PROGRAMAÇÃO EM LÓGICA

FUNDAÇÕES

Henrique Lopes Cardoso, DEI/FEUP, 2013

Plano

- Conhecimento
  - representação
  - inferência
- Lógica proposicional: revisões
- Lógica de primeira ordem
- Programação em Lógica

Representação de Conhecimento

- Base de conhecimento (*KB*)
  - conjunto de frases expressas numa linguagem de representação de conhecimento

- Sintaxe: especifica as frases que estão bem formadas
  - e.g. "x+y=4", mas não "x4y+="
- Semântica: atribui significado às frases, determina veracidade em relação a cada "mundo possível" ou *modelo*
  - e.g. "x+y=4" é verdade num mundo em que x e y são ambos 2

- Uma interpretação é um mapeamento entre: constantes da lógica e objetos de um determinado domínio; funções da lógica e funções do domínio (propriedades dos objetos); e predicados da lógica e relações nesse domínio

- Se a frase $\boldsymbol{\alpha}$ é verdadeira numa interpretação ***m***: diz-se que ***m*** **é um modelo de** $\boldsymbol{\alpha}$

- $M(\alpha)$: conjunto de todos os modelos de α

Raciocínio Lógico

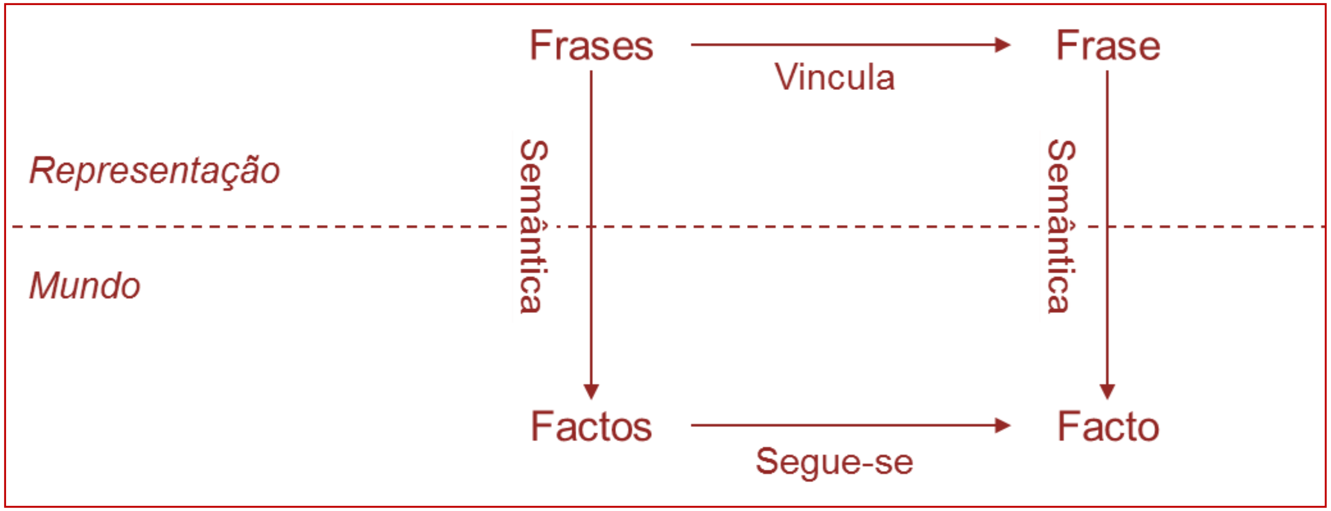
- **Implicação lógica** (Logical Implication)
  - $\alpha \vDash \beta$
    - α implica logicamente β (ou de α segue-se logicamente β)
    - $\alpha \vDash \beta$ se e só se $M(\alpha) \subseteq M(\beta)$
    - α será portanto uma asserção mais forte do que β

- Inferência lógica ou dedução
  - aplica a relação de implicação lógica entre frases para derivar conclusões

- Procedimento de inferência
  - **correto** (*sound*) se deriva *apenas* frases vinculadas
  - **completo** se é capaz de derivar *todas* as frases vinculadas

