Lógica de Primeira Ordem

Lógica Computacional

Carlos Bacelar Almeida

Departmento de Informática
Universidade do Minho

2007/2008

Lógica de Primeira Ordem

- A *Lógica de Primeira Ordem* (ou *Lógica de Predicados*) aumenta o poder expressivo da linguagem ao permitir associar as *asserções lógica* às propriedades de objectos de um determinado domínio.
- Distinguem-se duas classes sintácticas: a dos **termos** e a das **fórmulas**.
- Os *termos* denotam os vários objectos de discurso.
  - Como caso particular desses termos temos as **variáveis**, que denotam objectos não especificados ou *indeterminados*. À partida, o seu significado é atribuído por uma dada *valoração*, que determina o objecto associado a cada variável. Utilizaremos as letras maiúsculas A, B, ... para denotar variáveis.
  - Os restantes constituintes dos termos são as **constantes** (denotam objectos fixos e determinados) e as **funções** (que denotam transformações sobre os objectos). Utilizaremos letras minúsculas ou palavras para denotar constantes e funções (e.g. a, $f(-), \mathrm{suc}(-), ...$)

- As *fórmulas* denotam as asserções (factos) sobre os termos.
  - Na sua forma mais simples são **predicados atómicos** (ou simplesmente *átomos*). Serão denotados por letras maiúsculas ou palavras iniciadas por letras maiúsculas (e.g., $Maior(X, Y)$, $EPar(X + 1)$, ...)
  - As **conectivas lógicas** proposicionais permitem-nos compor fórmulas estruturadas (e.g. $EPar(X) \Rightarrow \neg EPar(x + 1)$)
  - Os **quantificadores** permitem-nos considerar o conjunto de possibilidades para as valorações de uma variável (e.g. $\forall X.EPar(X) \vee EPar(X + 1)$). Uma vez quantificada, uma variável deixa de representar uma objecto indeterminado (dizemos que as ocorrências da variável que se encontram no âmbito do quantificador se encontram *ligadas* a esse quantificador).

Sintaxe

- A sintaxe da lógica de primeira ordem é parametrizada pelo tuplo $\langle \Sigma, \mathcal{R} \rangle$, onde:
  - $\Sigma = \langle S, \mathrm{arS} \rangle$ é a assinatura de termos. S é um conjunto de símbolos de funções e $\mathrm{arS} : S \rightarrow \mathbb{N}$ é a função de *aridade* que, a cada símbolo S, associa o número de argumentos da respectiva função (0 para as constantes).
  - $\mathcal{R} = \langle R, \mathrm{arR} \rangle$ é a assinatura de predicados. R é um conjunto de símbolos de predicados e $\mathrm{arR} : R \rightarrow \mathbb{N}$ associa o número de argumentos a cada predicado.
- Os termos ($\mathcal{T}$) são definidos pelas regras:
  - Uma variável é um termo;
  - Seja $f \in S$ com $\mathrm{arS}(f) = n$. Se $t_1, \ldots, t_n$ forem termos então $f(t_1, \ldots, t_n)$ é um termo.
- As fórmulas ($\mathcal{L}$) são definidas por:
  - Seja $P \in R$ com $\mathrm{arR}(P) = n$. Se $t_1, \ldots, t_n$ forem termos então $P(t_1, \ldots, t_n)$ é uma fórmula;
  - Se φ e ψ forem fórmulas, então $\bot$, $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \Rightarrow \psi$ são fórmulas;
  - Se φ for uma fórmula e X uma variável, então $\forall X.\varphi$ e $\exists X.\varphi$ são fórmulas

Variáveis e Substituições

- Termos que não contenham ocorrência de variáveis dizem-se **termos base**. O conjunto de termos base é denotado por $\mathcal{T}_B$.
- As ocorrências das variáveis dizem-se **livres** ou **ligadas**, dependendo se se encontram (ou não) no âmbito de um quantificador. Os conjuntos das variáveis livres e ligadas são denotados por $FV(-)$ e $BV(-)$ respectivamente.
- A operação de **substituição** de uma variável (livre) X por um termo t é denotada por $[t/X]$ (lê-se "t substitui X").
- A aplicação da substituição a um termo é definida por recursividade estrutural:

$$Y[t/X] = \begin{cases} t & , \text{ se } X = Y \\ Y & , \text{ se } X \neq Y \end{cases}$$
$$f(t_1, \ldots, t_n)[t/X] = f(t_1[t/X], \ldots, t_n[t/X])$$

