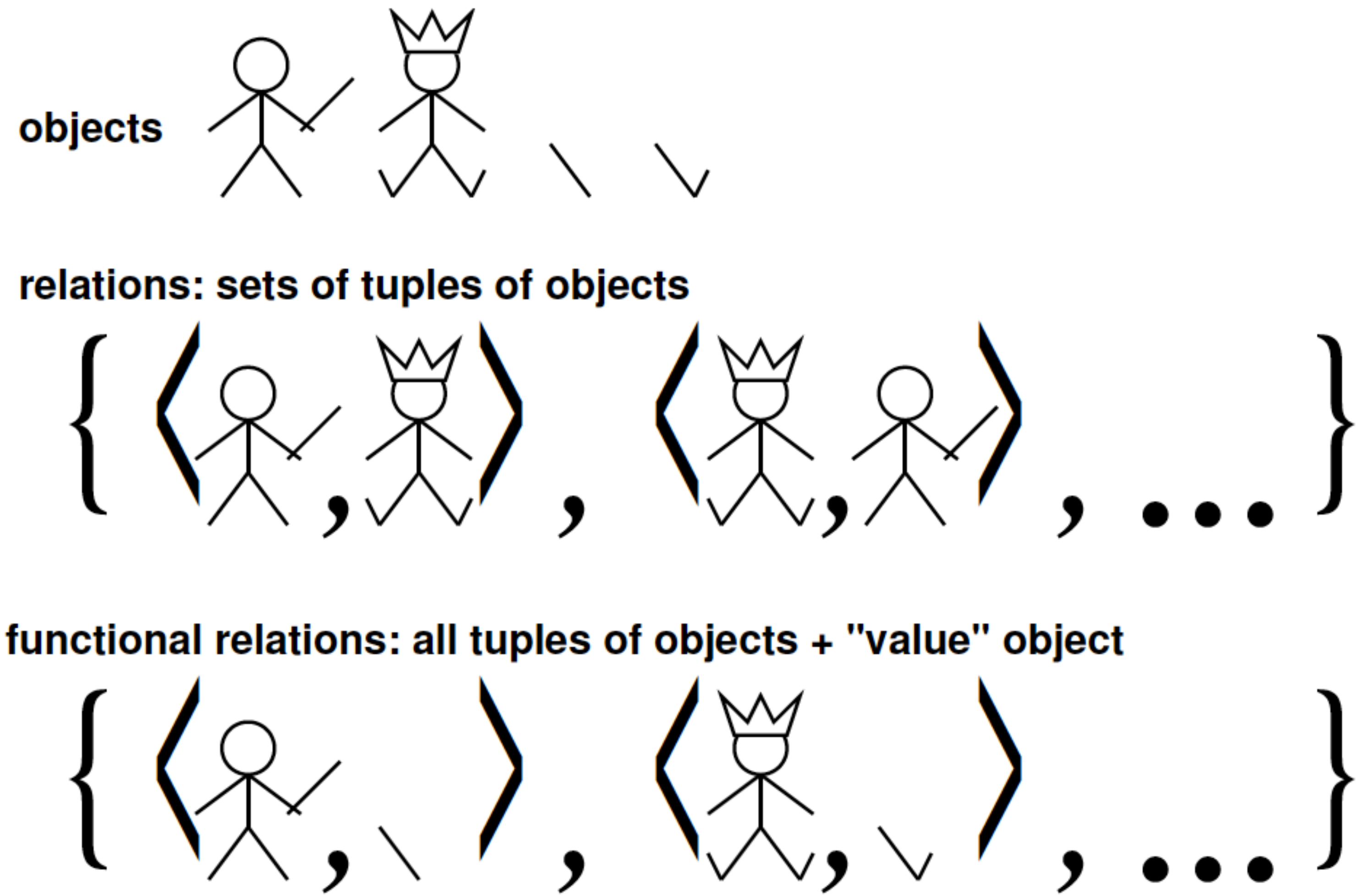
Verdade em Lógica de Primeira Ordem

- As frases bem formadas são avaliadas em modelos (Estruturas) $M=<D,I>$ 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 → objectos $I(c) \in D$;
  - Símbolos de predicado → relações $I(P) \subseteq D^n$ para pred P/n
  - Símbolos de função → relações funcionais $I(f){:}D^n \rightarrow D$
- Uma frase atómica $predicado(termo_1,\ldots,termo_n)$ é verdade sse os objectos referidos por $termo_1,\ldots,termo_n$ 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



Modelos para LPO: Imensas!