Correspondência

- Se *KB* é verdade no mundo real, então qualquer frase α derivada de *KB* por um procedimento de inferência correto é também verdade no mundo real



- O procedimento de inferência
  - opera sobre as representações sintáticas (frases), mas corresponde à relação existente entre os factos no mundo real
  - consiste em construir novas frases a partir das existentes
  - para ser correto, deve derivar apenas novas frases que representem factos que se seguem dos factos representados pela *KB*

Plano

- Conhecimento
- Lógica proposicional: revisões
  - sintaxe
  - semântica
  - inferência
- Lógica de primeira ordem
- Programação em Lógica

Lógica Proposicional: Sintaxe

- Símbolos:
  - constantes lógicas $True$ e $False$
  - símbolos proposicionais como P e Q
  - conectivas lógicas: $\wedge \quad \vee \quad \Rightarrow \quad \Leftrightarrow \quad \neg$
  - parêntesis (e)

- Frases são agrupamentos de símbolos, tais que:
  - $True$, $False$, P ou Q são frases por si só
  - frase dentro de parêntesis é uma frase: $(P \wedge Q)$
  - frase complexa combinando frases mais simples com conectivas lógicas:
    - $\wedge$ (e) – frase cuja conectiva principal é $\wedge$ chama-se uma **conjunção**: $P \wedge (Q \vee R)$
    - $\vee$ (ou) – frase cuja conectiva principal é $\vee$ chama-se uma **disjunção**: $A \vee (P \wedge Q)$
    - $\Rightarrow$ (implica) – frase do tipo $(P \wedge Q \Rightarrow R)$ chama-se uma **implicação material** <u>(ou simplesmente **implicação**</u>)
    - $\Leftrightarrow$ (equivalente) – a frase $(P \wedge Q) \Leftrightarrow (Q \wedge P)$ é uma **equivalência**
    - $\neg$ (não) – uma frase como $\neg P$ chama-se a **negação** de P
  - ordem de precedência dos operadores: $\neg \quad \wedge \quad \vee \quad \Rightarrow \quad \Leftrightarrow$
    - a frase $\neg P \vee Q \wedge R \Rightarrow S$ é equivalente à frase $\big((\neg P) \vee (Q \wedge R)\big) \Rightarrow S$

Lógica Proposicional: Semântica

- Símbolo proposicional pode ter como interpretação qualquer facto
- $True$ representa sempre um facto verdadeiro
- $False$ representa sempre um facto falso
- Frase complexa tem significado derivado do significado das partes
  - valor pode ser obtido através de uma tabela de verdade
- Tabela de verdade para as conectivas lógicas:

P	Q	$\neg P$	$P \land Q$	$P \lor Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	*False*	*True*	*False*	*False*	*True*	*True*
False	*True*	*True*	*False*	*True*	*True*	*False*
True	*False*	*False*	*False*	*True*	*False*	*False*
True	*True*	*False*	*True*	*True*	*True*	*True*

- Frases complexas são definidas por um processo de decomposição
  - $(P \lor Q) \land \neg S$: primeiro determina-se o significado de $(P \lor Q)$ e de $\neg S$, depois combinam-se os dois usando a definição de $\land$

Equivalência

- Duas frases α e β são **logicamente equivalentes** se são verdade no mesmo conjunto de modelos: $M(\alpha) = M(\beta)$
  - $\alpha \equiv \beta$ se e só se $\alpha \vDash \beta$ e $\beta \vDash \alpha$

$$
\begin{gathered}
(\alpha \wedge \beta) \equiv (\beta \wedge \alpha) \\
(\alpha \vee \beta) \equiv (\beta \vee \alpha) \\
\big((\alpha \wedge \beta) \wedge \gamma\big) \equiv \big(\alpha \wedge (\beta \wedge \alpha)\big) \\
\big((\alpha \vee \beta) \vee \gamma\big) \equiv \big(\alpha \vee (\beta \vee \alpha)\big) \\
\neg(\neg\alpha) \equiv \alpha \\
(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha) \\
(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta) \\
(\alpha \Leftrightarrow \beta) \equiv \big((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)\big) \\
\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta) \\
\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta) \\
\big(\alpha \wedge (\beta \vee \gamma)\big) \equiv \big((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)\big) \\
\big(\alpha \vee (\beta \wedge \gamma)\big) \equiv \big((\alpha \vee \beta) \wedge (\alpha \vee \gamma)\big)
\end{gathered}
$$