- O processo de substituição extende-se às fórmulas:

$$P(t_1, \ldots, t_n)[t/X] = P(t_1[t/X], \ldots, t_n[t/X])$$
$$(\neg\varphi)[t/X] = \neg(\varphi[t/X])$$
$$(QY.\varphi)[t/X] = \begin{cases} QY.\varphi & , \text{ se } X = Y \\ QY.(\varphi[t/X]) & , \text{ se } X \neq Y \end{cases}$$

  onde Q denota um quantificador ($\forall$ ou $\exists$).
- Note-se que a substituição nunca afecta as variáveis ligadas.
- De facto, as variáveis ligadas podem ser renomeadas por *variáveis novas*[1] sem se alterar o significado das fórmulas. Assim,

$$\forall X.\varphi \equiv \forall Y.(\varphi[Y/X]) \qquad , \text{quando } Y \notin FV(\varphi)$$

[1] Uma **variável nova** é uma variável que não é utilizada no contexto em questão (e.g. na fórmula considerada).

Interpretação

Um **modelo** para a lógica de primeira ordem é constituído por:

- Uma estrutura de interpretação $\mathcal{I}$ que dará significado aos objectos (termos) envolvidos nas fórmulas. Será constituída por:
  - Um conjunto I que constitui o *domínio de interpretação*;
  - Para cada função $f \in \Sigma$ (com aridade n), uma função $\mathcal{I}(f) : I^n \to I$;
  - Para cada predicado $P \in \mathcal{R}$ (com aridade n), uma relação n-ária $\mathcal{I}(P) \subseteq I^n$.
- Uma valoração $\rho : V \to I$ que atribua significado às variáveis (livres) envolvidas.

Uma vez fixado uma interpretação $\mathcal{I}$ e uma valoração ρ, podemos atribuir significado ao conjunto de termos (i.e. associar a cada termo da linguagem um elemento do domínio de interpretação):

$$\mathcal{I}^\rho(x) = \rho(x)$$
$$\mathcal{I}^\rho(f(t_1, \ldots, t_n) = \mathcal{I}(f)(\mathcal{I}^\rho(t_1), \ldots, \mathcal{I}^\rho(t_n))$$

Validade

- A validade de uma fórmula lógica num modelo é definida recursivamente:

$\mathcal{I}\rho \models P(t_1, \ldots, t_n)$ sse $(\mathcal{I}^\rho(t_1), \ldots, \mathcal{I}^\rho(t_n)) \in \mathcal{I}(P)$

$\mathcal{I}\rho \models \neg\varphi$ sse $\mathcal{I}\rho \not\models \varphi$

$\mathcal{I}\rho \models \varphi \wedge \psi$ sse $\mathcal{I}\rho \models \varphi$ e $\mathcal{I}\rho \models \psi$

$\mathcal{I}\rho \models \varphi \vee \psi$ sse $\mathcal{I}\rho \models \varphi$ ou $\mathcal{I}\rho \models \psi$

$\mathcal{I}\rho \models \varphi \Rightarrow \psi$ sse $\mathcal{I}\rho \not\models \varphi$ ou $\mathcal{I}\rho \models \psi$

$\mathcal{I}\rho \models \forall X.\varphi$ sse para qualquer $t \in I$ temos que $\mathcal{I}\rho \models \varphi[t/X]$

$\mathcal{I}\rho \models \exists X.\varphi$ sse existir um $t \in I$ tal que $\mathcal{I}\rho \models \varphi[t/X]$

- Uma fórmula diz-se **válida** quando qualquer modelo a validar, i.e.

$$\models \varphi \quad \text{sse} \quad \text{para qualquer } \mathcal{I} \text{ e } \rho, \mathcal{I}\rho \models \varphi$$

- Um conjunto de fórmulas diz-se **uma teoria**. A noção de validade extende-se a teorias da forma óbvia.
- Dizemos que uma fórmula φ é **consequência semântica** de uma teoria Γ (escreve-se $\Gamma \models \varphi$) quando qualquer modelo que valide Γ valide também φ.
- Uma teoria Γ é **inconsistente** (escreve-se $\Gamma \models \bot$) quando não existe qualquer modelo que valide Γ.