- 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.
  - $f(t_1,\ldots,t_n)_M[s]=I(f)((t_1)_M[s],\ldots,(t_n)_M[s])$ para uma função $f(t_1,\ldots,t_n)$.

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:
  - $M,s \vDash t_1{=}t_2$ sse $(t_1)_M{=}(t_2)_M$
  - $M,s \vDash P(t_1,\ldots,t_n)$ sse $((t_1)_M,\ldots,(t_n)_M) \in I(P)$
  - $M,s \vDash \neg\varphi$ sse $M,s \nvDash \varphi$
  - $M,s \vDash (\varphi \wedge \psi)$ sse $M,s \vDash \varphi$ e $M,s \vDash \psi$
  - $M,s \vDash (\varphi \vee \psi)$ sse $M,s \vDash \varphi$ ou $M,s \vDash \psi$
  - $M,s \vDash (\varphi \Rightarrow \psi)$ sse $M,s \vDash \psi$ ou $M,s \nvDash \varphi$
  - $M,s \vDash (\exists x\ \varphi)$ sse $M,s' \vDash \varphi$ para alguma atribuição de variáveis s' idêntica a s, excepto possívelmente na variável x.
  - $M,s \vDash (\forall x\ \varphi)$ sse $M,s' \vDash \varphi$ para toda a atribuição de variáveis s' idêntica a s, excepto possívelmente na variável x.

Consequência Lógica

- Uma fórmula φ é satisfazível se existir uma estrutura M e uma atribuição de variáveis s tal que $M,s \vDash \varphi$.
- Um conjunto de fbfs Γ é satisfazível se existir uma estrutura M e uma atribuição de variáveis s tal que $M,s \vDash \varphi$ 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 $M,s \vDash \varphi$ para toda a estrutura M e atribuição de variáveis s (representado por $\vDash \varphi$).
- 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 $M,s \vDash \psi$ para toda a fórmula ψ de Γ então $M,s \vDash \varphi$. Representa-se este facto por $\Gamma \vDash \varphi$.

Quantificadores

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

Quantificação Universal

- $\forall$ $\langle variáveis \rangle$ $\langle frase \rangle$
- Toda a gente na UNL é inteligente
  - $\forall x\ Em(x,UNL) \Rightarrow Inteligente(x)$
- $\forall x\ P$ 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:

$$\begin{aligned}&(Em(ReiArtur,UNL) \Rightarrow Inteligente(ReiArtur))\\ \wedge &(Em(Ana,UNL) \Rightarrow Inteligente(Ana\ )\\ \wedge &(Em(UNL,UNL) \Rightarrow Inteligente(UNL))\\ \wedge &\ldots\end{aligned}$$

- Normalmente $\Rightarrow$ é o conectivo principal de $\forall$.
- Erro comum: utilizar $\wedge$ como conectivo principal de $\forall$:
  - $\forall x\ Em(x,UNL) \wedge Inteligente(x)$
  - significa "Toda a gente está na UNL e toda a gente é inteligente"

Quantificação Existêncial

- $\exists$ *<variáveis> <frase>*
- 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:

$$(Em(ReiArtur, UNL) \wedge Inteligente(ReiArtur))$$
$$\vee \ (Em(Ana, UNL) \wedge Inteligente(Ana))$$
$$\vee (Em(UNL, UNL) \wedge Inteligente(UNL))$$
$$\vee \ldots$$

- 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 $\forall y\ \exists x$.
  - $\exists x\ \forall y\ Ama(x,y)$ significa "Existe alguém que ama toda a gente no mundo"
  - $\forall y\ \exists x\ Ama(x,y)$ significa "Toda a gente no mundo é amada por alguém"
- Dualidade dos quantificadores:
  - $\forall x\ P$ é equivalente a $\neg \exists x\ \neg P$
    - $\forall x\ Gosta(x, Gelado)$ $\qquad \neg \exists x\ \neg Gosta(x, Gelado)$
  - $\exists x\ P$ é equivalente a $\neg \forall x\ \neg P$
    - $\exists x\ Gosta(x, Brócolos)$ $\qquad \neg \forall x\ \neg Gosta(x, Brócolos)$

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

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*:
  - $\forall x,y\ Irmão(x,y) \Leftrightarrow [\neg(x=y) \wedge \exists m,f\ (\neg(m=f) \wedge Progenitor(m,x) \wedge Progenitor(f,x) \wedge Progenitor(m,y) \wedge Progenitor(f,y))]$

Interacção com KBs em LPO

- Suponhamos que um agente do mundo do Wumpus recorre a uma KB em LPO e percepciona 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; $\{a/Shoot\}$ ← 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 $KB \vDash S\sigma$