Validade e Consistência

- **Validade**
  - uma frase é válida se for verdadeira em todos os modelos (**tautologia**: frase necessariamente verdadeira)
    - $P \vee \neg P$
  - $\alpha \vDash \beta$ se e só se $(\alpha \Rightarrow \beta)$ é uma frase válida

- **Consistência** (*satisfiability*)
  - uma frase é consistente (satisfazível) se for verdadeira nalgum modelo
    - α é válida se $\neg\alpha$ for inconsistente
    - α é consistente se $\neg\alpha$ for não válida
    - $\alpha \vDash \beta$ se e só se $(\alpha \wedge \neg\beta)$ for uma frase inconsistente
      - princípio da *prova por contradição*

Teste de Validade

- Tabelas de verdade podem ser usadas para testar frases válidas
  - se a frase for verdadeira em todas as linhas, então é válida
    - $\big((P \lor H) \land \neg H\big) \Rightarrow P$

P	H	$P \lor H$	$(P \lor H) \land \neg H$	$\big((P \lor H) \land \neg H\big) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

- **Regras de inferência** permitem-nos fazer inferências sem necessidade de construir tabelas de verdade
  - uma regra de inferência é lógica (*sound*) se a conclusão é verdadeira sempre que as premissas são verdadeiras

Regras de Inferência

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}$$

- **Modus Ponens**: de uma implicação e da sua premissa, podemos inferir a conclusão

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n}{\alpha_i}$$

- **And-Elimination**: de uma conjunção, podemos inferir qualquer dos conjuntores

$$\frac{\alpha_1, \alpha_2, \ldots, \alpha_n}{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n}$$

- **And-Introduction**: de uma lista de frases, podemos inferir a sua conjunção

$$\frac{\alpha_i}{\alpha_1 \vee \alpha_2 \vee \cdots \vee \alpha_n}$$

- **Or-Introduction**: de uma frase, podemos inferir a sua disjunção com qualquer coisa

$$\frac{\neg\neg\alpha}{\alpha}$$

- **Double-Negation Elimination**: de uma frase duplamente negada, podemos inferir uma frase afirmada

$$\frac{\alpha \vee \beta, \quad \neg\beta}{\alpha}$$

- **Unit Resolution**: de uma disjunção, se um dos disjuntores é falso, podemos inferir que o outro é verdadeiro
- **Resolution**: como β não pode ser verdadeiro e falso, um dos outros disjuntores tem que ser verdadeiro

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{ou} \quad \frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Plano

- Conhecimento
- Lógica proposicional: revisões
- Lógica de 1ª ordem
  - sintaxe, semântica e inferência
  - unificação
  - cláusulas de Horn
  - resolução
  - Forma Normal Conjuntiva
- Programação em Lógica

Lógica de Primeira Ordem

- Mundo consiste em objetos com propriedades e relações

- Símbolos
  - Constantes (representam objetos): $A, B, C, João, PaiDoJoão, \ldots$
  - Predicados (representam relações): $Redondo, Irmão, MenorQue, \ldots$
  - Funções (relações com um só valor possível): $Coseno, Pai, PernaEsquerda, \ldots$
- Variáveis: $a, x, s, \ldots$
- Termos: constituídos por constantes, variáveis ou funções
  - $João, x, PernaEsquerda(João), \ldots$

- Frases
  - Atómicas: predicado e lista de termos
    - $Irmão(Ricardo, João)$
    - $Casados\big(Pai(Ricardo), Mãe(João)\big)$
  - Complexas: usar conectivas lógicas
    - $\neg \quad \wedge \quad \vee \quad \Rightarrow \quad \Leftrightarrow$

Quantificadores (∀ e ∃)