Proposição

Para quaisquer teorias Γ, Δ *e fórmulas* φ *temos:*

- $\{\varphi, \neg\varphi\} \models \bot$
- *Se* $\Gamma \models \bot$ *e* $\Delta \supseteq \Gamma$*, então* $\Delta \models \bot$
- $\Gamma \models \varphi$ *se e só se* $\Gamma \cup \{\neg\varphi\} \models \bot$

Demonstração.

(Detalhes apresentados na aula teórica) □

Prenex Normal Form

- A sintaxe da lógica de primeira ordem permite-nos construir fórmulas com quantificadores em qualquer nível da árvore sintáctica (e.g. $\forall x.P(x) \vee \exists x.Q(x)$).
- Os quantificadores podem ser "puxados" para o exterior da fórmula sem alterar o seu significado **desde que nesse processo não se capture ocorrências de variáveis livres**. Por exemplo:
  - $\forall x.(P(x,y) \Rightarrow \exists z.Q(x,z)) \equiv \forall x.\exists z.(P(x,y) \Rightarrow Q(x,z))$
  - $\forall x.(P(x,y) \Rightarrow \exists y.Q(x,y)) \not\equiv \forall x.\exists y.(P(x,\mathbf{y}) \Rightarrow Q(x,y))$
    (destaca-se a negrito a ocorrência da variável y capturada pela movimentação do quantificador).
- A captura de variáveis pode sempre ser evitada por renomeação apropriada das variáveis ligadas. No exemplo apresentado, o lado esquerdo do segundo caso pode ser transformado no lado esquerdo do primeiro.

Prenex Normal Form

Definição

*Uma fórmula da lógica de primeira ordem diz-se na **forma normal Prenex** quando todos os quantificadores ocorrem no exterior da fórmula.*

- Operacionalmente, podemos obter a forma normal Prenex aplicando sucessivamente as seguintes equivalências (facilmente demonstráveis pela definição de validade):
  - $\neg\forall x.\varphi \equiv \exists x.\neg\varphi$
  - $\neg\exists x.\varphi \equiv \forall x.\neg\varphi$
  - $(\forall x.\psi) \otimes \varphi \equiv \forall x'.(\psi[x'/x] \otimes \varphi)$
  - $(\exists x.\psi) \otimes \varphi \equiv \exists x'.(\psi[x'/x] \otimes \varphi)$
  - $\varphi \otimes (\forall x.\psi) \equiv \forall x'.(\varphi \otimes \psi[x'/x])$
  - $\varphi \otimes (\exists x.\psi) \equiv \exists x'.(\varphi \otimes \psi[x'/x])$

  onde $x' \notin FV(\varphi)$ e $\otimes$ é um qualquer conectivo binário da lógica proposicional.

Skolenização/Herbranização

- Quando o objectivo é verificar se uma dada fórmula é *válida* (ou uma *contradição*), podemos ainda ir mais longe no tratamento dos quantificadores.
- Uma fórmula na forma prenex diz uma **fórmula existencial** quando só contém quantificadores existenciais. De forma análoga, uma **fórmula universal** só dispõe de quantificadores universais.
- Dado uma fórmula φ, podemos sempre encontrar uma fórmula existencial ψ tal que

  $$\models \varphi \qquad \text{sse} \qquad \models \psi$$

  O processo de obtenção de ψ a partir de φ designa-se por **Herbranização**.
- De forma análoga, dada uma fórmula φ, podemos encontrar uma fórmula de ψ tal que

  $$\varphi \text{ contradição} \qquad \text{sse} \qquad \psi \text{ contradição}$$

  O processo de obtenção de ψ a partir de φ designa-se por **Skolenização**.

Herbranização

- A ideia base na *Herbranização* consiste em observar que, na noção de validade, está implícita uma quantificação universal (sobre todos os modelos).
- Considere-se as fórmulas $\forall X.\varphi$ e $\varphi[c/X]$ onde c é uma nova constante. A segunda será válida quando, para qualquer interpretação, $\varphi[c/X]$ for válida. Mas a constante c pode ser interpretada em qualquer valor do domínio de interpretação. Verificamos assim que a segunda fórmula será válida precisamente quando a primeira o for.
- Com fórmulas da forma $\exists X.\forall Y.\varphi$, a substituição da variável Y pode depender da escolha efectuada para X. Nesse caso, o argumento apresentado acima deve ser formulado com novo símbolo de função f que depende de X, i.e. $\exists X.\varphi[f(X)/Y]$ será válida quando $\exists X.\forall Y.\varphi$ o for.