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:
  - $\forall a \; BestAction(a, 5)$
- *ASK* resolve e retorna $\{a/Grab\}$

Base de Conhecimento para o Mundo de Wumpus

- Percepção
  - $\forall b,g,m,c,t\ Percep([Smell,b,g,m,c],t) \Rightarrow Smelt(t)$
  - $\forall s,b,m,c,t\ Percep([s,b,Glitter,m,c],t) \Rightarrow AtGold(t)$
- Reflexo
  - $\forall t\ AtGold(t) \Rightarrow BestAction(Grab,t)$
- Reflexo com estado interno (já temos o ouro?)
  - $\forall t\ AtGold(t) \land \neg Holding(Gold,t) \Rightarrow BestAction(Grab,t)$
  - $Holding(Gold,t)$ não pode ser observado $\Rightarrow$ manter registo das alterações é essencial!

Dedução de propriedades invisíveis

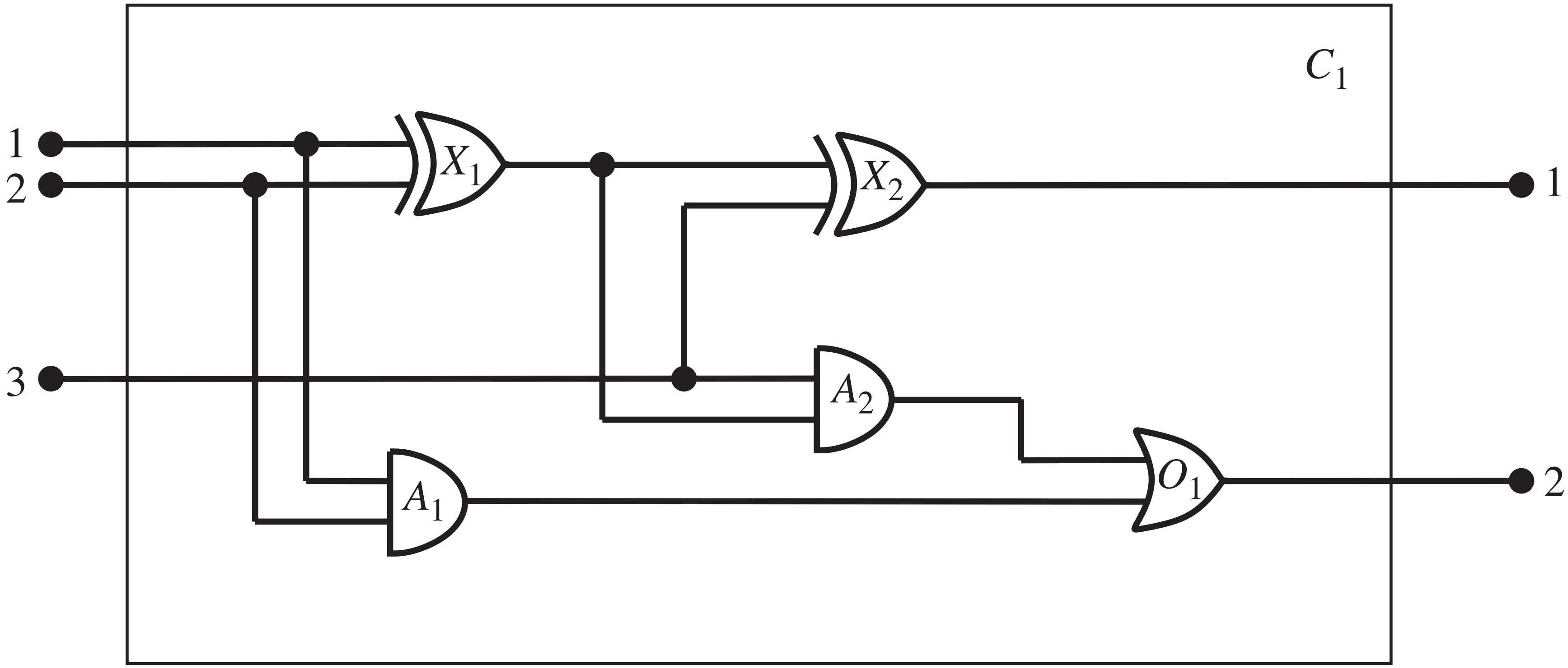
- $\forall x,y,a,b\ Adjacent([x,y],[a,b]) \Leftrightarrow [a,b] \in \{[x+1,y],[x-1,y],[x,y+1],[x,y-1]\}$
- Propriedades das posições
  - $\forall x,t\ At(Agent,x,t) \wedge Smelt(t) \Rightarrow Smelly(x)$
  - $\forall x,t\ At(Agent,x,t) \wedge Breeze(t) \Rightarrow Breezy(x)$
- Casas são ventosas ao pé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