- **Universal** (∀): expressar propriedades de colecções de objetos
  - "Todos os gatos são mamíferos."
    - $\forall x\ Gato(x) \Rightarrow Mamífero(x)$
- **Existencial** (∃): afirmar sobre um objeto sem dizer qual é
  - "O João tem uma irmã que é casada."
    - $\exists x\ Irmã(x, João) \wedge Casada(x)$

- Múltiplos quantificadores:
  - $\forall x, y \equiv \forall x\ \forall y \equiv \forall y\ \forall x$
  - $\forall x\ \exists y \neq \exists y\ \forall x$
    - $\forall x\ \exists y\ Gosta(x, y)$ "Toda a gente gosta de alguém."
    - $\exists y\ \forall x\ Gosta(x, y)$ "Existe alguém de quem toda a gente gosta."
    - $\forall y\ \exists x\ Gosta(x, y)$ "Toda a gente tem alguém que gosta dela."
    - $\exists x\ \forall y\ Gosta(x, y)$ "Existe alguém que gosta de toda a gente."
- Ligações entre ∀ e ∃, através da negação (leis de De Morgan)
  - $\forall x\ \neg Gosta(x, Exames) \equiv \neg\exists x\ Gosta(x, Exames)$
  - $\forall x\ Gosta(x, Saúde) \equiv \neg\exists x\ \neg Gosta(x, Saúde)$
  - $\exists x\ \neg Gosta(x, Sopa) \equiv \neg\forall x\ Gosta(x, Sopa)$
  - $\exists x\ Gosta(x, Sopa) \equiv \neg\forall x\ \neg Gosta(x, Sopa)$

Regras de Inferência com Quantificadores

- São mais complexas: consistem em substituir variáveis por indivíduos particulares
- $\boldsymbol{SUBST(\theta, \alpha)}$ representa o resultado de aplicar a **substituição** θ à frase α
  - $SUBST(\{x/João, y/Couve\}, Gosta(x, y)) = Gosta(João, Couve)$

- **Universal Elimination**:
  - para qualquer frase α, variável v e termo g: $$\frac{\forall v\ \alpha}{SUBST(\{v/g\}, \alpha)}$$
  - p. ex., de $\forall x\ Gosta(x, Gelado)$, podemos usar a substituição $\{x/Rui\}$ e inferir $Gosta(Rui, Gelado)$

- **Existential Elimination**:
  - para qualquer frase α, variável v e constante k ainda não utilizada: $$\frac{\exists v\ \alpha}{SUBST(\{v/k\}, \alpha)}$$
  - no fundo, damos um nome ao objeto que satisfaz a condição existencial
  - p. ex., de $\exists x\ Matou(x, Vítima)$ podemos inferir $Matou(Assassino, Vítima)$ desde que $Assassino$ não seja o nome de outro objeto qualquer

- **Existential Introduction**:
  - para qualquer frase α, variável v que não ocorre em α e termo g: $$\frac{\alpha}{\exists v\ SUBST(\{g/v\}, \alpha)}$$
  - p. ex., de $Gosta(João, Gelado)$ podemos inferir $\exists x\ Gosta(x, Gelado)$

Exemplo de Utilização

- É crime vender mísseis:
  $\forall x, y, z \; Missile(y) \land Sells(x, z, y) \Rightarrow Criminal(x)$ (1)
- O país *Nono* tem mísseis:
  $\exists x \; Owns(Nono, x) \land Missile(x)$ (2)
- Todos os seus mísseis foram vendidos pelo *West*:
  $\forall x \; Owns(Nono, x) \land Missile(x) \Rightarrow Sells(West, Nono, x)$ (3)
- *West* é um criminoso?
  - (2) e Existential Elimination:
    $Owns(Nono, M1) \land Missile(M1)$ (4)
  - (4) e And-Elimination:
    $Owns(Nono, M1)$ (5)
    $Missile(M1)$ (6)
  - (3) e Universal Elimination:
    $Owns(Nono, M1) \land Missile(M1) \Rightarrow Sells(West, Nono, M1)$ (7)
  - (7), (4) e Modus Ponens:
    $Sells(West, Nono, M1)$ (8)
  - (1) e Universal Elimination:
    $Missile(M1) \land Sells(West, Nono, M1) \Rightarrow Criminal(West)$ (9)
  - (6), (8) e And-Introduction:
    $Missile(M1) \land Sells(West, Nono, M1)$ (10)
  - (9), (10) e Modus Ponens:
    $Criminal(West)$ (11)