Herbranização

Definição

Dado uma fórmula na forma prenex, o processo de ***Herbranização*** *resulta da aplicação repetida da seguinte transformação que remove o primeiro dos quantificadores universais:*

$$\exists X_1 ... \exists X_n.\forall Y.\varphi \quad \rightsquigarrow \quad \exists X_1 ... \exists X_n.\varphi[f(X_1,\ldots,X_n)/Y]$$

onde f é um novo símbolo de função.

- Note que o processo de Herbranização não produzem fórmulas equivalentes às originais — de resto, elas nem sequer estão definidas sobre a mesma assinatura. O que se preserva é o *status de validade*.

Skolenização

- De forma dual, na *Skolenização*, o objectivo é encontrar uma fórmula que seja uma contradição se a original o for. Para tal, as substituições afectarão as variáveis quantificadas existencialmente.

Definição

Dado uma fórmula na forma prenex, o processo de ***Skolenização*** *resulta da aplicação repetida da seguinte transformação que remove o primeiro dos quantificadores existenciais:*

$$\forall X_1 ... \forall X_n . \exists Y . \varphi \quad \rightsquigarrow \quad \forall X_1 ... \forall X_n . \varphi[f(X_1, \ldots, X_n)/Y]$$

onde f é um novo símbolo de função.

Modelos de Herbrand

- Para aferir a validade (ou contradição) de uma fórmula, podemos nos restringir a uma classe de *modelos canónicos*.
- Esses modelos são definidos a partir da sintaxe da linguagem, e designam-se genericamente por **Modelos de Herbrand**.

Definição

Uma ***interpretação de Herbrand*** $\mathcal{H}$ *é um conjunto de átomos fechados H (i.e. predicados atómicos aplicados a termos base). A estrutura de interpretação é definido da seguinte forma:*

- *O domínio de interpretação é constituído pelo conjunto de termos base* $\mathcal{T}_B$*;*
- *A interpretação da função* $\mathcal{H}(f)$ *(com aridade n) consiste na função que, dados os termos base* $(t_1, \ldots, t_n)$ *retorna o termo base* $f(t_1, \ldots, t_n)$*.*
- *A interpretação dos predicados* $\mathcal{H}(P)$ *(com aridade n) consiste na relação* $\{(t_1, \ldots, t_n) | P(t_1, \ldots, t_n) \in H\}$

- Intuitivamente, uma interpretação de Herbrand é constituído por todos os átomos verdadeiros.
- Dado que só estão envolvidos termos base (sem variáveis), os modelos de Herbrand acabam por se aproximar dos modelos proposicionais.

Teorema

Uma fórmula existencial φ é válida se e só se qualquer interpretação de Herbrand validar φ.

Demonstração.

(Detalhes apresentados na aula teórica) □

- Note que a fórmula existencial pode ser obtida por Herbranização de uma qualquer fórmula.
- Dada a def. de validade de fórmulas existenciais, devem existir no modelo de Herbrand "instâncias base" que testemunhem a existência das variáveis envolvidas.

Sistemas Dedutivos

- Os sistemas dedutivos estudados para a lógica proposicional (dedução natural e cálculo de sequentes) podem ser extendidos para a lógica de primeira ordem.
- As novas regras lidam com os quantificadores — na dedução natural consideramos regras de introdução e eliminação dos quantificadores exitencial e universal; no cálculo de sequentes consideramos regras de introdução à direita e esquerda de cada quantificador.
- Vamos considerar unicamente o sistema de *Cálculo de Sequentes*.