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: $Connected/2$

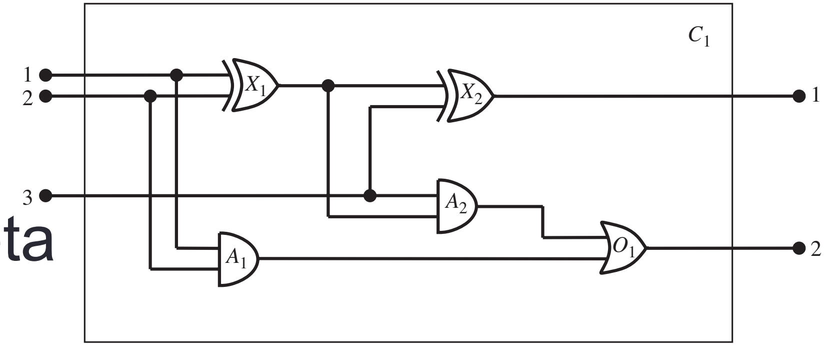
Circuitos digitais

4. Codificar conhecimento genérico acerca do domínio: ligações
   - Se dois terminais estão ligados, então têm o mesmo sinal
     - $\forall\, t_1, t_2\ Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$
   - O sinal para todo o terminal é 0 ou 1 (mas não ambos)
     - $\forall\, t\ Signal(t) = 0 \lor Signal(t) = 1$
     - $1 \neq 0$
   - O predicado *Connected* é comutativo
     - $\forall\, t_1, t_2\ Connected(t_1, t_2) \Rightarrow Connected(t_2, t_1)$

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
        - $\forall g\ Type(g) = OR \Rightarrow (Signal(Out(1,g)) = 1 \equiv \exists n\ Signal(In(n,g)) = 1)$
    - A saída de uma porta AND é 0 sse pelo menos uma das suas entradas é 0
        - $\forall g\ Type(g) = AND \Rightarrow (Signal(Out(1,g)) = 0 \equiv \exists n\ Signal(In(n,g)) = 0)$
    - A saída de uma porta XOR e 1 sse as suas duas entradas são diferentes
        - $\forall g\ Type(g) = XOR \Rightarrow (Signal(Out(1,g)) = 1 \equiv Signal(In(1,g)) \neq Signal(In(2,g)))$
    - A saída de uma porta NOT é diferente da sua entrada
        - $\forall g\ Type(g) = NOT \Rightarrow (Signal(Out(1,g)) \neq Signal(In(1,g)))$

Circuitos digitais



5. Codificar uma instância concreta
   - Portas existentes
   - $Type(X_1)=XOR$ $Type(X_2)=XOR$
   - $Type(A_1)=AND$ $Type(A_2)=AND$ $Type(O_1)=OR$
   - Ligações entre os componentes:
   - $Connected(Out(1,X_1),In(1,X_2))$ $Connected(In(1,C_1),In(1,X_1))$
   - $Connected(Out(1,X_1),In(2,A_2))$ $Connected(In(1,C_1),In(1,A_1))$
   - $Connected(Out(1,A_1),In(1,O_1))$ $Connected(In(2,C_1),In(2,X_1))$
   - $Connected(Out(1,A_2),In(2,O_1))$ $Connected(In(2,C_1),In(2,A_1))$
   - $Connected(Out(1,X_2),Out(1,C_1))$ $Connected(In(3,C_1),In(2,X_2))$
   - $Connected(Out(1,O_1),Out(2,C_1))$ $Connected(In(3,C_1),In(1,A_2))$

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?
        - $\exists\, i_1,i_2,i_3\ Signal(In(1,C_1))=i_1 \wedge Signal(In(2,C_1))=i_2 \wedge Signal(In(3,C_1))=i_3 \wedge Signal(Out(1,C_1))=0 \wedge Signal(Out(2,C_2))=1$
    - Quais os possíveis valores de todos os terminais (entrada/saída) do circuito?
        - $\exists\, i_1,i_2,i_3,o_1,o_2\ Signal(In(1,C_1))=i_1 \wedge Signal(In(2,C_1))=i_2 \wedge Signal(In(3,C_1))=i_3 \wedge Signal(Out(1,C_1))=o_1 \wedge Signal(Out(2,C_2))=o_2$
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.