Generalização do Modus Ponens

- Para frases atómicas p_i, p_i' e q, se houver uma substituição θ tal que $SUBST(\theta, p_i') = SUBST(\theta, p_i)$ para todo o i:

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

- I.e., se houver uma substituição que torne as premissas da implicação idênticas a frases que estejam na KB, então podemos inferir a conclusão da implicação, depois de aplicar a substituição
  - (3), (5), (6) e Gen. Modus Ponens: $\theta = \{x/M1\}$ unifica $Owns(Nono, x) \wedge Missile(x)$ com $Owns(Nono, M1)$ e $Missile(M1)$, obtendo $Sells(West, Nono, M1)$
  - (1), (6), (8) e Gen. Modus Ponens: $\theta = \{x/West, y/M1, z/Nono\}$ unifica $Missile(y) \wedge Sells(x, z, y)$ com $Missile(M1)$ e $Sells(West, Nono, M1)$, obtendo $Criminal(West)$

- Mais eficiente pois combina várias inferências numa só

- Utiliza um algoritmo de **unificação**, que toma duas frases e retorna uma substituição que as torne idênticas (se houver)

Unificação

- Dadas duas frases atómicas p e q retorna uma substituição que as torne idênticas: $\boldsymbol{UNIFY}(\boldsymbol{p},\boldsymbol{q}) = \boldsymbol{\theta}$ onde $SUBST(\theta,p) = SUBST(\theta,q)$
  - diz-se que θ é o unificador das duas frases

- Exemplo: de quem é que o João gosta?

  $Conhece(João, x) \Rightarrow Gosta(João, x)$

  $Conhece(João, Joana)$

  $Conhece(y, Mãe(y))$

  $Conhece(y, Leonel)$

  $Conhece(x, Elizabete)$

  - Modus Ponens: encontrar frases que unificam com a premissa e aplicar o unificador à conclusão

    $UNIFY\big(Conhece(João, x), Conhece(João, Joana)\big) = \{x/Joana\}$

    $UNIFY\big(Conhece(João, x), Conhece(y, Leonel)\big) = \{x/Leonel, y/João\}$

    $UNIFY\Big(Conhece(João, x), Conhece\big(y, Mãe(y)\big)\Big) = \{y/João, x/Mãe(João)\}$

    $UNIFY\big(Conhece(João, x), Conhece(x_2, Elizabete)\big) = \{x/Elizabete, x_2/João\}$

    - as frases $\forall x\ Conhece(x, Elizabete)$ e $\forall x_2\ Conhece(x_2, Elizabete)$ têm o mesmo significado

- Unificador Mais Geral (MGU)
  - Substituição que compromete as variáveis o mínimo possível

    $UNIFY\big(Conhece(João, x), Conhece(y, z)\big) = \{y/João, x/z\}$

    - em vez de, por exemplo, $\{y/João, x/João, z/João\}$

Cláusulas definidas

- "Limitação" do Modus Ponens: todas as frases da KB têm que estar numa das formas das premissas – frases ou **cláusulas definidas**:
  - frases atómicas
  - implicações com conjunção de frases atómicas na premissa e uma só frase atómica na conclusão

- Forma genérica: $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Rightarrow Q$

- Casos especiais:
  - se Q for $False$, obtemos uma frase do tipo $\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n$
  - se $n = 1$ e $P_1 = True$, obtemos $True \Rightarrow Q$, o que é equivalente à frase atómica Q

- Cláusulas definidas têm exactamente um literal positivo
  - a forma genérica pode ser escrita como $\neg P_1 \vee \neg P_2 \vee \cdots \vee \neg P_n \vee Q$

- Exemplos de frases que ficam de fora (mais do que um literal positivo, cláusulas não definidas):
  - $\forall x\ Pessoa(x) \Rightarrow Homem(x) \vee Mulher(x)$
  - $\forall x\ \neg Português(x) \Rightarrow Estrangeiro(x)$