Cálculo de Sequentes

$$\text{(Corte)}\ \frac{\Gamma, C \vdash \Delta \qquad \Gamma' \vdash C, \Delta'}{\Gamma, \Gamma' \vdash \Delta, \Delta'} \qquad \text{(Ax)}\ \frac{}{\Gamma, A \vdash A, \Delta}$$

$$\text{(D}\wedge\text{)}\ \frac{\Gamma \vdash P, \Delta \qquad \Gamma \vdash Q, \Delta}{\Gamma \vdash P \wedge Q, \Delta} \qquad \text{(D}\neg\text{)}\ \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash \neg P, \Delta}$$

$$\text{(D}\vee\text{)}\ \frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash P \vee Q, \Delta} \qquad \text{(D}\Rightarrow\text{)}\ \frac{\Gamma, P \vdash Q, \Delta}{\Gamma \vdash P \Rightarrow Q, \Delta}$$

$$\text{(E}\wedge\text{)}\ \frac{\Gamma, P, Q \vdash \Delta}{\Gamma, P \wedge Q \vdash \Delta} \qquad \text{(E}\neg\text{)}\ \frac{\Gamma \vdash P, \Delta}{\Gamma, \neg P \vdash \Delta}$$

$$\text{(E}\vee\text{)}\ \frac{\Gamma, P \vdash \Delta \qquad \Gamma, Q \vdash \Delta}{\Gamma, P \vee Q \vdash \Delta} \qquad \text{(E}\Rightarrow\text{)}\ \frac{\Gamma, Q \vdash \Delta \qquad \Gamma \vdash P, \Delta}{\Gamma, P \Rightarrow Q \vdash \Delta}$$

$$\text{(E}\forall\text{)}\ \frac{\Gamma, \forall X.\varphi, \varphi[t/X] \vdash \Delta}{\Gamma, \forall X.\varphi \vdash \Delta} \qquad \text{(E}\exists\text{)}\ \frac{\Gamma, \varphi[A/X] \vdash \Delta}{\Gamma, \exists X.\varphi \vdash \Delta}\ \text{sendo } A \text{ uma var. nova}$$

$$\text{(D}\forall\text{)}\ \frac{\Gamma \vdash \varphi[A/X], \Delta}{\Gamma \vdash \forall X.\varphi, \Delta}\ \text{sendo } A \text{ uma var. nova} \qquad \text{(D}\exists\text{)}\ \frac{\Gamma \vdash \exists X.\varphi, \varphi[t/X], \Delta}{\Gamma \vdash \exists X.\varphi, \Delta}$$

Propriedades

- Tal como no cálculo de sequentes proposicional, também aqui é possível **eliminar o corte** na apresentação do sistema dedutivo. A demonstração deste resultado é análogo à versão proposicional (faltando unicamente os casos relativos às novas regras. Ver detalhes em *Proof Theory and Automated Deduction*, Sec. 5.3).
- As regras ($D\ \exists$) e ($E\ \forall$) mantém a fórmula principal na premissa. Desta forma podemos instanciar a variável quantificada em diferentes termos.
- Nas regras ($E\ \exists$) e ($D\ \forall$) são introduzidas *variáveis novas*. Este facto determina que a ordem de aplicação das regras deixa de ser irrelevante...

A sua justificação (correcção) para as regras dos quantificadores resulta da seguinte observação:

Proposição

Para qualquer teoria Γ*, fórmula* φ*, termo* t *e variável nova* A*, temos:*

- $\{\Gamma, \forall X.\varphi, \varphi[t/X]\}$ *é inconsistente se e só se* $\{\Gamma, \forall X.\varphi\}$ *é inconsistente.*
- $\{\Gamma, \varphi[A/X]\}$ *é inconsistente se e só se* $\{\Gamma, \exists X.\varphi\}$ *é inconsistente.*
- $\{\Gamma, \neg(\exists X.\varphi), \neg(\varphi[t/X])\}$ *é inconsistente se e só se* $\{\Gamma, \neg(\exists X.\varphi)\}$ *é inconsistente.*
- $\{\Gamma, \neg(\varphi[A/X])\}$ *é inconsistente se e só se* $\{\Gamma, \neg(\forall X.\varphi)\}$ *é inconsistente.*

Demonstração.

(Detalhes apresentados na aula teórica.) □

- O Sistema Dedutivo *LC* é correcto e completo, i.e.

$$\models \varphi \qquad \text{sse} \qquad \vdash \varphi$$

- Mas note-se que este resultado diz-nos que, para qualquer fórmula válida, existe uma derivação no cálculo de sequentes. Não nos diz como obter essa derivação...
- De facto, ao contrário do cálculo de sequentes proposicional, na lógica de primeira ordem *não é possível utilizar directamente as regras do cálculo de sequentes como um procedimento de verificação da validade das fórmula*.
- Em rigor, não poderia ser de outra forma — **a verificação da validade em lógica de primeira ordem é indecidível**.