Generalização da Resolução

- Cláusulas não de Horn:
  - forma genérica: $p_1 \lor p_2 \lor \cdots \lor p_n$
  - ou: $p_1 \land p_2 \land \cdots \land p_{n1} \Rightarrow q_1 \lor q_2 \lor \cdots \lor q_{n2}$
- Generalizando a regra de inferência Resolution:
  - para duas disjunções de qualquer tamanho, se um dos disjuntores numa cláusula unificar com a *negação* de um disjuntor na outra cláusula, então podemos inferir a disjunção de todos os disjuntores excepto estes dois:

$$\left.\begin{array}{c} a \lor h \lor c \\ d \lor \neg h \lor e \end{array}\right\rangle \; a \lor c \lor d \lor e$$

  - para frases atómicas p_i e q_i, onde $UNIFY(p_j, \neg q_k) = \theta$:

$$\frac{\begin{array}{c} p_1 \lor \cdots \lor p_j \lor \cdots \lor p_m \\ q_1 \lor \cdots \lor q_k \lor \cdots \lor q_n \end{array}}{SUBST(\theta, p_1 \lor \cdots \lor p_{j-1} \lor p_{j+1} \lor \cdots \lor p_m \lor q_1 \lor \cdots \lor q_{k-1} \lor q_{k+1} \lor \cdots \lor q_n)}$$

  - qualquer frase em lógica de primeira ordem pode ser convertida na forma das premissas da regra da Resolução: **forma normal conjuntiva (CNF)**

Forma Normal Conjuntiva

- Conversão de frases para a CNF:
  - Eliminar implicações
    $p \Rightarrow q \equiv \neg p \vee q$
  - Mover as negações ($\neg$) para os átomos:
    $\neg(p \vee q) \equiv \neg p \wedge \neg q$ $\quad$ $\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\quad$ $\neg\neg p \equiv p$
    $\neg\forall x\, p \equiv \exists x\, \neg p$ $\quad$ $\neg\exists x\, p \equiv \forall x\, \neg p$
  - Estandardizar variáveis
    - $\big(\forall x\, P(x)\big) \vee \big(\exists x\, Q(x)\big)$ fica $\big(\forall x\, P(x)\big) \vee \big(\exists y\, Q(y)\big)$
  - Mover os quantificadores para a esquerda
    $p \vee \forall x\, q$ fica $\forall x\, p \vee q$ – é possível pois sabemos que p não contém x
  - **Skolemização**: processo de remover quantificadores existenciais
  - Já só há quantificadores universais: ignorá-los
  - Distribuir $\wedge$ por $\vee$:
    $(a \wedge b) \vee c$ fica $(a \vee c) \wedge (b \vee c)$

- CNF: a frase é uma conjunção onde cada conjuntor é uma disjunção de literais

Skolemização

- Caso mais simples: Existential Elimination
  - $\exists x\ P(x)$ fica $P(A)$, onde A é uma constante ainda não utilizada

- Se existirem quantificadores universais antes do existencial: substitui-se x por função das variáveis quantificadas universalmente
  - $\forall x\ \exists y\ Pessoa(x) \Rightarrow Coração(y) \land Tem(x, y)$
    - todas as pessoas têm coração
  - substituindo y por constante:
    - $\forall x\ Pessoa(x) \Rightarrow Coração(C) \land Tem(x, C)$
    - Errado: toda a gente tem o mesmo coração C!
  - $\forall x\ Pessoa(x) \Rightarrow Coração\big(F(x)\big) \land Tem\big(x, F(x)\big)$, onde F é o nome de uma função ainda não utilizada – **função de Skolem**

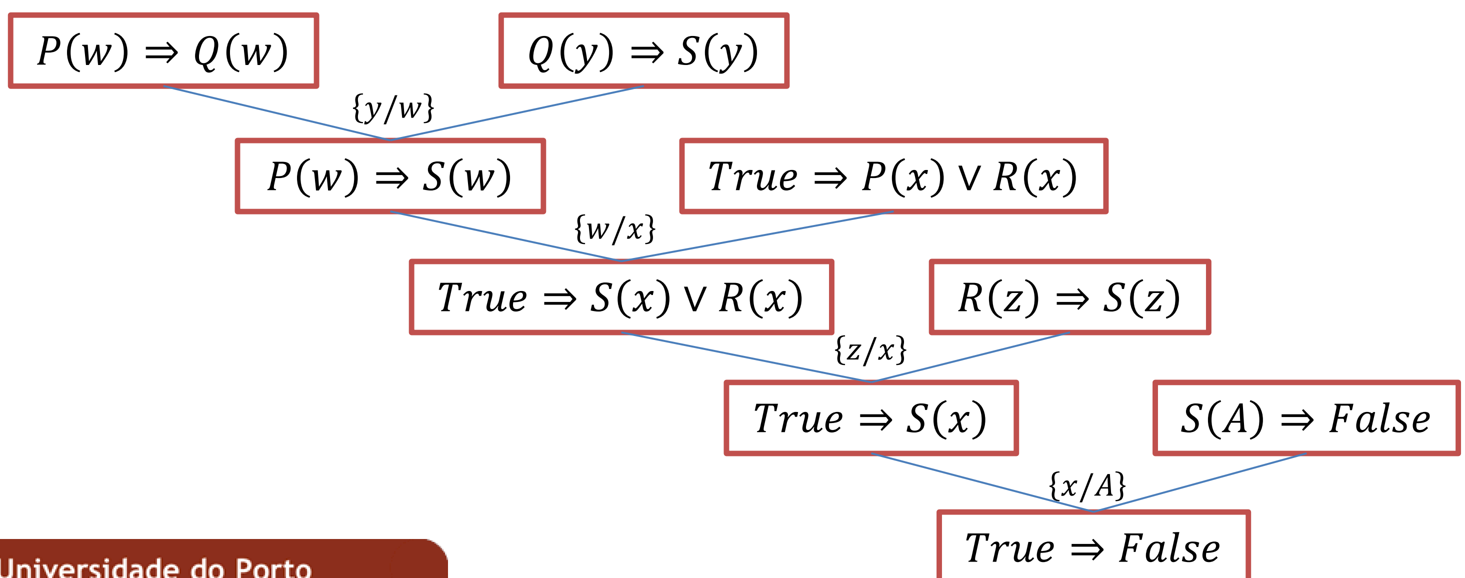
Provas com Resolução

- **Prova por contradição**: para provar P, assumir que P é falso (adicionar $\neg P$ à KB)
- Exemplo:

  C1: $\neg P(w) \lor Q(w) \quad \equiv P(w) \Rightarrow Q(w)$
  C2: $P(x) \lor R(x) \quad \equiv True \Rightarrow P(x) \lor R(x)$
  C3: $\neg Q(y) \lor S(y) \quad \equiv Q(y) \Rightarrow S(y)$
  C4: $\neg R(z) \lor S(z) \quad \equiv R(z) \Rightarrow S(z)$

  – Provar $S(A)$:

  C5: $\neg S(A) \quad \equiv S(A) \Rightarrow False$



Plano

- Conhecimento
- Lógica proposicional: revisões
- Lógica de 1ª ordem
- Programação em Lógica

Programação em Lógica

Programa = conjunto de axiomas

Computação = prova construtiva de um objectivo a partir do programa

- Interpretação procedimental de uma cláusula de Horn (Kowalski)

A se B_1 e B_2 e … e B_n

  – Leitura declarativa: A é verdade se os B_i são verdade
  – Leitura procedimental: para resolver (executar) A, resolver (executar) B_1, B_2 e … e B_n

  – Interpretador da linguagem: prova usando o princípio da resolução, com o algoritmo de unificação

Origens do Prolog

- 1970's: Kowalski
  - Leitura procedimental de cláusulas de Horn
- 1970's: Colmerauer (Marselha)
  - Implementação de demonstrador de teoremas chamado Prolog (*Programmation en Logique*)
- 1978/79: Warren (Edimburgo)
  - Primeira implementação eficiente de Prolog
  - *Edinburgh Prolog*: standard *de facto*
- 1981: Projecto Japonês da Quinta Geração de Computadores
  - Construir máquinas capazes de executar directamente Prolog