Verificação da Inconsistência em LPO

- Verificar a inconsistência de uma fórmula reduz-se, pelo processo de Skolenização, a verificar a inconsistência de uma fórmula universal, i.e. da forma

$$\forall X_1, \ldots, X_n . \varphi$$

  onde φ é uma fórmula livre de quantificadores.
- Desta forma, φ está próxima de uma fórmula proposicional. Em particular, pode ser expressa numa dada forma normal (e.g. na FNC).
- É por isso natural aplicar métodos estudados em lógica proposicional para estabelecer a inconsistência de φ. Em particular, poderemos utilizar *resolução* para procurar derivar $\perp$ da forma clausal de φ.
- É evidente que φ contém, em geral, variáveis. Este facto irá colocar desafios na generalização do método à lógica de primeira ordem,

Resolução Proposicional em LPO

- Comecemos por analisar a aplicação da *resolução proposicional* em fórmulas de primeira ordem da forma

$$\forall X_1, \ldots, X_n . \varphi$$

  onde φ se encontra na FNC.
- O teorema de Herbrand garante-nos a existência de instâncias base que estabelecem a inconsistência pretendida.
- Considerem-se cláusulas ψ_1 e ψ_2 e *substituições base* θ_1 e θ_2 tais que

$$\psi_1\theta_1 = A \vee \Gamma$$
$$\psi_2\theta_2 = \neg A \vee \Delta$$

  A resolução proposicional de $(A \vee \Gamma)$ e $(\neg A \vee \Delta)$ permite obter a cláusula $(\Gamma \vee \Delta)$.
- Note que, se Γ e Δ forem cláusulas vazia, é derivado o absurdo $\perp$ e, como tal, estabelecida a inconsistência de φ.

- A completude do método de resolução para lógica proposicional, e o teorema de Herbrand, garantem-nos que para qualquer forma universal inconsistente podemos derivar $\perp$ aplicando a resolução proposicional como apresentada atrás.
- O problema é que o método não é, de forma alguma, *efectivo* — teremos sempre de "adivinhar" as substituições base apropriadas para aplicar a resolução proposicional.
- O que seria desejável era uma forma de aplicar a resolução *directamente sobre as cláusulas de* φ.

Igualização de Termos com Variáveis

- Na aplicação da resolução proposicional em fórmulas de primeria ordem, são normalmente as substituições de base aplicadas que igualam os literais que intervém na resolução.
- Podemo-nos perguntar se não é possível igualar os literais sem ter necessidade de usar substituições de base (i.e. ter de concretizar todas as variáveis).
- Ilustremos o argumento com os termos $t_1 = f(X)$ e $t_2 = f(g(Y))$ — existe uma infinidade de substituições de base que igualam t_1 e t_2 (e.g., assumindo a existência de uma constante a, temos $[g(a)/X, a/Y]$, $[g(g(a))/X, g(a)/Y]$, etc.). A questão que se coloca é se não existe uma única solução que, de alguma forma, represente todas as restantes soluções?
- Essa solução "mais geral" de facto existe: considere-se a substituição $[g(Y)/X]$ — esta substituição iguala ambos os termos (em $f(g(Y))$), e deixa ainda liberdade para substituir Y pelo termo que se entender. De facto, qualquer substituição que iguale t_1 e t_2 é uma instância da solução apresentada.

Unificação

- A **unificação** consiste no problema de determinar (caso exista) a substituição mais geral que iguale dois termos. Essa substituição é designada por **unificador mais geral *(mgu)*** desses termos.
- Para definir o que se entende por "substituição mais geral", torna-se necessário estabelecer uma ordem (de especialização) nas substituições.

Definição (Ordem de Especialização)

Sejam σ e θ substituições. Dizemos que σ é mais geral do que θ ($\sigma \leq \theta$) quando existe uma substituição η tal que $\theta = \sigma \cdot \eta$.

Definição

*O **Unificador Mais Geral (mgu)** de termos t_1 e t_2 consiste numa substituição θ tal que $t_1\theta = t_2\theta$ e, para qualquer substituição σ para a qual $t_1\sigma = t_2\sigma$, temos que $\theta \leq \sigma$. Não existindo qualquer substituição que iguale os termos, convenciona-se que $mgu(t_1, t_2) = \bot$.*

Algoritmo de Unificação

- Podemos determinar o unificador mais geral de dois termos manipulando um sistema de equações.
- O estado do sistema é composto por um par $\langle \Gamma | \theta \rangle$ onde Γ é um conjunto de equações entre termos, e θ é a substituição calculada.
- Como estado inicial temos uma única equação entre os termos que pretendemos unificar. A substituição inicial é a identidade.
- O algoritmo prossegue aplicando as seguintes regras:

trivial	$\langle \Gamma, t = t \mid \theta \rangle \rightsquigarrow \langle \Gamma, \theta \rangle$
decompose	$\langle \Gamma, f(t_1, .., t_n) = f(s_1, .., s_n) \mid \theta \rangle \rightsquigarrow \langle \Gamma, t_1 = s_1, .., t_n = s_n \mid \theta \rangle$
clash	$\langle \Gamma, f(\ldots) = g(\ldots) \mid \theta \rangle \rightsquigarrow \bot$, se $f \neq g$
orient	$\langle \Gamma, t = x \mid \theta \rangle \rightsquigarrow \langle \Gamma, x = t \mid \theta \rangle$, se t não é uma variável
occurs	$\langle \Gamma, x = t \mid \theta \rangle \rightsquigarrow \bot$, se $x \in \mathrm{Vars}(t)$
variable	$\langle \Gamma, x = t \mid \theta \rangle \rightsquigarrow \langle \Gamma[t/x] \mid \theta \cdot [t/x] \rangle$, se $x \neq t$

Codificação em *Haskell*

...retirado de um exemplo do interpretador *hugs*.

```
--- Prolog Terms:
type Id       = (Int,String)
type Atom     = String
data Term     = Var Id | Struct Atom [Term]

--- Substitutions:
-- substitutions are represented by functions mapping identifiers to terms.
type Subst = Id -> Term

-- app s     extends the substitution s to a function mapping terms to terms
app                    :: Subst -> Term -> Term
app s (Var i)           = s i
app s (Struct a ts)     = Struct a (map (app s) ts)

-- nullSubst is the empty substitution which maps every identifier to the
--           same identifier (as a term).
nullSubst              :: Subst
nullSubst i             = Var i

infix  4 ->-
-- i ->- t   is the substitution which maps the identifier i to the term t,
--           but otherwise behaves like nullSubst.
(->-)                  :: Id -> Term -> Subst
(i ->- t) j | j==i      = t
            | otherwise = Var j

infixr 3 @@
-- s1@@ s2  is the composition of substitutions s1 and s2
(@@)                   :: Subst -> Subst -> Subst
s1 @@ s2                = app s1 . s2
```



```
--- Unification:

-- unify t1 t2 returns a list containing a single substitution s which is
--             the most general unifier of terms t1 t2.  If no unifier
--             exists, the list returned is empty.

unify :: Term -> Term -> [Subst]
unify (Var x)        (Var y)        = if x==y
                                        then [nullSubst]
                                        else [x->-Var y]
unify (Var x)        t2             = [ x ->- t2 | x `notElem` varsIn t2 ]
unify t1             (Var y)        = [ y ->- t1 | y `notElem` varsIn t1 ]
unify (Struct a ts)  (Struct b ss)  = [ u | a==b, u<-listUnify ts ss ]

listUnify :: [Term] -> [Term] -> [Subst]
listUnify []     []     = [nullSubst]
listUnify []     (r:rs) = []
listUnify (t:ts) []     = []
listUnify (t:ts) (r:rs) = [ u2 @@ u1 | u1<-unify t r,
                                       u2<-listUnify (map (app u1) ts)
                                                     (map (app u1) rs) ]
```

Resolução de Primeira Ordem

Definição

Sejam φ, φ' e $\neg\psi, \psi'$ cláusulas de primeira ordem que não contenham variáveis em comum. O ***resolvente*** *dessas cláusulas consiste na cláusula $\varphi'\theta, \psi'\theta$ quando $\theta = mgu(\varphi, \psi)$.*

- O requisito de as cláusulas não conterem variáveis em comum pode facilmente ser satisfeito por renomeação das variáveis de uma das cláusulas.
- φ e ψ são normalmente literais. No entanto podem ser também conjuntos não vazios de literais[2] que são todos unificados num só.
- φ' e ψ' são conjuntos de literais (possivelmente vazios). Quando ambos são vazios obtém-se $\perp$ como resolvente.

[2] Nesse caso, $\neg\psi$ denota um conjunto de literais negativos.

Cláusulas e Forma Condicional

- Uma cláusula da FNC pode sempre ser vista na **Forma Condicional** como uma implicação entre uma conjunção e uma disjunção de literais positivos:

  $$\neg A \vee B \vee \neg C \vee D \equiv \neg A \vee \neg C \vee B \vee D \equiv \neg(A \wedge C) \vee (B \vee D) \equiv (A \wedge C) \Rightarrow (B \vee D)$$

- É habitual apresentar-se uma cláusula na forma condicional como uma *regra*:

  $$B \vee D \leftarrow A \wedge C$$

  ou simplesmente $B, D \leftarrow A, C$ (deixando implícito as conectivas lógicas).
- As **Cláusulas de *Horn*** restringem o número de literais positivos contidos na cláusulas da FNC.

Cláusulas de Horn

Definição

*Uma cláusula da FNC diz-se uma **Cláusula de Horn** quando dispõe de, quanto muito, um literal positivo.*

- Podemos classificar as cláusulas de Horn como:
  - Questão: $\leftarrow \Gamma$. (ou Objectivo, ou *Goal*...)
  - Regra: $H \leftarrow \Gamma$.
  - Facto: F.
- No *Prolog*, o símbolo $\leftarrow$ é substituído pelo operador `:-`.

```
member(H, [H|T]).
member(X, [_|T]) :- member(X,T).
:- member(X, [1,2,3,4]), X mod 2 == 0.
```

- Quando nos restringimos ao fragmento lógico determinado pelas *teorias de Horn*, o problema de verificação da inconsistência dessas teorias passa a ser resolúvel.

Resolução Linear

- Admitamos que dispomos de uma **base de conhecimento** Γ (uma teoria) composta por factos e regras de *Horn*.
- Para verificar se uma dada conjunção de literais positivos G é consequência semântica da base de conhecimento (i.e. $\Gamma \models G$), é suficiente verificar se $\Gamma \cup (\leftarrow G)$ é uma teoria inconsistente (i.e. $\Gamma \cup (\leftarrow G) \vdash \bot$).
- Por outro lado, essa verificação é particularmente simples neste caso: basta-nos reescrever cada literal da questão G de acordo com as regras contidas em Γ.

Exemplo de Aplicação

- Considere a seguinte base de conhecimento:

```
a.
b :- d, e.
c :- d.
d :- e.
e.
f.
```

- Para avaliar o objectivo `:-a,b,c.` procede-se da seguinte forma (utilizamos o símbolo $\bowtie$ para significar *resolve com...*):
  1. $\neg a, \neg b, \neg c \quad \bowtie \quad a$
  2. $\neg b, \neg c \quad \bowtie \quad b, \neg d, \neg e$
  3. $\neg d, \neg e, \neg c \quad \bowtie \quad d, \neg e$
  4. $\neg e, \neg e, \neg c \quad \bowtie \quad e$ (2 vezes)
  5. $\neg c \quad \bowtie \quad c, \neg d$
  6. $\neg d \quad \bowtie \quad d, \neg e$
  7. $\neg e \quad \bowtie \quad e$
  8. YES

- O exemplo apresentado é meramente proposicional (não contém variáveis). No caso geral torna-se necessário realizar unificação do literal do objectivo com a cabeça da regra contida na base de conhecimento (devidamente renomeada, para evitar colisão de variáveis).
- Quando existe mais do que uma possibilidade para resolver um literal (e.g. várias regras são aplicáveis) torna-se necessário *ramificar* o processo de inferência.
- Operacionalmente, o *Prolog* trata esta ramificação pelo mecanismo de *backtracking*.
- As *árvores de inferência do Prolog* são uma forma de se explicitar o processo de resolução linear envolvido na verificação de um objectivo.

Árvore de Inferência do *Prolog*

Assumindo a definição usual do predicado `member/2`, a verificação do objectivo `member(X,[1,2]), X mod 2 == 0` resultaria em: