Programação em Lógica

Luís Cruz-Filipe

Maio de 2006

Conteúdo

Introdução

Estas folhas destinam-se a servir de apoio à disciplina de Programação em Lógica leccionada no Instituto Superior Técnico (Universidade Técnica de Lisboa) pela Secção de Lógica e Computação do Departamento de Matemática.

A Programação em Lógica é a concretização da ideia dar semântica operacional a um fragmento da Lógica de Primeira Ordem para obter conjuntos de fórmulas que são simultaneamente um programa executável (semântica operacional) e a sua especificação (semântica tradicional); tais programas são por natureza correctos no sentido em que satisfazem a sua especificação lógica. A semântica operacional baseia-se no mecanismo geral de resolução, introduzido por Robinson em 1965.

Organização

O Capítulo 1 faz uma revisão dos conceitos necessários de Lógica de Primeira Ordem e introduz algumas noções específicas da Programação em Lógica, em particular definindo cláusulas, programas determinados e o fragmento da Lógica de Primeira Ordem que se pretende estudar. Apresenta-se também a semântica tradicional da Lógica de Primeira Ordem e alguns resultados que permitem simplificar significativamente o estudo desta semântica quando aplicada apenas ao fragmento clausal.

No Capítulo 2 discute-se a noção de substituição, central à Programação em Lógica, e apresentam-se os conceitos de resposta, unificadora e unificadora mais geral (UMG). O capítulo termina com um algoritmo de unificação cuja correcção total se prova.

Neste ponto já é possível estudar a semântica operacional da Programação em Lógica, que é o tema do Capítulo 3. Aqui descreve-se pela primeira vez o mecanismo de resolução-SLD, provando-se a sua correcção, o resultado inesperado de Independência da Regra e a adequação computacional, que justifica o interesse prático desta semântica. Estes resultados são interpretados em termos práticos por via da noção de árvore-SLD.

O objectivo seguinte é provar a completude lógica do mecanismo de resolução-SLD. Isto é conseguido recorrendo a uma semântica denotacional da Programação em Lógica e da definição do modelo de Herbrand mínimo de um programa determinado. Este estudo é feito no Capítulo 5, após uma exposição introdutória dos pré-requisitos de teoria de reticulados que é feita no Capítulo 4.

A resolução-SLD é a base de todos os interpretadores de PROLOG, pelo que no Capítulo 5 se discutem também questões mais ligadas à implementação do mecanismo de *backtracking*. Esta discussão está ligada à completude da resolução-SLD, mas ver-se-á infelizmente que as implementações tradicionais têm algumas lacunas. Discute-se ainda brevemente o mecanismo do corte.

O Capítulo 6 estuda outra questão prática: a possibilidade de estender a linguagem da Programação em Lógica recorrendo à negação. A implementação intuitiva através da regra da Negação

por Falha dá origem ao mecanismo de resolução-SLDNF, cuja correcção se prova recorrendo à noção de programa completado. O capítulo termina com uma brevíssima exposição dos problemas relativos à completude deste mecanismo de resolução.

Agradecimentos

O material constante destas folhas é um enriquecimento substantial das Folhas de Programação Recursiva, escritas para a cadeira de (então) Programação Recursiva, hoje Programação em Lógica, por Amílcar Sernadas e Carlos Caleiro. Quer essas folhas quer estas seguem de perto a visão da Programação em Lógica transmitida no livro *Foundations of Logic Programming* de J.W. Lloyd, que desde sempre fez parte da bibliografia básica da disciplina.

O resultado final deve ainda muito ao Jaime Ramos, que leccionou a disciplina de Programação em Lógica durante vários anos e por essa via influenciou, indirecta mas significamente, a forma como a exposição aqui foi feita. Foi também fundamental a colaboração do Hélio Pais, cuja leitura atenta das várias versões preliminares permitiu corrigir vários erros e melhorar a qualidade do trabalho.

A responsabilidade por eventuais lapsos, omissões ou incorrecções reside, claro está, exclusivamente com o autor.

Capítulo 1

Fundamentos de Lógica de Primeira Ordem

A Programação em Lógica debruça-se sobre um fragmento criteriosamente escolhido da Lógica de Primeira Ordem. Neste capítulo apresenta-se a Lógica de Primeira Ordem e discutem-se as suas propriedades fundamentais; em simultâneo, define-se o fragmento da linguagem que será o objecto de estudo principal desta disciplina.

1.1 Sintaxe da Lógica de Primeira Ordem

Definição 1.1.1 Uma *assinatura de primeira ordem* é constituída por:

- símbolos de constante;
- símbolos de função;
- símbolos de predicado.

A cada símbolo de função e de predicado está associada uma *aridade* (número de argumentos que essa função ou predicado recebe).

Definição 1.1.2 Um *alfabeto de primeira ordem* é constituído por:

- uma assinatura de primeira ordem;
- um conjunto numerável X de variáveis;
- os conectivos $\neg$, $\wedge$, $\vee$, $\Rightarrow$ e $\Leftrightarrow$;
- os quantificadores $\forall$ e $\exists$;
- sinais de pontuação (parêntesis e vírgula).

Notação 1.1.3 Convenciona-se o uso dos seguintes símbolos:

- letras do final do alfabeto para variáveis (x, y, u, x_1, x');
- letras do início do alfabeto para símbolos de constante (a, b, a_1, a');
- as letras f, g, e h (ou variantes f_1, g') para símbolos de função;
- as letras p, q, e r (ou variantes p_1, q') para símbolos de predicado.

Definição 1.1.4 O conjunto dos *termos* sobre um alfabeto define-se indutivamente da seguinte forma.

- Toda a variável é um termo.
- Todo o símbolo de constante é um termo.
- Se $t_1, \ldots, t_n$ são termos e f é um símbolo de função de aridade n, então $f(t_1, \ldots, t_n)$ é um termo.

Termos arbitrários serão representados por t, t_1, t'.

Definição 1.1.5 O conjunto das *fórmulas* sobre um alfabeto define-se indutivamente da seguinte forma.

- Se $t_1, \ldots, t_n$ são termos e p é um símbolo de predicado de aridade n, então $p(t_1, \ldots, t_n)$ é uma fórmula dita *fórmula atómica* ou *átomo*.
- Se F e G são fórmulas, então $(\neg F)$, $(F \wedge G)$, $(F \vee G)$, $(F \Rightarrow G)$ e $(F \Leftrightarrow G)$ são fórmulas.
- Se F é uma fórmula e x é um símbolo de variável, então $(\forall x.F)$ e $(\exists x.F)$ são fórmulas.

Fórmulas arbitrárias serão representadas por F, G, F_1, G'.

Notação 1.1.6 Para evitar sobrecarregar as fórmulas de parêntesis, convenciona-se que os operadores $\neg$, $\forall$ e $\exists$ têm precedência sobre todos os outros; $\vee$ tem precedência sobre $\wedge$, $\Rightarrow$ e $\Leftrightarrow$; e $\wedge$ tem precedência sobre $\Rightarrow$ e $\Leftrightarrow$. Com esta convenção, a fórmula

$$(((\forall x.(p(x) \Rightarrow q(x))) \wedge (\forall x.p(x))) \Rightarrow (\forall x.q(x)))$$

pode ser escrita simplesmente como

$$\forall x.(p(x) \Rightarrow q(x)) \wedge \forall x.p(x) \Rightarrow \forall x.q(x)$$

Definição 1.1.7 O *alcance* de um quantificador $\forall x$ ou $\exists x$ numa (sub)fórmula $\forall x.F$ ou $\exists x.F$ é a subfórmula F.

Definição 1.1.8 Uma ocorrência de uma variável x numa fórmula F diz-se *muda* se estiver no alcance de um quantificador $\forall x$ ou $\exists x$ em F. Caso contrário, diz-se uma ocorrência *livre*.

Uma fórmula que não contém ocorrências livres de nenhuma variável diz-se uma *fórmula fechada*.

É importante observar que a mesma variável pode ocorrer livre e muda na mesma fórmula, como por exemplo x em $p(x) \wedge \forall x.q(x, y)$. No entanto, é boa política evitar escrever este tipo de fórmulas escrevendo por exemplo $p(x) \wedge \forall z.q(z, y)$.

Definição 1.1.9 Dada uma fórmula F, o seu *fecho universal* $\forall F$ é a fórmula que se obtém quantificando universalmente todas as variáveis que ocorrem livres em F. Analogamente, o *fecho existencial* $\exists F$ de F é a fórmula que se obtém quantificando existencialmente todas as variáveis que ocorrem livres em F.

Exemplo 1.1.10 Se F for a fórmula $p(a) \wedge \forall x.(p(x, y) \Rightarrow q(x, y, z))$, então os seus fechos universal e existencial são

$$\begin{aligned} \forall F &\equiv \forall y.\forall z.(p(a) \wedge \forall x.(p(x, y) \Rightarrow q(x, y, z))) \\ \exists F &\equiv \exists y.\exists z.(p(a) \wedge \forall x.(p(x, y) \Rightarrow q(x, y, z))) \end{aligned}$$

Se G for uma fórmula fechada, então $\forall G \equiv \exists G \equiv G$.

1.2 Sintaxe da Programação em Lógica

Conforme foi dito inicialmente, o objectivo desta disciplina é estudar um fragmento da Lógica de Primeira Ordem com boas propriedades computacionais. Nesta secção define-se formalmente este fragmento como subconjunto da linguagem de primeira ordem.

Definição 1.2.1 Um *literal* é uma fórmula atómica ou a negação duma fórmula atómica. Um átomo também se diz um literal *positivo*, enquanto a negação dum átomo se diz um literal *negativo*.

Utilizar-se-ão as letras A, B, A_1, B' para designar átomos e L, L_1, L' para representar literais.

Definição 1.2.2 Uma *cláusula* é uma fórmula da forma

$$\forall x_1 \ldots \forall x_n.(L_1 \vee \ldots \vee L_m)$$

em que $L_1, \ldots, L_m$ são literais (positivos ou negativos) e $x_1, \ldots, x_n$ são todas as variáveis que ocorrem (livres) em $L_1, \ldots, L_m$.

Note-se que por definição uma cláusula é sempre uma fórmula fechada. Para além disso, conhecendo os literais que compõem uma cláusula consegue-se reconstituir (a menos da ordem dos quantificadores) a cláusula original.

Observe-se ainda que se tem a equivalência

$$\begin{aligned} & \forall(A_1 \vee \ldots \vee A_p \vee \neg B_1 \vee \ldots \vee \neg B_q) \\ \longleftrightarrow\ & \forall((A_1 \vee \ldots \vee A_p) \vee \neg(B_1 \wedge \ldots \wedge B_q)) \\ \longleftrightarrow\ & \forall((B_1 \wedge \ldots \wedge B_q) \Rightarrow (A_1 \vee \ldots \vee A_p)) \end{aligned}$$

justificando-se assim a notação seguinte.

Notação 1.2.3 Denota-se por

$$A_1, \ldots, A_p \longleftarrow B_1, \ldots, B_q$$

a cláusula

$$\forall x_1 \ldots \forall x_n.(A_1 \vee \ldots \vee A_p \vee \neg B_1 \vee \ldots \vee \neg B_q).$$

Os seguintes tipos de cláusulas são de particular importância.

Definição 1.2.4 Uma cláusula diz-se:

- uma cláusula *determinada* se for da forma $A \longleftarrow B_1, \ldots, B_q$; neste caso, A diz-se a *cabeça* e $B_1, \ldots, B_q$ o *corpo* da cláusula;
- uma cláusula *unitária* ou um *facto* se for da forma $A \longleftarrow$;
- um *objectivo determinado* se for da forma $\longleftarrow B_1, \ldots, B_q$; cada B_i diz-se um sub-objectivo;
- uma *cláusula de Horn* se for uma cláusula determinada ou um objectivo determinado;
- a *cláusula vazia*, denotada por $\square$, se for a cláusula $\longleftarrow$.

Tal como atrás, é útil analisar em pormenor o significado dum objectivo determinado $\longleftarrow B_1, \ldots, B_q$.

$$\begin{aligned} &\longleftarrow B_1, \ldots, B_q \\ &\longleftrightarrow \forall(\neg B_1 \vee \ldots \vee \neg B_q) \\ &\longleftrightarrow \neg\exists(B_1 \wedge \ldots \wedge B_q) \end{aligned}$$

Ou seja, um objectivo determinado expressa o facto de não haver nenhuma instanciação das variáveis livres em $B_1, \ldots, B_q$ que torne todos estes literais verdadeiros em simultâneo.

Em particular, a cláusula vazia é um objectivo determinado que é uma contradição (já que a conjunção de zero literais é trivialmente satisfeita).

Definição 1.2.5 Um *programa determinado* é um conjunto finito de cláusulas determinadas.

Num programa determinado, o conjunto das cláusulas cuja cabeça é construída recorrendo ao mesmo símbolo de predicado p diz-se a *definição* de p.

Exemplo 1.2.6 Apresenta-se um exemplo de um programa determinado.

ordenação$(x, y) \longleftarrow$ ordenado(y), permutação(x, y)
ordenado$(NIL) \longleftarrow$ ordenado$(x.NIL) \longleftarrow$ ordenado$(x.y.z) \longleftarrow x \leq y$, ordenado$(y.z)$
permutação$(NIL, NIL) \longleftarrow$ permutação$(x, u.v) \longleftarrow$ retira(u, x, z), permutaçao(z, v)
retira$(x, x.y, y) \longleftarrow$ retira$(x, y.z, y.w) \longleftarrow$ retira(x, z, w)
$0 \leq x \longleftarrow$ suc$(x) \leq$ suc$(y) \longleftarrow x \leq y$

As cláusulas encontram-se agrupadas de acordo com os predicados que definem. Assim, a primeira cláusula contém a definição do predicado "ordenação", as três cláusulas seguintes contêm a definição do predicado "ordenado", e assim sucessivamente. Qual é o alfabeto de primeira ordem subjacente a este programa?

1.3 Semântica da Lógica de Primeira Ordem

Descreve-se agora em breves linhas a semântica usual da Lógica de Primeira Ordem. Na secção seguinte particularizar-se-á esta semântica ao fragmento clausal da linguagem, obtendo-se resultados fortes de caracterização.

Definição 1.3.1 Uma *estrutura de interpretação* para uma assinatura de primeira ordem é um par $J = \langle D, \cdot_J \rangle$ onde:

- D é um conjunto não vazio, dito o *domínio* da interpretação;
- $\cdot_J$ é um mapa que faz corresponder:
  - a cada constante c, um elemento $c_J \in D$;
  - a cada símbolo de função f de aridade n, uma aplicação $f_J : D^n \to D$;

– a cada símbolo de predicado p de aridade n, uma aplicação $p_J : D^n \to \{0, 1\}$.

Uma estrutura de interpretação é portanto um conjunto com um mapa que permite interpretar qualquer termo fechado como um elemento desse conjunto. Para poder tratar fórmulas (e em particular termos com variáveis livres) é preciso saber também como interpretar variáveis.

Definição 1.3.2 Seja J uma estrutura de interpretação para uma assinatura de primeira ordem. Uma *atribuição* (de valores em J) é uma aplicação $\rho : X \to D$ que faz corresponder a cada variável $x \in X$ um valor $\rho(x) \in D$.

Mais à frente serão introduzidas diversas operações sobre atribuições; neste ponto só é necessário definir, dados uma atribuição ρ, uma variável x e um valor $d \in D$, a atribuição $\rho\,[^x/_d]$.

$$\rho\,[^x/_d]\,(y) = \begin{cases} d & \text{se } y \text{ é } x \\ \rho(y) & \text{caso contrário} \end{cases}$$

Definição 1.3.3 Sejam J uma estrutura de interpretação para uma assinatura de primeira ordem e ρ uma atribuição de valores em J. A *interpretação* dos termos é um mapa $[\![\cdot]\!]_{J,\rho}$ definido indutivamente como se segue.

- $[\![x]\!]_{J,\rho} = \rho(x)$ para cada variável x.
- $[\![c]\!]_{J,\rho} = c_J$ para cada símbolo de constante c.
- $[\![f(t_1, \ldots, t_n)]\!]_{J,\rho} = f_J([\![t_1]\!]_{J,\rho}, \ldots, [\![t_n]\!]_{J,\rho})$ para cada símbolo de função f de aridade n e termos $t_1, \ldots, t_n$.

Definição 1.3.4 Sejam J uma estrutura de interpretação para uma assinatura de primeira ordem e ρ uma atribuição de valores em J. A *interpretação* das fórmulas é um mapa $[\![\cdot]\!]_{J,\rho}$ definido indutivamente como se segue.

- $[\![p(t_1, \ldots, t_n)]\!]_{J,\rho} = p_J([\![t_1]\!]_{J,\rho}, \ldots, [\![t_n]\!]_{J,\rho})$ para cada símbolo de predicado p de aridade n e termos $t_1, \ldots, t_n$.
- $[\![(\neg F)]\!]_{J,\rho} = 1 - [\![F]\!]_{J,\rho}$ (ou seja, $[\![(\neg F)]\!]_{J,\rho} = 1$ sse $[\![F]\!]_{J,\rho} = 0$).
- $[\![(F \wedge G)]\!]_{J,\rho} = 1$ sse $[\![F]\!]_{J,\rho} = 1$ e $[\![G]\!]_{J,\rho} = 1$.
- $[\![(F \vee G)]\!]_{J,\rho} = 1$ sse $[\![F]\!]_{J,\rho} = 1$ ou $[\![G]\!]_{J,\rho} = 1$.
- $[\![(F \Rightarrow G)]\!]_{J,\rho} = 1$ sse $[\![F]\!]_{J,\rho} = 0$ ou $[\![G]\!]_{J,\rho} = 1$.
- $[\![(F \Leftrightarrow G)]\!]_{J,\rho} = 1$ sse $[\![F]\!]_{J,\rho} = [\![G]\!]_{J,\rho}$.
- $[\![(\forall x.F)]\!]_{J,\rho} = 1$ sse $[\![F]\!]_{J,\rho[^x/_d]} = 1$ para qualquer $d \in D$.
- $[\![(\exists x.F)]\!]_{J,\rho} = 1$ sse $[\![F]\!]_{J,\rho[^x/_d]} = 1$ para algum $d \in D$.

É importante salientar que a interpretação de um *termo* é um valor do domínio D enquanto a interpretação duma *fórmula* é 0 ou 1.

Lema 1.3.5 *(Lema dos símbolos omissos em termo.)* Sejam J uma estrutura de interpretação, t um termo e ρ e σ duas atribuições tais que $\rho(x) = \sigma(x)$ para toda a variável x que ocorre em t. Então $[\![t]\!]_{J,\rho} = [\![t]\!]_{J,\sigma}$.

Prova. Por indução na estrutura do termo t.

- Se t é um símbolo de constante c, então $[\![c]\!]_{J,\rho} = c_J = [\![c]\!]_{J,\sigma}$ por definição de interpretação (para constantes).
- Se t é uma variável x, então $[\![x]\!]_{J,\rho} = \rho(x) = \sigma(x) = [\![x]\!]_{J,\sigma}$ por definição de interpretação (para variáveis) e usando a hipótese.
- Se t é um termo $f(t_1, \ldots, t_n)$, então por definição de interpretação tem-se $[\![f(t_1, \ldots, t_n)]\!]_{J,\rho} = f_J([\![t_1]\!]_{J,\rho}, \ldots, [\![t_n]\!]_{J,\rho}) \stackrel{*}{=} f_J([\![t_1]\!]_{J,\sigma}, \ldots, [\![t_n]\!]_{J,\sigma}) = [\![f(t_1, \ldots, t_n)]\!]_{J,\sigma}$, onde no passo $*$ se utilizou a hipótese de indução (aplicável uma vez que toda a variável de t_i ocorre necessariamente em t).

□

Lema 1.3.6 *(Lema dos símbolos omissos em fórmula.)* Sejam J uma estrutura de interpretação, F uma fórmula e ρ e σ duas atribuições tais que $\rho(x) = \sigma(x)$ para toda a variável x que ocorre livre em F. Então $[\![F]\!]_{J,\rho} = [\![F]\!]_{J,\sigma}$.

Prova. Por indução na estrutura da fórmula F.

- Se F é atómica, ou seja, da forma $p(t_1, \ldots, t_n)$, então por definição de interpretação tem-se $[\![p(t_1, \ldots, t_n)]\!]_{J,\rho} = p_J([\![t_1]\!]_{J,\rho}, \ldots, [\![t_n]\!]_{J,\rho}) \stackrel{*}{=} p_J([\![t_1]\!]_{J,\sigma}, \ldots, [\![t_n]\!]_{J,\sigma}) = [\![p(t_1, \ldots, t_n)]\!]_{J,\sigma}$, onde em $*$ se utilizou o lema anterior (aplicável já que toda a variável que ocorre em t_i está necessariamente livre em F).
- Se F é $(\neg G)$ então $[\![(\neg G)]\!]_{J,\rho} = 1$ sse (por definição de interpretação) $[\![G]\!]_{J,\rho} = 0$ sse (por hipótese de indução) $[\![G]\!]_{J,\sigma} = 0$ sse (por definição de interpretação) $[\![(\neg G)]\!]_{J,\sigma} = 1$.
- Se F é $(G \wedge H)$ então $[\![(G \wedge H)]\!]_{J,\rho} = 1$ sse (por definição de interpretação) $[\![G]\!]_{J,\rho} = 1$ e $[\![H]\!]_{J,\rho} = 1$ sse (por hipótese de indução) $[\![G]\!]_{J,\sigma} = 1$ e $[\![H]\!]_{J,\sigma} = 1$ sse (por definição de interpretação) $[\![(G \wedge H)]\!]_{J,\sigma} = 1$.
- Se F é $(G \vee H)$ então $[\![(G \vee H)]\!]_{J,\rho} = 0$ sse $[\![G]\!]_{J,\rho} = 0$ e $[\![H]\!]_{J,\rho} = 0$ sse $[\![G]\!]_{J,\sigma} = 0$ e $[\![H]\!]_{J,\sigma} = 0$ sse $[\![(G \vee H)]\!]_{J,\sigma} = 0$.
- Se F é $(G \Rightarrow H)$ então $[\![(G \Rightarrow H)]\!]_{J,\rho} = 0$ sse $[\![G]\!]_{J,\rho} = 1$ e $[\![H]\!]_{J,\rho} = 0$ sse $[\![G]\!]_{J,\sigma} = 1$ e $[\![H]\!]_{J,\sigma} = 0$ sse $[\![(G \Rightarrow H)]\!]_{J,\sigma} = 1$.
- Se F é $(G \Leftrightarrow H)$ então $[\![(G \Leftrightarrow H)]\!]_{J,\rho} = 1$ sse $[\![G]\!]_{J,\rho} = [\![H]\!]_{J,\rho}$ sse $[\![G]\!]_{J,\sigma} = [\![H]\!]_{J,\sigma}$ sse $[\![(G \Leftrightarrow H)]\!]_{J,\sigma} = 1$.
- Se F é $(\forall x.G)$ então $[\![(\forall x.G)]\!]_{J,\rho} = 1$ sse $[\![G]\!]_{J,\rho[x/d]} = 1$ para qualquer $d \in D$. Fixando d e denotando por ρ' a atribuição $\rho[x/d]$ e por σ' a atribuição $\sigma[x/d]$, a hipótese de indução é aplicável já que ρ' e σ' coincidem nas variáveis livres de G (coincidem nas de F por hipótese e coincidem em x por construção; é fácil verificar que são estas precisamente as variáveis livres de G), donde se conclui que $[\![G]\!]_{J,\rho'} = [\![G]\!]_{J,\sigma'}$. Como isto se verifica independentemente de d, conclui-se finalmente que $[\![(\forall x.G)]\!]_{J,\rho} = 1$ sse $[\![(\forall x.G)]\!]_{J,\sigma} = 1$.
- Se F é $(\exists x.G)$ então $[\![(\exists x.G)]\!]_{J,\rho} = 1$ sse $[\![G]\!]_{J,\rho[x/d]} = 1$ para algum $d \in D$. Fixando d e definindo ρ' e σ' como atrás, a hipótese de indução é novamente aplicável pelo mesmo raciocínio, donde se conclui analogamente que $[\![(\exists x.G)]\!]_{J,\rho} = 1$ sse $[\![(\exists x.G)]\!]_{J,\sigma} = 1$.

□

Corolário 1.3.7 Sejam J uma estrutura de interpretação e F uma fórmula fechada. Então $[\![F]\!]_{J,\rho} = [\![F]\!]_{J,\sigma}$ para quaisquer atribuições ρ e σ.

Prova. Basta observar que quaisquer duas atribuições coincidem no conjunto vazio. □

O corolário anterior justifica que se escreva simplesmente $[\![F]\!]_J$ quando F for uma fórmula fechada.

Definição 1.3.8 Sejam J uma estrutura de interpretação e F uma fórmula sobre uma assinatura de primeira ordem. Diz-se que a fórmula F é:

- *possível* em J se $[\![\exists F]\!]_J = 1$;
- *contraditória* em J se $[\![\exists F]\!]_J = 0$;
- *válida* em J se $[\![\forall F]\!]_J = 1$;
- *falsificável* em J se $[\![\forall F]\!]_J = 0$.

Quando F é válida em J diz-se ainda que J é *modelo* de F.

Definição 1.3.9 Uma estrutura de interpretação J diz-se um *modelo* de um conjunto de fórmulas S se J é modelo de cada uma das fórmulas de S.

Definição 1.3.10 Seja S um conjunto de fórmulas sobre uma assinatura de primeira ordem. Diz-se que o conjunto S é:

- *possível* se alguma estrutura de interpretação para essa assinatura for modelo de S;
- *contraditório* se nenhuma estrutura de interpretação para essa assinatura for modelo de S;
- *válido* se toda a estrutura de interpretação para essa assinatura for modelo de S;
- *falsificável* se alguma estrutura de interpretação para essa assinatura não for modelo de S.

Por abuso de linguagem, diz-se que uma fórmula F é possível, contraditória, válida ou falsificável consoante o conjunto $\{F\}$ o seja.

É consequência da definição que F é possível (em J) sse $(\neg F)$ é falsificável (em J) e que F é válida (em J) sse $(\neg F)$ é contraditória (em J); daí que se fale frequentemente apenas em *possibilidade* e *validade*. Observe-se ainda que para fórmulas fechadas estas duas noções coincidem.

O conceito fundamental desta secção é o seguinte.

Definição 1.3.11 Sejam S um conjunto de fórmulas e F uma fórmula sobre uma dada assinatura de primeira ordem. Diz-se que F é *consequência semântica* de S, denotado por $S \models F$, se todo o modelo de S for um modelo de F.

Proposição 1.3.12 Sejam $F, F_1, \ldots, F_n$ fórmulas fechadas sobre uma assinatura de primeira ordem. Então $\{F_1, \ldots, F_n\} \models F$ sse $F_1 \wedge \ldots \wedge F_n \Rightarrow F$ for uma fórmula válida.

Prova.

($\longrightarrow$) Suponha-se que $\{F_1, \ldots, F_n\} \models F$ e seja J uma estrutura de interpretação arbitrária. Há dois casos a considerar.

(i) Se J é modelo de $\{F_1, \ldots, F_n\}$, então por hipótese J é modelo de F, ou seja, $[\![F]\!]_J = 1$. Então $[\![F_1 \wedge \ldots \wedge F_n \Rightarrow F]\!]_J = 1$.

(ii) Se J não é modelo de $\{F_1, \ldots, F_n\}$, então existe i tal que $[\![F_i]\!]_J = 0$, donde $[\![F_1 \wedge \ldots \wedge F_n]\!]_J = 0$ e por conseguinte $[\![F_1 \wedge \ldots \wedge F_n \Rightarrow F]\!]_J = 1$.

($\longleftarrow$) Suponha-se que $F_1 \wedge \ldots \wedge F_n \Rightarrow F$ é uma fórmula válida e seja J uma estrutura de interpretação arbitrária tal que J é modelo de $\{F_1, \ldots, F_n\}$. Então $[\![F_i]\!]_J = 1$ para qualquer i, pelo que $[\![F_1 \wedge \ldots \wedge F_n]\!]_J = 1$. Por outro lado, uma vez que $[\![F_1 \wedge \ldots \wedge F_n \Rightarrow F]\!]_J = 1$ (pois esta fórmula é válida) conclui-se que $[\![F]\!]_J = 1$. Como J é arbitrária, tem-se que $\{F_1, \ldots, F_n\} \models F$.

□

Importa salientar que este resultado não é válido para fórmulas que não sejam fechadas. É um bom exercício perceber que passo do raciocínio anterior é que não está correcto se as fórmulas $F, F_1, \ldots, F_n$ não forem todas fechadas.

O resultado seguinte fornece uma caracterização muito útil da consequência semântica.

Proposição 1.3.13 Sejam S um conjunto de fórmulas e F uma fórmula fechada. Então $S \models F$ sse $S \cup \{(\neg F)\}$ for contraditório.

Prova.

($\longrightarrow$) Suponha-se que $S \models F$ e seja J uma estrutura de interpretação arbitrária. Se J não for modelo de S, então também não é modelo de $S \cup \{(\neg F)\}$, portanto suponha-se que J é modelo de S. Então J é modelo de F, donde se conclui (como F é fechada) que $[\![F]\!]_J = 1$; então $[\![(\neg F)]\!]_J = 0$, donde J também não é modelo de $S \cup \{(\neg F)\}$. Logo $S \cup \{(\neg F)\}$ não tem modelos e portanto é contraditório.

($\longleftarrow$) Suponha-se que $S \not\models F$. Então existe uma estrutura de interpretação J tal que J é modelo de S mas J não é modelo de F. Como F é fechada, pode-se concluir como atrás que $[\![F]\!]_J = 0$, donde $[\![(\neg F)]\!]_J = 1$. Então J é modelo de $S \cup \{(\neg F)\}$, donde este conjunto não é contraditório.

□

Mais uma vez, se F não for fechada apenas uma destas implicações é válida.

1.4 Semântica da Programação em Lógica

O problema fundamental da Programação em Lógica é determinar, dados um programa determinado P e um objectivo determinado G, se $P \models G$; pelo último resultado da secção anterior, isto equivale a determinar se $P \cup \{\neg G\}$ tem ou não um modelo. Nesta secção ver-se-á que para responder a esta questão neste caso particular não é necessário olhar para todos os modelos mas apenas para uma classe restrita de estruturas de interpretação. Nos capítulos seguintes apresentar-se-ão diferentes métodos de encontrar essas estruturas de interpretação.

Definição 1.4.1 Um termo ou átomo diz-se *fechado* se não contém variáveis.

Em particular, um átomo é fechado sse for uma fórmula atómica fechada.

Definição 1.4.2 Seja Σ uma assinatura de primeira ordem com pelo menos um símbolo de constante.

- O *universo de Herbrand* para Σ, denotado por $\mathcal{U}_\Sigma$, é o conjunto de todos os termos fechados sobre a assinatura Σ.

- A *base de Herbrand* para Σ, denotada por $\mathcal{B}_\Sigma$, é o conjunto de todos os átomos fechados sobre a assinatura Σ.
- Uma *estrutura de Herbrand* para Σ é uma estrutura de interpretação H tal que:
  (i) o domínio de H é $\mathcal{U}_\Sigma$;
  (ii) para cada constante c, tem-se que $c_H = c$;
  (iii) para cada símbolo de função f tem-se que $f_H(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$.
- Um *modelo de Herbrand* para um conjunto de fórmulas S sobre Σ é uma estrutura de Herbrand para Σ que é modelo de S.

Proposição 1.4.3 Seja Σ uma assinatura de primeira ordem. Existe uma bijecção entre as estruturas de Herbrand sobre Σ e os subconjuntos de $\mathcal{B}_\Sigma$.

Prova. Repare-se que a definição de estrutura de Herbrand fixa a definição da interpretação de todos os termos fechados. Assim, as estruturas de Herbrand distinguem-se apenas pelos valores que dão à interpretação dos predicados. A bijecção pretendida é então definida do seguinte modo:

- dada uma estrutura de Herbrand H, define-se $\beta(H) = \{p(t_1, \ldots, t_n) \mid [\![p(t_1, \ldots, t_n)]\!]_H = 1\}$;
- dado um conjunto $Y \subseteq \mathcal{B}_\Sigma$, define-se $\beta^{-1}(Y) = H$ tal que $p_H(t_1, \ldots, t_n) = 1$ sse $p(t_1, \ldots, t_n) \in Y$.

É fácil verificar que β e β^{-1} estão bem definidas, que $\beta \circ \beta^{-1} = \mathrm{id}$ e que $\beta^{-1} \circ \beta = \mathrm{id}$. □

Notação 1.4.4 Uma vez que só raramente se menciona explicitamente a assinatura subjacente a um conjunto de fórmulas, é usual convencionar que esta é a assinatura definida pelos símbolos (de constante, predicado e de função) que ocorrem no conjunto. Assim, dado um conjunto de fórmulas S, utilizam-se as notações $\mathcal{U}_S$ e $\mathcal{B}_S$ para denotar o universo de Herbrand e a base de Herbrand (respectivamente) para a assinatura definida implicitamente por S.

No caso particular de um programa determinado P denotam-se analogamente o seu universo de Herbrand por $\mathcal{U}_P$ e a sua base de Herbrand por $\mathcal{B}_P$. É importante salientar que estas notações só fazem sentido quando P tem pelo menos um símbolo de constante.

O interesse de considerar estruturas de Herbrand é expresso pela seguinte proposição.

Proposição 1.4.5 Seja S um conjunto de cláusulas. Se S tem um modelo, então tem um modelo de Herbrand.

Prova. Sejam S um conjunto de cláusulas com modelo e J uma estrutura de interpretação tal que J é modelo de S. Utilizando J, defina-se do seguinte modo uma estrutura de interpretação H.

- Domínio: o universo de Herbrand $\mathcal{U}_S$.
- Interpretação: $c_H = c$, $f_H(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$ e $p_H(t_1, \ldots, t_n) = p_J([\![t_1]\!]_J, \ldots, [\![t_n]\!]_J)$.

Claramente H é uma estrutura de Herbrand; observe-se também que p_H está bem definido, já que os elementos do domínio são termos fechados e portanto a sua interpretação em J é independente da atribuição escolhida (Lema 1.3.5).

Por hipótese J é modelo de S, ou seja, para qualquer cláusula $F \in S$ tem-se que $[\![F]\!]_{J,\sigma} = 1$ para toda a atribuição σ. Para mostrar que H também é modelo de S, comece-se por se fixar uma

atribuição ρ de valores em H e defina-se uma atribuição $\hat{\rho}$ de valores em J por $\hat{\rho}(x) = [\![\rho(x)]\!]_J$. Mais uma vez, esta definição faz sentido já que $\rho(x)$ é um termo fechado interpretável em J.

Lema. Para todo o $t \in \mathcal{U}_S$, tem-se que $\left[\!\left[[\![t]\!]_{H,\rho} \right]\!\right]_J = [\![t]\!]_{J,\hat{\rho}}$.

Prova. Por indução na estrutura de t.

- Se t é um símbolo de constante c, então $\left[\!\left[[\![c]\!]_{H,\rho} \right]\!\right]_J = [\![c]\!]_J = [\![c]\!]_{J,\hat{\rho}}$ por definição de interpretação.
- Se t é uma variável x, então $\left[\!\left[[\![x]\!]_{H,\rho} \right]\!\right]_J = [\![\rho(x)]\!]_J = \hat{\rho}(x) = [\![x]\!]_{J,\hat{\rho}}$ por definição de interpretação e escolha de $\hat{\rho}$.
- Se t é $f(t_1, \ldots, t_n)$, então tem-se:

$$
\begin{array}{lll}
\left[\!\left[[\![f(t_1, \ldots, t_n)]\!]_{H,\rho} \right]\!\right]_J = & & \\
= & \left[\!\left[f\left([\![t_1]\!]_{H,\rho}, \ldots, [\![t_n]\!]_{H,\rho} \right) \right]\!\right]_J & \text{por definição de interpretação (em } H) \\
= & f_J\left(\left[\!\left[[\![t_1]\!]_{H,\rho} \right]\!\right]_J, \ldots, \left[\!\left[[\![t_n]\!]_{H,\rho} \right]\!\right]_J \right) & \text{por definição de interpretação (em } J) \\
= & f_J([\![t_1]\!]_{J,\hat{\rho}}, \ldots, [\![t_n]\!]_{J,\hat{\rho}}) & \text{por hipótese de indução} \\
= & [\![f(t_1, \ldots, t_n)]\!]_{J,\hat{\rho}} & \text{por definição de interpretação (em } J)
\end{array}
$$

■

Seja agora $A_1, \ldots, A_p \longleftarrow B_1, \ldots, B_q$ uma cláusula de S. Como J é modelo de S, tem-se em particular que $[\![A_1, \ldots, A_p \longleftarrow B_1, \ldots, B_q]\!]_J = 1$. Atendendo à definição de cláusula, isto implica que $[\![A_1 \vee \ldots \vee A_p \vee \neg B_1 \vee \ldots \vee \neg B_q]\!]_{J,\hat{\rho}} = 1$; ou seja, existe algum i tal que $[\![A_i]\!]_{J,\hat{\rho}} = 1$ ou existe algum j tal que $[\![B_j]\!]_{J,\hat{\rho}} = 0$. Mas quer A_i quer B_j são átomos; tem-se:

$$
\begin{array}{lll}
[\![p(t_1, \ldots, t_n)]\!]_{J,\hat{\rho}} = & & \\
= & p_J([\![t_1]\!]_{J,\hat{\rho}}, \ldots, [\![t_n]\!]_{J,\hat{\rho}}) & \text{por definição de interpretação (em } J) \\
= & p_J\left(\left[\!\left[[\![t_1]\!]_{H,\rho} \right]\!\right]_J, \ldots, \left[\!\left[[\![t_n]\!]_{H,\rho} \right]\!\right]_J \right) & \text{pelo lema acima provado} \\
= & p_H([\![t_1]\!]_{H,\rho}, \ldots, [\![t_1]\!]_{H,\rho}) & \text{por definição de } H \\
= & [\![p(t_1, \ldots, t_n)]\!]_{H,\rho} & \text{por definição de interpretação (em } H)
\end{array}
$$

Logo, conclui-se respectivamente que $[\![A_i]\!]_{H,\rho} = 1$ ou $[\![B_j]\!]_{H,\rho} = 0$; em qualquer dos casos segue imediatamente que $[\![A_1, \ldots, A_p \longleftarrow B_1, \ldots, B_q]\!]_{H,\rho} = 1$. Uma vez que esta cláusula é arbitrária, H é modelo de S. □

Corolário 1.4.6 Seja S um conjunto de fórmulas. Então S é contraditório sse S não tem nenhum modelo de Herbrand.

Prova.

($\longrightarrow$) Suponha-se que S é contradiório. Então S não tem nenhum modelo, logo em particular não tem modelos de Herbrand.

($\longleftarrow$) Suponha-se que S não é contraditório. Então S tem um modelo; pela Proposição 1.4.5, S tem um modelo de Herbrand.

□

O corolário anterior é de interesse fundamental: consegue-se reduzir o problema de determinar se $P \models G$ ao problema da existência de um modelo de Herbrand para $P \cup \{\neg G\}$ no caso de P ser um programa determinado e de G ser um objectivo determinado. Nos capítulos seguintes ver-se-á que as propriedades do fragmento clausal da Lógica de Primeira Ordem permitem responder a esta questão com alguma facilidade.

Observe-se no entanto que a Proposição 1.4.5 não é válida para conjuntos arbitrários de fórmulas, como o seguinte exemplo mostra.

Exemplo 1.4.7 O conjunto $S = \{p(a), \exists x.\neg p(x)\}$ é possível mas não tem modelos de Herbrand.

De facto, tomando uma estrutura de interpretação J com domínio $\{0, 1\}$ e definindo $a_J = 1$, $p_J(0) = 0$ e $p_J(1) = 1$ é fácil de verificar que J é modelo de S. No entanto, as estruturas de Herbrand para S têm domínio $\mathcal{U}_S = \{a\}$, pelo que apenas existem duas estruturas de Herbrand para S (Proposição 1.4.3): H_1 tal que $p_{H_1}(a) = 0$ e H_2 tal que $p_{H_2}(a) = 1$. É fácil verificar que H_1 satisfaz $\exists x.\neg p(x)$ mas não $p(a)$, enquanto H_2 satisfaz $p(a)$ mas não $\exists x.\neg p(x)$.

Capítulo 2

Substituições e Unificação

No capítulo anterior definiram-se programas determinados e objectivos determinados e introduziu-se a noção de consequência semântica. Neste capítulo discutem-se o conceito fundamental de substituição e a noção de unificação. Apresenta-se ainda um algoritmo de unificação cuja correcção total se prova.

2.1 Substituições

O conceito de substituição é central à Lógica de Primeira Ordem, já que é o que permite instanciar fórmulas universalmente quantificadas para obter conclusões acerca de objectos concretos. Nesta secção estuda-se mais aprofundadamente o conceito de substituição e apresenta-se um algoritmo para determinar se um conjunto de expressões é ou não unificável.

Definição 2.1.1 Uma *substituição* é um conjunto finito de pares $\{v_1/t_1, \ldots, v_n/t_n\}$ em que:

- cada v_i é uma variável;
- cada t_i é um termo distinto de v_i;
- se $i \neq j$ então v_i é distinta de v_j.

Cada par v_i/t_i diz-se uma *instanciação* de v_i. Se os termos $t_1, \ldots, t_n$ forem todos fechados, diz-se que a substituição é *fechada* ou *terminal*; se os termos $t_1, \ldots, t_n$ forem todos variáveis a substituição diz-se *pura*.

Notação 2.1.2 A substituição vazia denota-se por ε.

Definição 2.1.3 Uma *expressão* é um termo, um literal, uma conjunção de literais ou uma disjunção de literais. Termos e literais dizem-se ainda expressões *simples*.

Definição 2.1.4 Sejam $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ uma substituição e E uma expressão. A *instância* de E por θ, denotada por $E\theta$, é a expressão que se obtém de E substituindo simultaneamente cada ocorrência da variável v_i por t_i.

Uma *instância fechada* de E é uma instância de E (por alguma substituição θ) que não contém variáveis.

O conceito de instância estende-se naturalmente a conjuntos: se S é um conjunto de expressões, então $S\theta = \{E\theta \mid E \in S\}$ diz-se a instância de S por θ. É importante salientar que a cardinalidade de $S\theta$ pode ser menor que a de S (mas não maior).

Definição 2.1.5 Sejam E uma expressão e V o conjunto das variáveis que ocorrem em E. Uma *mudança de variáveis* para E é uma substituição pura $\{x_1/y_1, \ldots, x_n/y_n\}$ tal que:

- $\{x_1, \ldots, x_n\} \subseteq V$ (*domínio*);
- se $i \neq j$ então $y_i \neq y_j$ (*injectividade*);
- $(V \setminus \{x_1, \ldots, x_n\}) \cap \{y_1, \ldots, y_n\} = \emptyset$ (*compatibilidade*).

No seguimento será útil ter uma noção de composição de substituições que satisfaça a igualdade $E(\theta\sigma) = (E\theta)\sigma$. A definição seguinte tem essa propriedade, como adiante se verá.

Definição 2.1.6 Sejam $\theta = \{u_1/s_1, \ldots, u_p/s_p\}$ e $\sigma = \{v_1/t_1, \ldots, v_q/t_q\}$ substituições. A sua *composição* $\theta\sigma$ é a substituição obtida a partir de $\{u_1/s_1\sigma, \ldots, u_p/s_p\sigma, v_1/t_1, \ldots, v_q/t_q\}$ suprimindo os pares $u_i/s_i\sigma$ tais que $u_i = s_i\sigma$ e os pares v_j/t_j tais que $v_j \in \{u_1, \ldots, u_p\}$.

É fácil verificar que a composição de substituições ainda é uma substituição.

Exemplo 2.1.7

1. Sejam $\theta = \{x/f(y, w, x), y/z\}$ e $\sigma = \{x/a, y/b, z/y\}$. Então $\theta\sigma = \{x/f(b, w, a), z/y\}$.
2. Sejam $\theta = \{x/f(y), y/z\}$ e $\sigma = \{x/a, z/b\}$. Então $\theta\sigma = \{x/f(y), y/b, z/b\}$.

Proposição 2.1.8 Sejam θ e σ substituições e E uma expressão. Então $E(\theta\sigma) = (E\theta)\sigma$.

Prova. Por indução na estrutura de E. Sejam $\theta = \{u_1/s_1, \ldots, u_p/s_p\}$ e $\sigma = \{v_1/t_1, \ldots, v_q/t_q\}$.

- Se E é uma constante c, então $c(\theta\sigma) = c = c\sigma = (c\theta)\sigma$.
- Se E é uma variável há vários casos a considerar.

  Se E é u_i e $u_i \neq s_i\sigma$, então $u_i(\theta\sigma) = s_i\sigma = (u_i\theta)\sigma$.

  Se E é u_i e $u_i = s_i\sigma$, então $u_i(\theta\sigma) = u_i = s_i\sigma = (u_i\theta)\sigma$.

  Se E é v_j e $v_j = u_i$, então aplica-se um dos casos anteriores.

  Se E é v_j e $v_j \neq u_i$ para todo o i, então $E(\theta\sigma) = t_j = v_j\sigma = (v_j\theta)\sigma$, onde na última igualdade se utilizou o facto de $v_j = v_j\theta$, que é consequência de v_j não ocorrer entre $u_1, \ldots, u_p$.
- Se E é um símbolo de função ou de predicado τ aplicado a termos, $\tau(t_1, \ldots, t_n)$, então tem-se, utilizando a hipótese de indução, que $(\tau(t_1, \ldots, t_n))(\theta\sigma) = \tau(t_1(\theta\sigma), \ldots, t_n(\theta\sigma)) = \tau((t_1\theta)\sigma, \ldots, (t_n\theta)\sigma) = ((\tau(t_1, \ldots, t_n))\theta)\sigma$.
- Se E é a negação dum átomo $(\neg A)$, então $(\neg A)(\theta\sigma) = \neg(A(\theta\sigma)) = \neg((A\theta)\sigma) = ((\neg A)\theta)\sigma$, onde a segunda igualdade é consequência da hipótese de indução.
- Se E é uma conjunção ou disjunção de literais, $L_1 * \ldots * L_n$, tem-se analogamente $(L_1 * \ldots * L_n)(\theta\sigma) = L_1(\theta\sigma) * \ldots * L_n(\theta\sigma) = (L_1\theta)\sigma * \ldots * (L_n\theta)\sigma = ((L_1 * \ldots * L_n)\theta)\sigma)$.

□

Atendendo a este resultado, utilizar-se-á a notação $E\theta\sigma$ quando não for relevante distinguir entre as expressões $E(\theta\sigma)$ e $(E\theta)\sigma$.

Proposição 2.1.9 As substituições com a operação de composição constituem um monóide com elemento neutro ε.

Prova. É necessário mostrar que a composição de substituições é uma operação associativa e que $\theta\varepsilon = \varepsilon\theta = \theta$ para qualquer substituição θ.

Para a associatividade, convém observar em primeiro lugar que $\sigma = \theta$ desde que $x\sigma = x\theta$ para qualquer variável x. Usando a Proposição 2.1.8 verifica-se facilmente que, sendo θ, σ e δ substituições quaisquer, se tem $x((\theta\sigma)\delta) = (x(\theta\sigma))\delta = ((x\theta)\sigma)\delta = (x\theta)(\sigma\delta) = x(\theta(\sigma\delta))$, donde se conclui que $(\theta\sigma)\delta = \theta(\sigma\delta)$.

Quanto à existência de elemento neutro, sai da definição de composição que $\theta\varepsilon = \{x/t\varepsilon \mid x/t \in \theta\} = \theta$ e que $\varepsilon\theta = \{x/s\theta \mid x/s \in \varepsilon\} \cup \{y/t \mid y/t \in \theta \text{ e } y/t' \notin \varepsilon\} = \emptyset \cup \theta = \theta$. □

Definição 2.1.10 Duas expressões E e F dizem-se *variantes* se E for uma instância de F e F for uma instância de E.

Equivalentemente, E e F são variantes se existirem substituições θ e σ tais que $E = F\theta$ e $F = E\sigma$. O resultado seguinte mostra que se pode tornar esta definição mais restritiva.

Proposição 2.1.11 Duas expressões E e F são variantes sse existirem mudanças de variáveis θ (para F) e σ (para E) tais que $E = F\theta$ e $F = E\sigma$.

Prova.

($\longrightarrow$) Suponha-se que E e F são expressões variantes. Então existem substituições $\hat{\sigma}$ e $\hat{\theta}$ tais que $E = F\hat{\theta}$ e $F = E\hat{\sigma}$. Tome-se $\sigma = \{x/t \mid x/t \in \hat{\sigma} \text{ e } x \text{ ocorre em } E\}$; então $F = E\sigma$ e que σ é uma mudança de variáveis para E.

- Prova-se que $F = E\sigma$ por indução na estrutura de E. Se E for uma constante o resultado é trivial; se E for uma variável então $E\sigma = E\hat{\sigma} = F$ atendendo à construção de σ. Os restantes casos são consequência imediata da hipótese de indução.
- Para provar que σ é uma mudança de variáveis para E, note-se que $E = F\hat{\theta} = (E\sigma)\hat{\theta} = E(\sigma\hat{\theta})$. Por conseguinte, se x for uma variável que ocorra em E não pode haver nenhum par x/t em $\sigma\hat{\theta}$ (já que o único par possível seria x/x, que não é permitido pela definição de substituição). Então:
  - se $x/t \in \sigma$, então x ocorre em E por definição de σ, donde, atendendo à observação acima, se tem $t\hat{\theta} = x$, o que só é possível se t for uma variável;
  - por construção de σ, se $x/t \in \sigma$ então x ocorre em E;
  - se $x\sigma = y\sigma$ então $x = x\sigma\hat{\theta} = y\sigma\hat{\theta} = y$; em particular, σ é injectiva (tome-se x e y tais que $\{x/z, y/z\} \subseteq \sigma$) e compatível (tome-se $x/z \in \sigma$ e y uma variável de E não afectada por σ).

  Logo σ é uma mudança de variáveis para E.

Analogamente se definiria uma mudança de variáveis θ com $E = F\theta$.

($\longleftarrow$) Trivial, atendendo a que mudanças de variável são substituições.

□

No contexto da Programação em Lógica, determinadas substituições têm um interesse especial.

Definição 2.1.12 Sejam P um programa determinado e $G \equiv \longleftarrow B_1, \ldots, B_q$ um objectivo determinado. Uma *resposta* de P a G é uma substituição para variáveis de G.

Uma resposta σ de P a G diz-se *correcta* se $P \models \forall((B_1 \wedge \ldots \wedge B_q)\sigma)$.

Proposição 2.1.13 Sejam P um programa determinado e $G \equiv \longleftarrow B_1, \ldots, B_q$ um objectivo determinado. Uma substituição σ é uma resposta correcta de P a G sse $P \cup \{\neg\forall((B_1 \wedge \ldots \wedge B_q)\sigma)\}$ for contraditório.

Prova. Consequência imediata da Proposição 1.3.13. □

Definição 2.1.14 Sejam P um programa determinado e G um objectivo determinado. A *resposta booleana correcta* de P a G é `não` se $P \cup \{G\}$ não for contraditório e `sim` se $P \cup \{G\}$ for contraditório e G não tiver variáveis.

Observe-se que `sim` é uma resposta booleana correcta de P a G sse ε for uma resposta correcta de P a G e G não tiver variáveis, enquanto que `não` é uma resposta booleana correcta de P a G sse não houver respostas correctas de P a G.

O objectivo principal da Programação em Lógica é obter respostas correctas de programas determinados a objectivos determinados. Os próximos capítulos discutem diferentes maneiras de se atingir este objectivo.

2.2 Unificação

Outro caso de particular interesse é o de substituições que reduzem um conjunto de expressões a uma única expressão.

Definição 2.2.1 Seja S um conjunto finito e não vazio de expressões simples. Uma substituição θ diz-se uma *unificadora* de S se $S\theta$ for um conjunto singular. Quando existe uma substituição nestas condições, diz-se que S é *unificável*.

Uma unificadora θ de S diz-se uma *unificadora mais geral (UMG)* se para toda a unificadora σ de S existir uma substituição γ tal que $\sigma = \theta\gamma$.

Exemplo 2.2.2

1. Seja $S = \{p(f(x), z), p(y, a)\}$. Então S é unificável: $\theta_1 = \{x/z, y/f(z), z/a\}$, $\theta_2 = \{x/f(a), y/f(f(a)), z/a\}$ e $\theta_3 = \{y/f(x), z/a\}$ são unificadoras de S. Destas, θ_1 e θ_3 são UMGs de S (exercício).
2. Seja $S = \{p(f(x), a), p(y, f(w))\}$. Então S não é unificável: não há nenhuma substituição que possa unificar a e $f(w)$.

Proposição 2.2.3 Sejam θ e σ UMGs para um conjunto $S = \{E_1, \ldots, E_n\}$ de expressões simples.

(i) Existem substituições puras γ_1 e γ_2 tais que $\sigma = \theta\gamma_1$ e $\theta = \sigma\gamma_2$.

(ii) Para cada $i = 1, \ldots, n$, as expressões $E_i\sigma$ e $E_i\theta$ são variantes.

Prova.

(i) Suponha-se que σ e θ são ambas UMGs de S; então por definição de UMG existem substituições $\hat{\gamma_1}$ e $\hat{\gamma_2}$ tais que $\sigma = \theta\hat{\gamma_1}$ e $\theta = \sigma\hat{\gamma_2}$.

Tome-se $\gamma_i = \{x/y \mid x/y \in \hat{\gamma_i}\}$, ou seja, γ_i é a parte pura de $\hat{\gamma_i}$. Por construção cada γ_i é uma substituição pura, restando apenas mostrar que $\sigma = \theta\gamma_1$ e $\theta = \sigma\gamma_2$.

Por hipótese, $\sigma = \theta\hat{\gamma_1}$ e $\theta = \sigma\hat{\gamma_2}$, donde se conclui que $x\theta = x\theta\hat{\gamma_1}\hat{\gamma_2}$. Seja y uma variável que ocorre em $x\theta$; atendendo à definição de instanciação, necessariamente se tem $y = y\hat{\gamma_1}\hat{\gamma_2}$. Forçosamente, $y\hat{\gamma_1}$ é uma variável, donde $y/y\hat{\gamma_1} \in \gamma_1$.

Conclui-se então que $x\theta\hat{\gamma_1} = x\theta\gamma_1$ para qualquer variável x, donde $\sigma = \theta\hat{\gamma_1} = \theta\gamma_1$. A prova de que $\theta = \sigma\gamma_2$ é análoga.

(ii) Uma vez que σ e θ são UMGs de S, existem substituições γ_1 e γ_2 tais que $\sigma = \theta\gamma_1$ e $\theta = \sigma\gamma_2$; então $E_i\sigma = E_i\theta\gamma_1$ e $E_i\theta = E_i\sigma\gamma_2$, donde $E_i\sigma$ e $E_i\theta$ são expressões variantes.

□

Definição 2.2.4 Seja $S = \{\varphi_1(E_{1,1}, \ldots, E_{1,n_1}), \ldots, \varphi_k(E_{k,1}, \ldots, E_{k,n_k})\}$ um conjunto finito não vazio de expressões simples. O *conjunto de conflitos* de S, denotado por $\delta(S)$, define-se indutivamente como se segue.

- $\delta(S) = S$ se houver índices i e j tais que $\varphi_i \neq \varphi_j$.
- $\delta(S) = \emptyset$ se S for um conjunto singular.
- $\delta(S) = \delta(\{E_{1,i}, \ldots, E_{k,i}\})$ com i mínimo tal que $\delta(\{E_{1,i}, \ldots, E_{k,i}\}) \neq \emptyset$ caso nenhuma das condições anteriores se verifique.

É importante observar que a terceira condição só é aplicada quando os φ_is são todos idênticos, caso em que $n_1 = \ldots = n_k$.

Exemplo 2.2.5

1. $\delta(\{q(f(x), y), q(z, g(w))\}) = \delta(\{f(x), z\}) = \{f(x), z\}$;
2. $\delta(\{p(f(a), g(x)), p(f(a), f(a))\}) = \delta(\{g(x), f(a)\}) = \{g(x), f(a)\}$;
3. $\delta(\{p(a, h(y), h(g(a))), p(a, h(y), h(y))\}) = \delta(\{h(g(a)), h(y)\}) = \delta(\{g(a), y\}) = \{g(a), y\}$.

Algoritmo 2.2.6 O *algoritmo de unificação* recebe como dados um conjunto S de expressões simples e devolve `true` se S for unificável e `false` caso contrário; no primeiro caso devolve ainda uma UMG de S.

```
begin
   σ := ε
   r := true
   while #(Sσ) > 1 and r do
      if v, t ∈ δ(Sσ) e v não ocorre em t
      then σ := σ{v/t}
      else r := false
   end do
end
```

Na sequência mostrar-se-á que este algoritmo termina para todos os valores do argumento e devolve o resultado pretendido. Antes, porém, apresentam-se alguns exemplos de execução deste algoritmo.

Exemplo 2.2.7

$S = \{p(f(a), g(x)), p(y, y)\}$	
0.	$\sigma = \varepsilon$ $r = \texttt{true}$
1.	$S\sigma = S$ $\#(S\sigma) = 2$ $\delta(S\sigma) = \{f(a), y\}$ $\sigma = \{y/f(a)\}$
2.	$S\sigma = \{p(f(a), g(x)), p(f(a), f(a))\}$ $\#(S\sigma) = 2$ $\delta(S\sigma) = \{g(x), f(a)\}$ $r = \texttt{false}$

O algoritmo termina com resultado `false`.

Exemplo 2.2.8

	$S = \{p(a, x, h(g(z))), p(z, h(y), h(y))\}$
0.	$\sigma = \varepsilon$ $r = \texttt{true}$
1.	$S\sigma = S$ $\#(S\sigma) = 2$ $\delta(S\sigma) = \{a, z\}$ $\sigma = \{z/a\}$
2.	$S\sigma = \{p(a, x, h(g(a))), p(a, h(y), h(y))\}$ $\#(S\sigma) = 2$ $\delta(S\sigma) = \{x, h(y)\}$ $\sigma = \{z/a\}\{x/h(y)\}$ $= \{z/a, x/h(y)\}$
3.	$S\sigma = \{p(a, h(y), h(g(a))), p(a, h(y), h(y))\}$ $\#(S\sigma) = 2$ $\delta(S\sigma) = \{g(a), y\}$ $\sigma = \{z/a, x/h(y)\}\{y/g(a)\}$ $= \{z/a, x/h(g(a)), y/g(a)\}$
4.	$S\sigma = \{p(a, h(g(a)), h(g(a)))\}$ $\#(S\sigma) = 1$

O algoritmo termina com resultado `true` e substituição $\{z/a, x/h(g(a)), y/g(a)\}$.

É instrutivo verificar que a substituição encontrada não é apenas uma unificadora de S, mas trata-se de facto de uma UMG para este conjunto.

O último exemplo apresenta uma situação em que o conjunto dado não é unificável e a terminação do algoritmo é devida à verificação de ocorrência na guarda do `if`. É importante notar que sem esta verificação de ocorrência o programa não terminaria, obtendo-se portanto um procedimento apenas parcialmente correcto.

Exemplo 2.2.9

$S = \{p(x,x), p(y,f(y))\}$	
0.	$\sigma = \varepsilon$ $r = \texttt{true}$
1.	$S\sigma = S$ $\#(S\sigma) = 2$ $\delta(S\sigma) = \{x, y\}$ $\sigma = \{x/y\}$
2.	$S\sigma = \{p(y,y), p(y,f(y))\}$ $\#(S\sigma) = 2$ $\delta(S\sigma) = \{y, f(y)\}$ $r = \texttt{false}$

O algoritmo termina com resultado `false`.

Teorema 2.2.10 *(Teorema da Unificação.)* Seja S um conjunto finito de expressões simples. Se S é unificável, então o algoritmo de unificação termina e devolve resultado `true` conjuntamente com uma UMG de S. Se S não é unificável, então o algoritmo de unificação termina e devolve resultado `false`.

Prova. Começa por se provar a correcção parcial do algoritmo, mostrando-se posteriormente a sua terminação para todos os valores do argumento.

Correcção parcial. Se o algoritmo termina, as condições do teorema verificam-se.

Suponha-se que o algoritmo termina. Então a guarda do ciclo `while` é falsa, logo ou $\#(S\sigma) \leq 1$ ou $r = \texttt{false}$.

- Se S não é unificável, então $\#(S\sigma) > 1$ (caso contrário σ seria uma unificadora de S). Logo $r = \texttt{false}$ e o programa devolve `false` como o teorema afirma.
- Suponha-se agora que S é unificável. Então a condição

$$I \equiv (r = \texttt{true}) \wedge [\forall\theta.(\#(S\theta) = 1) \Rightarrow (\exists\gamma.\theta = \sigma\gamma)]$$

é uma condição invariante do ciclo.

  - A inicialização estabelece I, pois a r é atribuído o valor `true` e $\sigma = \varepsilon$ satisfaz trivialmente a segunda parte da conjunção.
  - A execução do corpo do ciclo quando a guarda é verdadeira preserva a condição I. Por hipótese S é unificável, logo existe θ tal que $S\theta$ é um conjunto singular. Por I existe uma substituição γ tal que $\theta = \sigma\gamma$. Então $S\theta = S\sigma\gamma = (S\sigma)\gamma$, logo $S\sigma$ é unificável. Como $S\sigma$ não é singular, o conjunto $\delta(S\sigma)$ também não é singular; mas uma vez que $S\sigma$ é unificável, existe necessariamente uma variável v em $\delta(S\sigma)$. Escolhendo $t \in (\delta(S\sigma) \setminus \{v\})$, de $v\gamma = t\gamma$ conclui-se que v não ocorre em t. Logo a guarda do `if` verifica-se.
    Resta mostrar que $\sigma' = \sigma\{v/t\}$ ainda satisfaz a segunda parte da condição I. Sejam θ uma unificadora de S e γ uma substituição satisfazendo $\theta = \sigma\gamma$. Então $\gamma' = \gamma \setminus \{v/t\}$ satisfaz $\theta = \sigma'\gamma'$:
    * se $\{v/t\} \in \gamma$, então $\{v/t\}\gamma' = \{v/t\gamma'\} \cup \gamma' = \{v/t\gamma\} \cup \gamma' = \{v/v\gamma\} \cup \gamma = \gamma$, onde na segunda igualdade se usou o facto de v não ocorrer em t;

- se $\{v/t\} \notin \gamma$, então $\gamma = \gamma'$; tal como acima, mostra-se que $v\gamma = t\gamma$, donde $v = t\gamma$; então $\{v/t\}\gamma' = \{v/t\}\gamma = \gamma$ por definição de composição de substituições.

Então em ambos os casos tem-se $\theta = \sigma\gamma = \sigma(\{v/t\}\gamma') = (\sigma\{v/t\})\gamma' = \sigma'\gamma'$. Logo I é satisfeita.

Suponha-se então que o programa termina; então a guarda é falsa. Por hipótese I é satisfeita, donde $r =$ `true`; logo $\#(S\sigma) = 1$, donde σ é uma unificadora de S. Da segunda parte da condição I conclui-se que σ é uma UMG de S.

Terminação. A cada passo do ciclo ou r toma o valor `false` e o programa termina na iteração seguinte ou r mantém-se com o valor `true` e a substituição σ é aumentada com um par x/t, em que x é uma variável que ocorre em $S\sigma$. Neste último caso, o número de variáveis em $S\sigma$ diminui (devido à verificação de ocorrência); como S é finito, isto implica que esta situação só pode ocorrer um número finito de vezes.

□

Capítulo 3

Semântica Operacional da Programação em Lógica

Neste capítulo descreve-se a semântica operacional da Programação em Lógica. Como o nome indica, trata-se de dar ao fragmento clausal da Lógica de Primeira Ordem uma interpretação computacional, com a qual é possível calcular. Ver-se-á mais adiante que esta semântica é correcta e completa.

3.1 Resolução-SLD

Nesta secção descreve-se o método de *resolução-SLD*. A resolução-SLD é uma adaptação do mecanismo geral de resolução utilizado em Inteligência Artifical, proposto por Robinson. As iniciais SLD indicam que se trata de uma particularização da resolução Linear com função de Selecção ao universo das cláusulas Determinadas.

Empiricamente, este método consiste em encontrar uma resposta de um programa a um objectivo determinado substituindo sucessivamente cada sub-objectivo pelo corpo duma cláusula cuja cabeça seja unificável com o sub-objectivo escolhido, até que não haja mais sub-objectivos ou que nenhum dos sub-objectivos que restam seja unificável com a cabeça de nenhuma cláusula do programa.

Antes de mais, é necessário determinar qual o sub-objectivo que se escolhe a cada passo. Ver-se-á mais adiante quais as implicações de tal escolha.

Definição 3.1.1 Uma *regra de computação* ou *função de selecção* é uma aplicação $\mathcal{S}$ do conjunto de objectivos determinados para o conjunto de átomos tal que $\mathcal{S}(\longleftarrow A_1, \ldots, A_p) \in \{A_1, \ldots, A_p\}$.

Definição 3.1.2 Sejam $G \equiv \longleftarrow A_1, \ldots, A_p$ um objectivo determinado, $C = A \longleftarrow B_1, \ldots, B_q$ uma cláusula determinada, $\mathcal{S}$ uma função de selecção e θ uma UMG de $\mathcal{S}(G)$ e A. O *objectivo derivado* de G e C via $\mathcal{S}$ usando θ é o objectivo

$$G' \equiv \longleftarrow (A_1, \ldots, A_{n-1}, B_1, \ldots, B_q, A_{n+1}, \ldots, A_p)\theta$$

em que $A_n = \mathcal{S}(G)$. O átomo A_n diz-se o *átomo seleccionado* em G; o objectivo G' diz-se também um *resolvente* de G e C via $\mathcal{S}$.

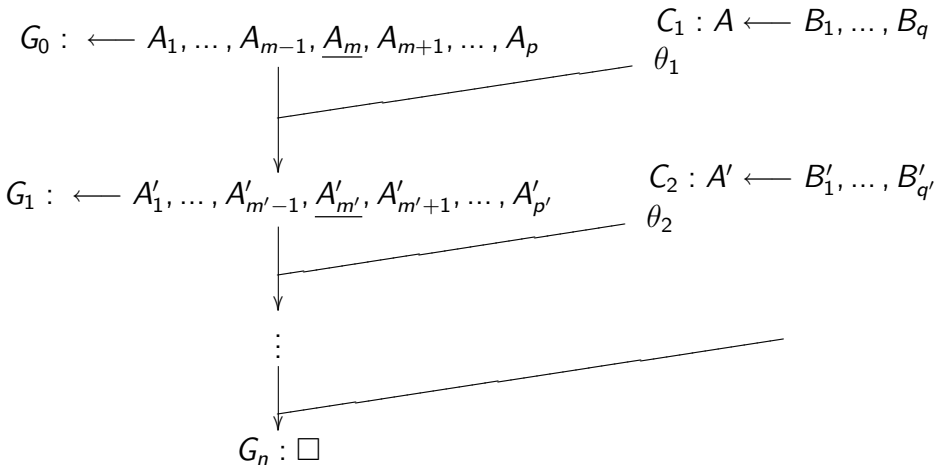
Definição 3.1.3 Sejam P um programa determinado, G_0 um objectivo determinado e $\mathcal{S}$ uma função de selecção. Uma *derivação-SLD* de P e G_0 via $\mathcal{S}$ é um triplo $\langle \mathcal{G}, \mathcal{C}, \Theta \rangle$ em que:

- $\mathcal{G}$ é uma sequência $G_0, G_1, \ldots$ de objectivos determinados;
- $\mathcal{C}$ é uma sequência $C_1, C_2, \ldots$ de variantes de cláusulas de P;
- Θ é uma sequência $\theta_1, \theta_2, \ldots$ de substituições;
- para cada i, G_{i+1} é um resolvente de G_i e C_{i+1} usando θ_{i+1};
- para cada i, C_i é escolhido por forma a não conter nenhuma variável que já tenha ocorrido em $G_0, \ldots, G_i, C_1, \ldots, C_{i-1}$;
- as sequências G, C e Θ ou são infinitas ou terminam em G_n, C_n e θ_n respectivamente, com $G_n = \square$ ou $\mathcal{S}(G_n)$ não unificável com a cabeça de nenhuma cláusula de P.

Uma derivação-SLD finita que termina com a cláusula vazia diz-se uma *refutação-SLD* de P e G; o natural n tal que $G_n = \square$ diz-se o *comprimento* da refutação-SLD. Neste caso, a substituição θ obtida restringindo a composição $\theta_1 \ldots \theta_n$ às variáveis que ocorrem em G_0 diz-se uma *resposta calculada* de P a G_0 via $\mathcal{S}$.

Dito de outra forma, uma derivação-SLD é construída procurando a cada passo uma cláusula C_{i+1} cuja cabeça seja unificável com $\mathcal{S}(G_i)$, calculando uma UMG θ_{i+1} destes dois átomos e o objectivo derivado G_{i+1} de C_{i+1} e G_i usando θ_{i+1}. Este processo continua enquanto for possível. A condição sobre os variantes tem como função garantir que não há unificações "acidentais".

Notação 3.1.4 É usual representar derivações/refutações-SLD graficamente de uma forma que agora se descreve.

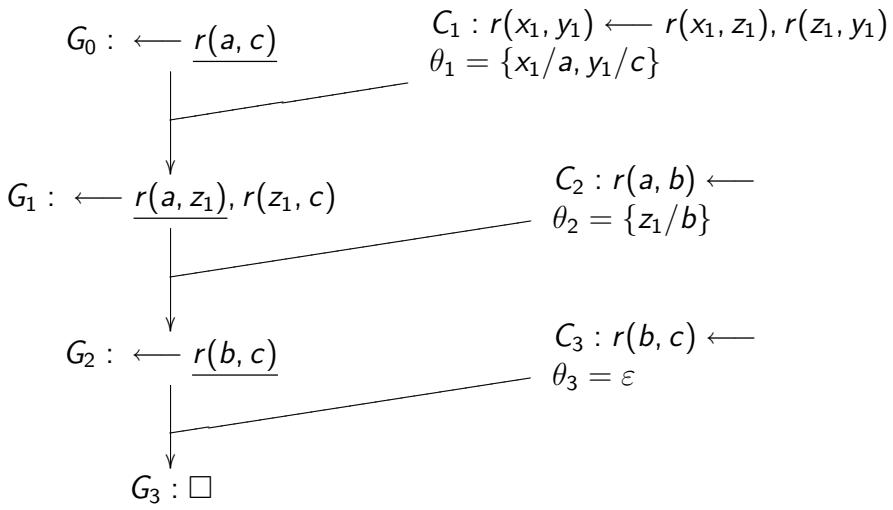


Em cada passo, o átomo seleccionado está sublinhado.

Exemplo 3.1.5 Seja P o programa determinado seguinte.

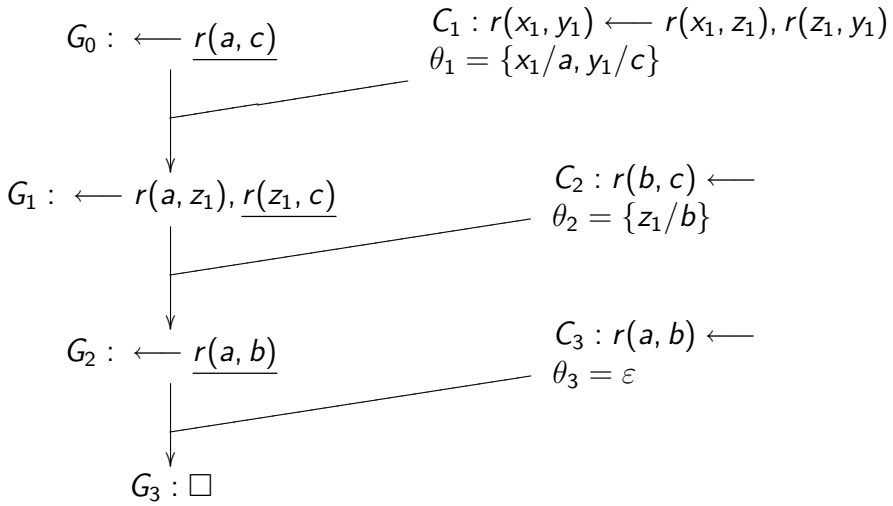
$$\begin{array}{l} r(a,b) \longleftarrow \\ r(b,c) \longleftarrow \\ r(x,y) \longleftarrow r(x,z), r(z,y) \end{array}$$

A seguinte derivação-SLD é uma refutação-SLD de P e $G \equiv \longleftarrow r(a, c)$.

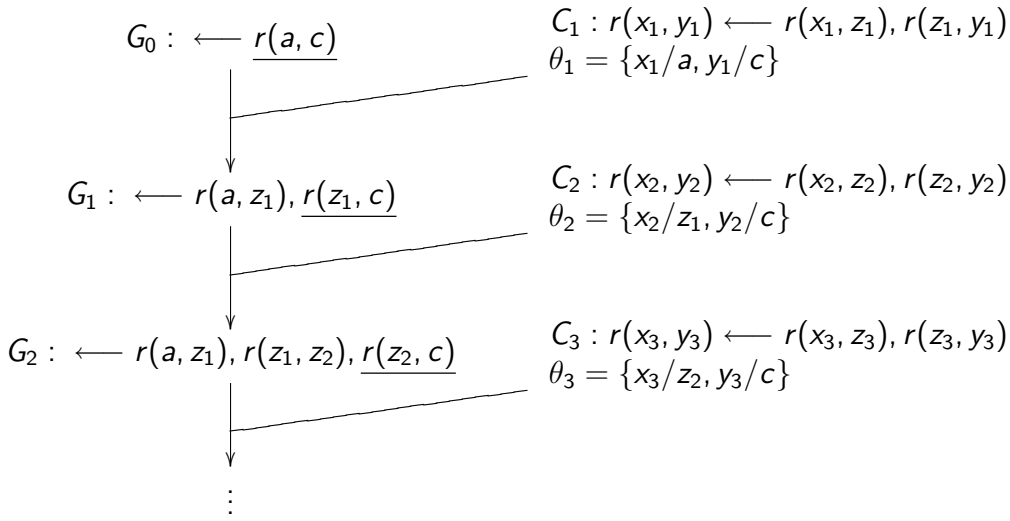


A resposta calculada é $\theta_1\theta_2\theta_3|_\emptyset = \varepsilon$.

Outra refutação-SLD de P e G, também com resposta calculada ε, é a seguinte.



Finalmente, apresenta-se uma derivação-SLD infinita de P e G.



Este último exemplo mostra que para além da função de selecção também é importante a escolha (do variante) da cláusula a utilizar em cada passo. Esta questão será discutida em maior detalhe num capítulo posterior.

Definição 3.1.6 Sejam P um programa determinado e $\mathcal{S}$ uma função de selecção. O *conjunto de sucessos* de P via $\mathcal{S}$ é o conjunto dos átomos $A \in \mathcal{B}_P$ para os quais existe uma refutação-SLD de P e $\longleftarrow A$ via $\mathcal{S}$.

O seguinte teorema é o resultado fundamental de correcção da resolução-SLD.

Teorema 3.1.7 *(Clark.)* Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção. Então toda a resposta calculada de P a G_0 via $\mathcal{S}$ é uma resposta correcta de P a G_0.

Prova. Seja θ uma resposta calculada de P a $G_0 = \longleftarrow A_1, \ldots, A_p$ via $\mathcal{S}$. Então existe uma refutação-SLD de P e G_0 via $\mathcal{S}$ com sequências $G_0, \ldots, G_n$ de objectivos, $C_1, \ldots, C_n$ de variantes de cláusulas de P e $\theta_1, \ldots, \theta_n$ de substituições tal que $\theta = \theta_1 \ldots \theta_n$ restrita às variáveis que ocorrem em G_0.

É conveniente começar por mostrar o seguinte resultado.

Lema. Sejam F uma fórmula e σ uma substituição. Se $P \models \forall F$ então $P \models \forall(F\sigma)$.

Prova. Suponha-se que $P \models \forall F$ e seja J uma estrutura de interpretação que satisfaz P. Então $[\![\forall F]\!]_J = 1$, donde se conclui que $[\![F]\!]_{J,\rho} = 1$ para qualquer atribuição ρ.

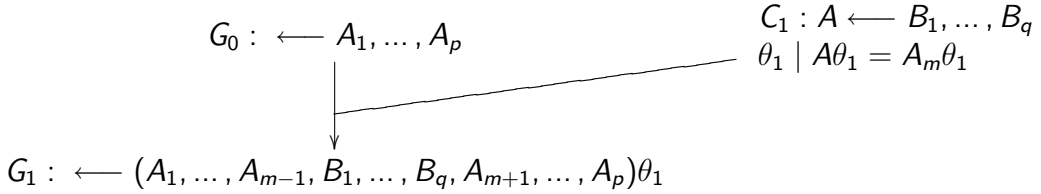
Seja agora ρ' uma atribuição. Mostra-se facilmente por indução estrutural que $[\![F\sigma']\!]_{J,\rho'} = [\![F]\!]_{J,\sigma' * \rho'}$, com $\sigma' * \rho'(x) = \rho'(\sigma'(x))$, de onde segue que $[\![F\sigma']\!]_{J,\rho'} = 1$. Por arbitrariedade de ρ' conclui-se que J é modelo de $F\sigma$ e portanto de $\forall(F\sigma)$. Logo $P \models \forall(F\sigma)$.

■

Observe-se que θ é resposta correcta de P a G sse $P \models \forall((A_1 \wedge \ldots \wedge A_p)\theta_1 \ldots \theta_n)$. Prova-se então por indução no comprimento n da refutação-SLD que isto se passa.

Base. Se $n = 1$, então por definição de derivação-SLD tem-se também $p = 1$ e A_1 é unificável com a cabeça de um facto de P. Seja C_1 o variante escolhido deste facto; então $C_1 = A \longleftarrow$ e $A\theta_1 = A_1\theta_1$. Claramente $P \models \forall A$, donde pelo lema se conclui que $P \models \forall(A\theta_1)$ e portanto $P \models \forall(A_1\theta_1)$.

Passo. Suponha-se que o resultado é válido para derivações de comprimento menor que n. Se $n > 1$, então a refutação-SLD considerada começa da seguinte forma, onde se supõe que $A_m = \mathcal{S}(G_0)$.



Por hipótese de indução, $P \models \forall(((A_1 \wedge \ldots \wedge A_{m-1}, B_1 \wedge \ldots \wedge B_q \wedge A_{m+1} \wedge \ldots \wedge A_p)\theta_1)\theta_2 \ldots \theta_n)$, uma vez que a derivação a partir de G_1 é uma refutação-SLD de P e G_1 com comprimento $n - 1$. Há dois casos a considerar.

- Se $q = 0$, então conclui-se como atrás que $P \models \forall(A_m\theta_1)$, donde se conclui pelo lema que $P \models \forall(A_m\theta_1 \ldots \theta_n)$; conjugando com a hipótese de indução conclui-se que $P \models \forall((A_1 \wedge \ldots \wedge A_p)\theta_1 \ldots \theta_n)$.
- Suponha-se em alternativa que $q > 0$. Como C_1 é um variante duma cláusula de P, tem-se $P \models \forall C_1$, e pelo lema conclui-se que $P \models \forall(C_1\theta_1)$ e que $P \models \forall(C_1\theta_1 \ldots \theta_n)$. Por outro lado, da hipótese de indução sai também que $P \models \forall((B_1 \wedge \ldots \wedge B_q)\theta_1 \ldots \theta_n)$; tendo em conta que C_1 é $A \longleftarrow B_1, \ldots, B_q$, conclui-se que $P \models \forall(A\theta_1 \ldots \theta_n)$, donde $P \models \forall(A_m\theta_1 \ldots \theta_n)$ já que $A\theta_1 = A_m\theta_1$. Mas da hipótese de indução sai ainda que $P \models \forall((A_1 \wedge \ldots A_{m-1} \wedge A_{m+1} \wedge \ldots \wedge A_p)\theta_1 \ldots \theta_n)$. Conjugando estas duas propriedades conclui-se que $P \models \forall((A_1 \wedge \ldots \wedge A_p)\theta_1 \ldots \theta_n)$.

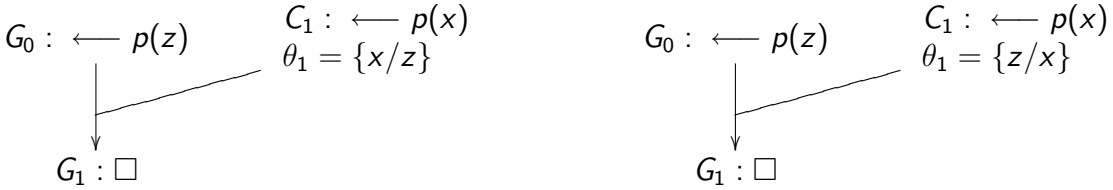
Conclui-se assim a prova de que $P \models \forall((A_1 \wedge \ldots \wedge A_p)\theta_1 \ldots \theta_n)$. Ora $(A_1 \wedge \ldots \wedge A_p)\theta_1 \ldots \theta_n$ e $(A_1 \wedge \ldots \wedge A_p)\theta$ coincidem, uma vez que θ e $\theta_1 \ldots \theta_n$ coincidem em todas as variáveis que ocorrem em $A_1,\ldots,A_p$. Logo θ é uma resposta correcta de P a G_0. □

Observe-se desde já que o recíproco deste resultado não é verdadeiro na sua forma mais geral, como o exemplo seguinte mostra.

Exemplo 3.1.8 Considere-se o programa P seguinte.

$$p(x) \longleftarrow$$
$$q(f(a)) \longleftarrow$$

Há duas respostas calculadas de P ao objectivo $\longleftarrow p(z)$, conforme as refutações-SLD seguintes ilustram.



A primeira refutação-SLD tem resposta calculada ε, enquanto que a segunda tem resposta calculada $\{z/x\}$. Ambas são respostas correctas (como se pode verificar facilmente) e alterando a segunda derivação-SLD obter-se-ia qualquer resposta da forma $\{z/y\}$ com y variável arbitrária. Porém há muitas outras respostas correctas que não são respostas calculadas, como por exemplo $\{z/a\}$ ou $\{z/f(x)\}$. Isto é consequência da exigência de que os θ_is sejam UMGs.

Posteriormente ver-se-á que hipóteses é necessário acrescentar para obter um resultado recíproco ao Teorema de Clark.

Corolário 3.1.9 Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção. Se existe uma refutação-SLD de P e G via $\mathcal{S}$, então o conjunto $P \cup \{G\}$ é contraditório.

Prova. Seja $G \equiv \longleftarrow A_1, \ldots, A_p$. Nas condições da hipótese, se θ for a resposta calculada pela refutação-SLD em questão, o teorema anterior garante que $P \models \forall((A_1 \wedge \ldots \wedge A_p)\theta)$, donde $P \models \exists((A_1 \wedge \ldots \wedge A_p)\theta)$ e por conseguinte $P \models \exists(A_1 \wedge \ldots \wedge A_p)$. Logo todo o modelo de P é modelo de $\exists(A_1 \wedge \ldots \wedge A_p)$, ou seja, de $\neg G$, donde $P \cup \{G\}$ não tem modelos. □

3.2 Robustez da resolução-SLD

Os resultados apresentados na secção anterior dependem fortemente da escolha da função de selecção. Nesta secção mostra-se que, contrariamente ao que talvez tenha vindo a ser sugerido até aqui, esta escolha é de facto bastante secundária e com poucos efeitos nas propriedades *teóricas* da resolução-SLD. O seu impacto na prática será discutido mais adiante.

Notação 3.2.1 Nesta secção, se C for uma cláusula determinada, denotar-se-á por C^+ a sua cabeça e por C^- o seu corpo.

Lema 3.2.2 *(Lema da troca.)* Sejam P um programa determinado e G um objectivo determinado. Para cada refutação-SLD de P e G com sequência de objectivos $G_0, \ldots, G_n$, sequência de variantes de cláusulas $C_1, \ldots, C_n$ e sequência de substituições $\theta_1, \ldots, \theta_n$ em que

- G_{k-1} é $\longleftarrow A_1, \ldots, A_p$
- o átomo seleccionado no passo $k-1$ é A_i
- o átomo seleccionado no passo k é $A_j\theta_k$

existe uma refutação-SLD de P e G em que o átomo seleccionado no passo $k-1$ é A_j e o átomo seleccionado no passo k é $A_i\theta'_k$.

Mais, se θ for a resposta calculada de P a G com a refutação-SLD original e θ' for a nova resposta calculada de P a G, então $G\theta$ e $G\theta'$ são variantes.

Prova. O objectivo é construir uma refutação-SLD com sequência de objectivos $G'_0, \ldots, G'_n$, sequência de cláusulas $C'_1, \ldots, C'_n$ e sequência de substituições θ'_n a partir da refutação-SLD original. Para $i < k$ toma-se $G'_i = G_i$, $C'_i = C_i$ e $\theta'_i = \theta_i$.

O passo seguinte é mostrar que se pode seleccionar em G_{k-1} o átomo A_j e tomar $C'_k = C_{k+1}$. Por definição de refutação-SLD, θ_{k+1} é uma UMG de $A_j\theta_k$ e C^+_{k+1}; em particular, $A_j\theta_k\theta_{k+1} = C^+_{k+1}\theta_{k+1}$. Por outro lado, C_{k+1} não contém variáveis que ocorram na derivação até ao passo k, e como θ_k é também uma UMG de átomos que ocorrem na derivação tem-se que $C^+_{k+1}\theta_k = C^+_{k+1}$; então $A_j\theta_k\theta_{k+1} = C^+_{k+1}\theta_k\theta_{k+1}$, donde A_j e C^+_{k+1} são unificáveis com UMG θ'_k. Por definição de UMG, existe ainda uma substituição σ tal que $\theta_k\theta_{k+1} = \theta'_k\sigma$.

Seja G'_k o objectivo derivado de G'_{k-1} e C'_k usando θ'_k. Mostra-se agora que se pode seleccionar em G'_k o átomo $A_i\theta'_k$. Pelo mesmo raciocínio que atrás, θ'_k não afecta as variáveis de C_k, já que estas não ocorrem nem em A_j nem em C^+_{k+1}. Então $C^+_k\theta'_k = C^+_k$, donde se conclui que $C^+_k\sigma = C^+_k\theta'_k\sigma = C_k\theta_k\theta_{k+1} = A_i\theta_k\theta_{k+1} = A_i\theta'_k\sigma$, usando o facto $C^+_k\theta_k = A_i\theta_k$, e portanto $A_i\theta'_k$ e C^+_k são unificáveis com UMG θ'_{k+1} satisfazendo ainda $\theta'_{k+1}\sigma' = \sigma$ para alguma substituição σ'. Por definição de σ, tem-se ainda a relação $\theta_k\theta_{k+1} = \theta'_k\theta'_{k+1}\sigma'$. Por outro lado, $C^+_k\theta'_k\theta'_{k+1} = C^+_k\theta'_{k+1} = A_i\theta'_k\theta'_{k+1}$; como θ_k é UMG de A_i e C^+_k, existe uma substituição ρ tal que $\theta_k\rho = \theta'_k\theta'_{k+1}$. Mas então $A_j\theta_k\rho = A_j\theta'_k\theta'_{k+1} = C^+_{k+1}\theta'_k\theta'_{k+1} = C^+_{k+1}\theta_k\rho = C^+_{k+1}\rho$, donde ρ unifica $A_j\theta_k$ e C^+_{k+1}; como θ_{k+1} é UMG destes dois átomos, existe ρ' tal que $\theta_{k+1}\rho' = \rho$. Logo $\theta'_k\theta'_{k+1} = \theta_k\theta_{k+1}\rho'$.

Prova-se agora por indução que a partir do passo $k+1$ se pode continuar a derivação-SLD escolhendo sempre o mesmo átomo e um variante da mesma cláusula que na derivação-SLD original; mais, existem sequências de derivações σ_m e ρ_m tais que $G_m = G'_m\sigma_m$, $\theta_1 \ldots \theta_m = \theta'_1\theta'_m\sigma_m$, $G'_m = G_m\rho_m$ e $\theta'_1 \ldots \theta'_m = \theta_1 \ldots \theta_m\rho_m$. Para $m = k+1$ basta tomar $\sigma_k = \sigma'$ e $\rho_k = \rho'$ construídos acima.

Num passo m arbitrário, seja A o átomo escolhido em G_m. Então θ_{m+1} é uma UMG de A e C^+_{m+1}. O átomo correspondente A' em G'_m satisfaz $A'\sigma_m = A$; mais uma vez, $C^+_{m+1}\sigma_m =$

C^+_{m+1}, donde $A'\sigma_m\theta_{m+1} = A\theta_{m+1} = C^+_{m+1}\theta_{m+1} = C^+_{m+1}\sigma_m\theta_{m+1}$. Então A' e C^+_{m+1} são unificáveis com UMG θ'_{m+1} e existe σ_{m+1} tal que $\theta'_{m+1}\sigma_{m+1} = \sigma_m\theta_{m+1}$. Segue que $\theta_1 \ldots \theta_m\theta_{m+1} = \theta'_1 \ldots \theta'_m\sigma_m\theta_{m+1} = \theta'_1 \ldots \theta'_m\theta'_{m+1}\sigma_{m+1}$. Por outro lado, os sub-objectivos em G_{m+1} ou são da forma $B\theta_{m+1} = B\sigma_m\theta_{m+1} = B\theta'_{m+1}\sigma_{m+1}$ com B em C^-_{m+1} ou da forma $B\theta_{m+1}$ com B em G_m, caso em que por hipótese de indução $B = B'\sigma_m$ para o átomo B' correspondente de G'_m, e $B'\sigma_m\theta_m = B'\theta_{m+1}\sigma_{m+1}$; segue-se que $G_{m+1} = G'_{m+1}\sigma_{m+1}$.

Finalmente, $A'\rho_m\theta_{m+1} = A\theta_{m+1} = C^+_{m+1}\theta_{m+1} = C^+_{m+1}\rho_m\theta_{m+1}$, donde por definição de UMG existe ρ_{m+1} tal que $\theta'_{m+1}\rho_{m+1} = \rho_m\theta_{m+1}$. Um raciocínio análogo ao anterior mostra que $\theta'_1 \ldots \theta'_m\theta'_{m+1} = \theta_1 \ldots \theta_m\theta_{m+1}\rho_{m+1}$ e que $G'_{m+1} = G_{m+1}\rho_{m+1}$.

Obviamente que $G_n = \square = G'_n$, pelo que este processo permite construir uma refutação-SLD para P e G. Por construção, tem-se ainda que $G\theta_1 \ldots \theta_n$ e $G\theta'_1 \ldots \theta'_n$ são variantes. □

Teorema 3.2.3 *(Independência da regra.)* Sejam P um programa determinado e G um objectivo determinado tais que existe uma refutação-SLD para P e G com resposta calculada σ. Então, para cada função de selecção $\mathcal{S}$ existe uma refutação-SLD de P e G via $\mathcal{S}$ com resposta calculada θ tal que $G\sigma$ e $G\theta$ são variantes.

Prova. Por indução no comprimento da refutação-SLD para P e G com resposta calculada σ. Seja $\mathcal{R}$ a função de selecção utilizada nesta refutação-SLD.

Se a refutação-SLD tiver comprimento 0, então o resultado é trivial.

Suponha-se que a refutação-SLD tem comprimento n. Se $\mathcal{R}(G) = \mathcal{S}(G)$ a hipótese de indução aplicada ao resto da derivação-SLD permite concluir imediatamente o resultado. Caso contrário, sejam $A_i = \mathcal{R}(G)$ e $A_j = \mathcal{S}(G)$. Note-se que A_j terá de ser seleccionado nalgum passo k na refutação-SLD de P a G via $\mathcal{R}$; aplicando o lema da troca k vezes, encontra-se uma refutação-SLD de P a G em que A_j é seleccionado no primeiro passo, os passos 2 a k reproduzem a refutação-SLD de P a G via $\mathcal{R}$ e a resposta calculada ρ é tal que $G\sigma$ e $G\rho$ são variantes. Por hipótese de indução aplicada à refutação-SLD a partir de G_1 (que tem resposta calculada ρ' com $\rho = \theta_1\rho'$) existe uma refutação-SLD de P e G_1 via $\mathcal{S}$ com resposta calculada θ' tal que $G_1\theta'$ e $G_1\rho'$ são variantes. Então $G\theta = G\theta_1\theta'$ e $G\rho = G\theta_1\rho'$ são variantes: os átomos A de G ou são tais que $A\theta_1$ está em G_1, e o resultado é imediato, ou vêm de C^-_1 e o resultado também é imediato. □

É importante salientar que a prova anterior apenas se aplica a refutações-SLD. Em derivações-SLD arbitrárias não há garantia que o átomo A_j seleccionado em G por $\mathcal{S}$ alguma vez o seja na refutação-SLD via $\mathcal{R}$, uma vez que esta pode terminar sem chegar à cláusula vazia (e em particular conter ainda uma instância de A_j) ou ser infinita (e nenhuma instância de A_j ser alguma vez seleccionada).

Observe-se no entanto que a prova do Lema da Troca não tem esta restrição, pelo que o resultado se aplica a qualquer derivação-SLD (exceptuando naturalmente a observação relativa à resposta calculada).

3.3 Adequação computacional

Esta secção mostra um resultado importante do ponto de vista aplicacional: o poder computacional da Programação em Lógica é equivalente ao da máquina de Turing e, portanto, máximo.

Definição 3.3.1 A classe das funções parciais recursivas define-se indutivamente como se segue.

- A função $\lambda x.0$ é parcial recursiva.
- A função $\lambda x.x+1$ é parcial recursiva.
- Para qualquer $n \in \mathbb{N}$ e $i \leq n$, a função $\lambda x_1 \ldots x_n.x_i$ é parcial recursiva.
- Se $f_1, \ldots, f_n : \mathbb{N}^k \to \mathbb{N}$ e $g : \mathbb{N}^n \to N$ são parciais recursivas, então a função

$$\lambda x_1 \ldots x_k.g(f_1(x_1, \ldots, x_k), \ldots, f_n(x_1, \ldots, x_k))$$

  definida por composição a partir de $f_1, \ldots, f_n$ e g também o é.

- Se $f : \mathbb{N}^k \to \mathbb{N}$ e $g : \mathbb{N}^{k+2} \to \mathbb{N}$ são parciais recursivas, então a função

$$h \stackrel{\text{def}}{=} \lambda x_1 \ldots x_{k+1}. \begin{cases} f(x_1, \ldots, x_k) & \text{se } x_{k+1} = 0 \\ g(x_1, \ldots, x_k, x_{k+1} - 1, h(x_1, \ldots, x_k, x_{k+1} - 1)) & \text{se } x_{k+1} > 0 \end{cases}$$

  definida por recursão a partir de f e g também o é.

- Se $f : \mathbb{N}^{k+1} \to \mathbb{N}$ é parcial recursiva, então a função

$$\lambda x_1 \ldots x_k.\mu z(f(x_1, \ldots, x_k, z) = 0)$$

  definida por minimização a partir de f também o é, onde $\mu z(g(z) = 0)$ denota o menor z tal que $g(z) = 0$ e $g(x)$ está definido para $x < z$, ou está indefinido caso não exista nenhum z satisfazendo simultaneamente essas duas condições.

Postulado 3.3.2 *(Church–Markov–Turing.)* A classe das funções parciais recursivas corresponde à classe de funções efectivamente computáveis por algum algoritmo.

Teorema 3.3.3 *(Adequação computacional.)* Seja $f : \mathbb{N}^m \to \mathbb{N}$ uma função parcial recursiva. Então existe um programa determinado P_f contendo um predicado p_f de aridade $m+1$ tal que:

- as respostas calculadas de P_f a $\longleftarrow p_f(s^{k_1}(0), \ldots, s^{k_m}(0), x)$ são todas da forma $\{x/s^k(0)\}$;
- $\{x/s^k(0)\}$ é uma resposta calculada de P_f ao objectivo $\longleftarrow p_f(s^{k_1}(0), \ldots, s^{k_m}(0), x)$ sse $f(k_1, \ldots, k_m) = k$.

Prova. Apresenta-se em primeiro lugar a descrição de como construir o programa determinado P_f por indução na prova de que f é parcial recursiva.

- Para $f = \lambda x.0$, tem-se $P_f = \{p_f(x, 0) \longleftarrow\}$.
- Para $f = \lambda x.x+1$, tem-se $P_f = \{p_f(x, s(x)) \longleftarrow\}$.
- Para $f = \lambda x_1 \ldots x_n.x_i$, tem-se $P_f = \{p_f(x_1, \ldots, x_n, x_i) \longleftarrow\}$.
- Se h for definida por composição a partir de $f_1, \ldots, f_n$ e g, então por hipótese de indução existem programas $P_{f_1}, \ldots, P_{f_n}$ e P_g nas condições do teorema. Então

$$\begin{aligned} P_h \;=\; & P_{f_1} \cup \ldots \cup P_{f_n} \cup P_g \cup \\ & \cup\{p_h(x_1, \ldots, x_k, z) \longleftarrow p_{f_1}(x_1, \ldots, x_k, y_1), \ldots, p_{f_n}(x_1, \ldots, x_k, y_n), p_g(y_1, \ldots, y_n, z)\}. \end{aligned}$$

- Se h for definida por recursão a partir de f e g então por hipótese de indução existem programas P_f e P_g nas condições do teorema. Então

$$\begin{aligned} P_h \;=\; & P_f \cup P_g \cup \\ & \cup\{p_h(x_1, \ldots, x_k, 0, y) \longleftarrow p_f(x_1, \ldots, x_k, y)\} \cup \\ & \cup\{p_h(x_1, \ldots, x_k, s(x), y) \longleftarrow p_h(x_1, \ldots, x_k, x, z), p_g(x_1, \ldots, x_k, x, z, y)\}. \end{aligned}$$

- Se h for definida por minimização a partir de f então por hipótese de indução existe um programa P_f nas condições do teorema. Então

$$\begin{aligned} P_h \;=\; & P_f \cup \\ & \cup\{p_h(x_1,\ldots,x_k,x) \longleftarrow p_f(x_1,\ldots,x_k,x,0), d_f(x_1,\ldots,x_k,x)\} \cup \\ & \cup\{d_f(x_1,\ldots,x_k,0) \longleftarrow\} \cup \\ & \cup\{d_f(x_1,\ldots,x_k,s(x)) \longleftarrow p_f(x_1,\ldots,x_k,x,s(z)), d_f(x_1,\ldots,x_k,x)\}. \end{aligned}$$

É simples provar por indução que se $f(k_1,\ldots,k_m) = k$ então $\{x/s^k(0)\}$ é resposta calculada de P_f a $\longleftarrow p_f(s^{k_1}(0),\ldots,s^{k_m}(0),x)$. A prova de que esta é a *única* resposta calculada de P_f faz-se também por indução mostrando que esta é a única resposta correcta de P_f àquele objectivo e aplicando o Teorema de Clark. □

3.4 Árvores-SLD

O objectivo desta secção é interpretar em termos práticos os resultados teóricos que apresentados neste capítulo. A semântica operacional da Programação em Lógica fornece uma interpretação computacional de programas determinados, através do cálculo de respostas a objectivos determinados; porém, conforme se verá, há questões de implementação que não são triviais.

Fixada uma função de selecção, há várias possibilidades de construção de uma refutação-SLD consoante (o variante d)a cláusula do programa que se escolhe.

Definição 3.4.1 Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção. A *árvore-SLD* de P e G via $\mathcal{S}$ é uma árvore etiquetada construída da seguinte forma:

- a etiqueta de cada nó é um objectivo;
- a etiqueta da raiz é G;
- cada nó com etiqueta $G' \equiv \longleftarrow A_1,\ldots,A_p$ tem um descendente por cada cláusula $C \equiv A \longleftarrow B_1,\ldots,B_q$ de P cuja cabeça seja unificável com $\mathcal{S}(\longleftarrow A_1,\ldots,A_p) = A_m$; sendo θ uma UMG de A e A_m, o descendente correspondente é etiquetado pelo resolvente de C e G' usando θ.

Um ramo cuja folha está etiquetada pela cláusula vazia □ diz-se *bem sucedido*. Um ramo com uma folha etiquetada com outra cláusula diz-se *falhado*. Os restantes ramos dizem-se ramos *infinitos*.

Em particular, os nós etiquetados com □ não têm descendentes.

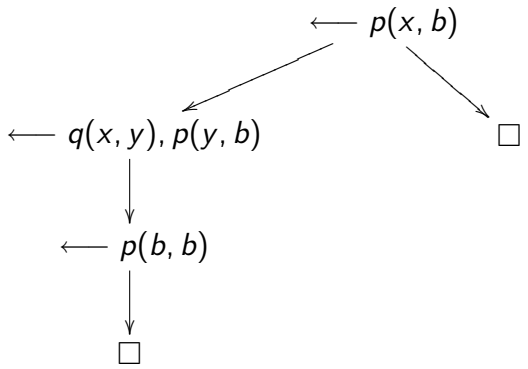
Uma árvore-SLD é uma forma de representar todas as derivações-SLD possíveis de P e G via $\mathcal{S}$. Observe-se que os ramos bem sucedidos correspondem precisamente às refutações-SLD de P e G via $\mathcal{S}$.

Exemplo 3.4.2 Seja P o programa determinado seguinte.

$$\begin{aligned} & p(x,z) \longleftarrow q(x,y), p(y,z) \\ & p(x,x) \longleftarrow \\ & q(a,b) \longleftarrow \end{aligned}$$

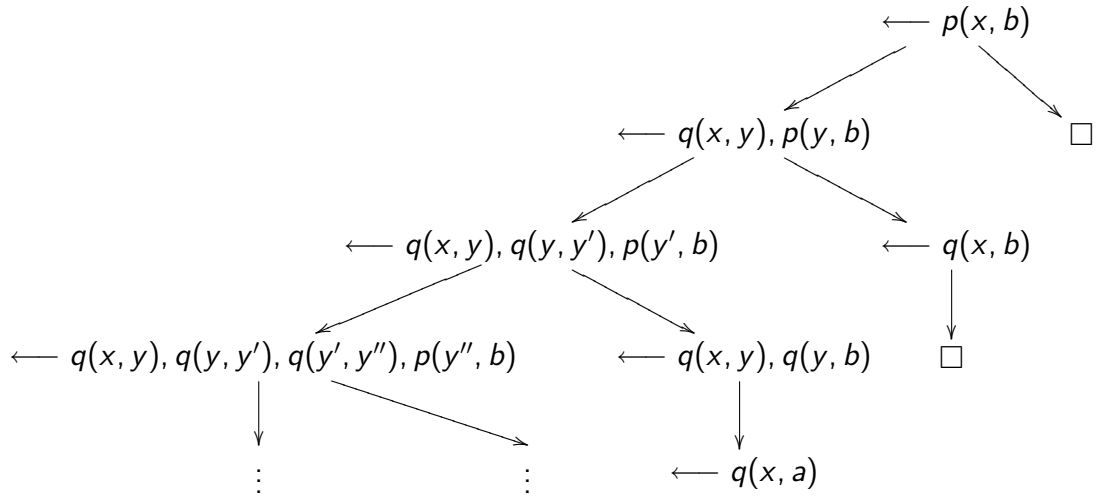
Considere-se o objectivo $G \equiv \longleftarrow p(x, b)$.

Seja $\mathcal{S}_1$ uma função de selecção tal que $\mathcal{S}_1(\longleftarrow A_1, \ldots, A_q) = A_1$. A árvore-SLD de P e G via $\mathcal{S}_1$ é a seguinte, onde os descendentes de cada nó aparecem pela mesma ordem das cláusulas de P cuja cabeça é unificável com o átomo seleccionado.



O ramo da esquerda corresponde a uma refutação-SLD com resposta calculada $\{x/a\}$; o ramo da direita corresponde a uma refutação-SLD com resposta calculada $\{x/b\}$.

Seja agora $\mathcal{S}_2$ uma função de selecção tal que $\mathcal{S}_1(\longleftarrow A_1, \ldots, A_q) = A_q$. A árvore-SLD de P e G via $\mathcal{S}_2$ é a seguinte.



O ramo mais à direita corresponde novamente a uma refutação-SLD com resposta calculada $\{x/b\}$; o outro ramo bem sucedido corresponde a uma refutação-SLD com resposta calculada $\{x/a\}$. Esta árvore tem ainda um ramo infinito e uma infinidade de ramos falhados.

Observe-se que o número de ramos bem sucedidos coincide em ambas as árvores e que as respostas calculadas são as mesmas. Mostrar-se-á na sequência que este facto é perfeitamente geral. Este exemplo mostra ainda que o mesmo não se verifica para os ramos falhados ou infinitos.

O seguinte resultado é a interpretação da Independência da Regra em termos de árvores-SLD.

Proposição 3.4.3 Sejam P um programa determinado e G um objectivo determinado. Então todas as árvores-SLD de P e G têm o mesmo número (finito ou infinito) de ramos bem sucedidos.

Prova. Sejam $\mathcal{A}_1$ e $\mathcal{A}_2$ árvores-SLD para P e G com funções de selecção respectivamente $\mathcal{S}_1$ e $\mathcal{S}_2$. Cada ramo bem sucedido de $\mathcal{A}_1$ corresponde a uma refutação-SLD de P a G via $\mathcal{S}_1$; pela

Independência da Regra, a esta refutação-SLD corresponde uma outra refutação-SLD de P a G via $\mathcal{S}_2$, à qual corresponde por sua vez um ramo em $\mathcal{A}_2$.

Uma análise da prova do Lema da Troca mostra ainda que esta correspondência é unívoca, ou seja, a refutações-SLD distintas via $\mathcal{S}_1$ correspondem refutações-SLD distintas via $\mathcal{S}_2$ (uma vez que o único factor em que as refutações-SLD podem diferir é na escolha das cláusulas de P a utilizar a cada passo). Então a cada ramo bem sucedido de $\mathcal{A}_1$ corresponde univocamente um ramo bem sucedido de $\mathcal{A}_2$; o raciocínio em sentido inverso permite concluir que o número de ramos bem sucedidos em ambas as árvores é igual. □

Em termos computacionais, o interesse destes resultados é grande: essencialmente, estão a afirmar que a qualidade duma implementação da resolução-SLD é independente da função de selecção escolhida, pelo menos em termos de respostas que podem ser calculadas. É por este motivo que a esmagadora maioria das implementações utiliza a função de selecção "escolher o átomo mais à esquerda": é muitas vezes a mais eficiente (por exemplo, se os objectivos estiverem implementados como listas dinâmicas) de entre várias opções com igual poder computacional.

Importa salientar que, contrariamente ao que o Exemplo 3.4.2 poderá dar a entender, não é relevante comparar funções de selecção em termos de ramos falhados vs ramos infinitos. É fácil alterar o programa determinado do exemplo por forma a que a árvore-SLD via $\mathcal{S}_1$ seja infinita e a árvore via $\mathcal{S}_2$ seja finita. Em geral, para cada programa haverá funções de selecção que dão árvores-SLD finitas e outras que dão árvores-SLD infinitas, mas quando as regras são puramente estruturais (estilo "seleccionar o átomo mais à esquerda") não há grandes esperanças de conseguir melhor.

No capítulo 5 voltar-se-á à questão de como encontrar respostas calculadas de um programa determinado a um objectivo determinado.

Capítulo 4

Ordinais e Pontos Fixos

O objectivo do próximo capítulo é desenvolver o estudo da semântica denotacional da Programação em Lógica. Este capítulo resume os conceitos e resultados da teoria de reticulados que serão necessários para esse estudo.

4.1 Relações binárias e reticulados

Definição 4.1.1 Uma *relação binária* R num conjunto A é um conjunto de pares de elementos de A.

Por outras palavras, uma relação binária em A é um subconjunto $R \subseteq A \times A$.

Notação 4.1.2 Utiliza-se frequentemente a notação $\langle A, R \rangle$ para especificar simultaneamente a relação binária R e o conjunto A em que R está definida. É também usual escrever $x\ R\ y$ para denotar o facto $\langle x, y \rangle \in R$.

Definição 4.1.3 Uma relação binária R num conjunto A diz-se:

- *reflexiva* se $x\ R\ x$ para qualquer $x \in A$;
- *simétrica* se sempre que $x\ R\ y$ se tiver também $y\ R\ x$, para quaisquer $x, y \in A$;
- *anti-simétrica* se $x\ R\ y$ e $y\ R\ x$ implicar que $x = y$, para quaisquer $x, y \in A$;
- *transitiva* se de $x\ R\ y$ e $y\ R\ z$ se concluir que $x\ R\ z$, para quaisquer $x, y, z \in A$.

Definição 4.1.4 Uma relação binária diz-se uma *pré-ordem* se for reflexiva e transitiva.

- Uma pré-ordem anti-simétrica diz-se uma *ordem parcial*.
- Uma pré-ordem simétrica diz-se uma *relação de equivalência*.

Exemplo 4.1.5 Em qualquer conjunto A podem-se definir três relações binárias:

- a *relação diagonal* Δ_A, em que $x\ \Delta_A\ y$ sse $x = y$;
- a *relação universal* U_A, em que $x\ U_A\ y$ para quaisquer $x, y \in A$;
- a *relação vazia* $\emptyset_A$, em que nunca se tem $x\ \emptyset_A\ y$.

Quer Δ_A quer U_A são relações de equivalência, enquanto que $\emptyset_A$ não é sequer uma pré-ordem. No entanto, $\emptyset_A$ é simétrica, anti-simétrica e transitiva. Por outro lado, Δ_A é também uma ordem parcial (dita *discreta*).

Exemplo 4.1.6 Relações binárias bem conhecidas sobre o conjunto $\mathbb{N}$ dos naturais são por exemplo as seguintes.

- A relação de "menor que", $x < y$ sse $y = x + z + 1$ para algum natural z.
- A relação de "menor ou igual", $x \leq y$ sse $x < y$ ou $x = y$.
- A relação de igualdade, $x = y$ (que é simplesmente $\Delta_{\mathbb{N}}$).

A relação $\leq$ é uma ordem parcial, enquanto que $=$ é uma relação de equivalência. Quanto a $<$, é uma relação transitiva e anti-simétrica, mas não reflexiva.

Exemplo 4.1.7 Seja A um conjunto e $\wp A$ o conjunto das partes de A (ou seja, $X \in \wp A$ sse $X \subseteq A$). A relação $\langle \wp A, \subseteq \rangle$ é uma ordem parcial.

Notação 4.1.8 Atendendo aos exemplos anteriores, é usual denotar relações binárias genéricas por R, S, R_1, R'; ordens parciais (e por vezes pré-ordens) por $\leq$, $\preceq$, $\sqsubseteq$; e relações de equivalência por $=$, $\equiv$ ou $\approx$.

O exemplo seguinte, mais específico ao contexto da Programação em Lógica, ilustra bem a generalidade destes conceitos.

Exemplo 4.1.9 Seja S um conjunto finito de expressões simples e $\Theta_S = \{\theta \mid \#(S\theta) = 1\}$ o conjunto de todas as unificadoras de S. Defina-se em Θ_S uma relação R por $\theta_1 \; R \; \theta_2$ sse $\theta_1 = \theta_2\gamma$ para alguma substituição γ.

Então o par $\langle \Theta_S, R \rangle$ é uma pré-ordem. De facto:

- para qualquer unificadora $\theta \in \Theta_S$, tem-se que $\theta = \theta\varepsilon$, logo $\theta \; R \; \theta$;
- se $\theta_1 \; R \; \theta_2$ e $\theta_2 \; R \; \theta_3$, então existem substituições γ e δ tais que $\theta_1 = \theta_2\gamma$ e $\theta_2 = \theta_3\delta$, de onde sai imediatamente que $\theta_1 = (\theta_3\delta)\gamma = \theta_3(\delta\gamma)$, e portanto $\theta_1 \; R \; \theta_3$.

Definição 4.1.10 Seja $\approx$ uma relação de equivalência num conjunto A. A *classe de equivalência* de um elemento $x \in A$ é o conjunto

$$[x]_{\approx} = \{y \in A \mid x \approx y\}.$$

É frequente escrever simplesmente $[x]$ quando a relação $\approx$ é evidente.

Definição 4.1.11 Seja $\approx$ uma relação de equivalência num conjunto A. O *quociente* $A/\!\approx$ de A por $\approx$ é o conjunto das classes de equivalência de elementos de A.

$$A/\!\approx = \{[x]_{\approx} \mid x \in A\}$$

A designação de pré-ordem dá a entender que de alguma forma existe uma ordem parcial implícita numa tal relação. A proposição seguinte justifica esta nomenclatura.

Proposição 4.1.12 Seja R uma pré-ordem num conjunto A. Então:

- a relação $=_R$ em A definida por $x =_R y$ sse $x \; R \; y$ e $y \; R \; x$ é uma relação de equivalência;

- a relação $\leq_R$ em $A/=_R$ definida por $[x] \leq_R [y]$ sse $x\ R\ y$ é uma ordem parcial.

Prova. Sejam $=_R$ e $\leq_R$ definidas como acima. Observe-se que R, sendo pré-ordem, é reflexiva e transitiva.

Prova-se em primeiro lugar que $=_R$ é relação de equivalência.

- *Reflexividade.* Como R é reflexiva, para qualquer $x \in A$ tem-se $x\ R\ x$, donde $x =_R x$.
- *Simetria.* A condição de definição de $=_R$ é simétrica em x e y, logo se $x =_R y$ forçosamente se tem $y =_R x$.
- *Transitividade.* Se $x =_R y$, então (1) $x\ R\ y$ e (2) $y\ R\ z$; se $y =_R z$, então (3) $y\ R\ z$ e (4) $z\ R\ y$. Por transitividade de R, conclui-se de (1) e (3) que $x\ R\ z$, enquanto que de (4) e (2) sai que $z\ R\ x$. Logo $x =_R z$.

Por conseguinte, o quociente $A/=_R$ está bem definido. O próximo passo é ver que a definição de $\leq_R$ faz sentido: se $[x]$ e $[x']$ denotarem uma mesma classe de equivalência e $[y]$ e $[y']$ denotarem uma outra classe de equivalência, então a condição que define $[x] \leq_R [y]$ é equivalente à que define $[x'] \leq_R [y']$.

Sejam x, x', y e y' nestas condições. Então $x\ R\ x'$, $x'\ R\ x$, $y\ R\ y'$ e $y'\ R\ y$. Então se $[x] \leq_R [y]$ tem-se por definição $x\ R\ y$; por transitividade, de $x'\ R\ x\ R\ y\ R\ y'$ conclui-se que $x'\ R\ y'$, donde $[x'] \leq_R [y']$. A prova no sentido inverso é análoga.

Finalmente mostra-se que $\leq_R$ é uma ordem parcial em $A/=_R$.

- *Reflexividade.* Como R é reflexiva, para qualquer $x \in A$ tem-se $x\ R\ x$, donde $[x] \leq_R [x]$.
- *Anti-simetria.* Se $[x] \leq_R [y]$ e $[y] \leq_R [x]$, então $x\ R\ y$ e $y\ R\ x$; por definição de $=_R$, conclui-se que $x =_R y$, donde $[x] = [y]$.
- *Transitividade.* Se $[x] \leq_R [y]$, então $x\ R\ y$; se $[y] \leq_R [z]$, então $y\ R\ z$. Por transitividade de R, conclui-se que $x\ R\ z$, donde $[x] \leq_R [z]$.

□

Definição 4.1.13 Sejam $\langle A, \preceq \rangle$ uma pré-ordem e $X \subseteq A$.

- Um elemento $a \in A$ diz-se um *majorante* de X se $x \preceq a$ para todo o $x \in X$.
- Um elemento $a \in A$ diz-se um *minorante* de X se $a \preceq x$ para todo o $x \in X$.
- Um elemento $a \in A$ diz-se um *supremo* de X se a for um majorante de X e se tiver $a \preceq m$ para todo o majorante m de X.
- Um elemento $a \in A$ diz-se um *ínfimo* de X se a for um minorante de X e se tiver $m \preceq a$ para todo o minorante m de X.

Exemplo 4.1.14 Na pré-ordem das unificadoras sobre um conjunto S de expressões simples (Exemplo 4.1.9), as UMGs de S são supremos de Θ_S.

De facto, se σ é uma UMG de S e θ é qualquer outra unificadora de S (ou seja, se $\theta \in \Theta_S$), então existe uma substituição γ tal que $\theta = \sigma\gamma$. Por definição de R, tem-se que $\theta\ R\ \sigma$, donde σ é majorante de Θ_S. Por outro lado, como $\sigma \in \Theta_S$, qualquer outro majorante σ' de Θ_S satisfaz necessariamente $\sigma\ R\ \sigma'$; logo σ é um supremo de Θ_S.

Reciprocamente, qualquer majorante de Θ_S é uma UMG de S: por definição, um majorante σ de Θ_S satisfaz $\theta\ R\ \sigma$ para qualquer $\theta \in \Theta_S$; um σ nestas condições é uma unificadora de S tal

que para qualquer outra unificadora θ de S existe uma substituição γ com $\theta = \sigma\gamma$. Logo σ é uma UMG de S.

Na presença de anti-simetria, os supremos e ínfimos ganham um estatuto especial.

Proposição 4.1.15 Sejam $\langle A, \preceq \rangle$ uma ordem parcial e $X \subseteq A$. Se X tiver um supremo (ínfimo), então esse supremo (ínfimo) é único.

Prova. Sejam a e a' dois supremos de X. Como a é supremo de X e a' é majorante de X, tem-se que $a \preceq a'$; como a' é supremo de X e a é majorante de X, tem-se que $a' \preceq a$. Como $\preceq$ é uma ordem parcial, $a = a'$.

A prova para ínfimos é análoga. □

Notação 4.1.16 Atendendo à proposição anterior, em ordens parciais pode-se falar n' "o supremo" de X e n' "o" ínfimo de X. Quando estes existirem, serão denotados respectivamente por $\sup(X)$ e $\inf(X)$.

Definição 4.1.17 Um *reticulado completo* é uma ordem parcial $\langle A, \preceq \rangle$ em que todo o conjunto X tem supremo e ínfimo.

Em particular, num reticulado completo existem sempre um elemento *topo* $\top = \sup(A)$ e *base* $\bot = \inf(S)$.

Em virtude da existência de elementos topo e base, um reticulado completo nunca pode ser vazio. No entanto a base e o topo podem coincidir, caso em que o reticulado tem apenas um elemento: é o caso de $\langle \{*\}, U_{\{*\}} \rangle$. Em geral tem-se a relação $\bot \preceq a \preceq \top$ para qualquer $a \in A$.

Exemplo 4.1.18 A ordem parcial dos subconjuntos dum conjunto A (Exemplo 4.1.7) é um reticulado completo. Se $\{X_i\}_{i \in I}$ for um conjunto de subconjuntos de A (ou seja, $X_i \subseteq A$ para todo o $i \in I$), então $\bigcup_{i \in I} X_i$ é um supremo de $\{X_i\}_{i \in I}$ e $\bigcap_{i \in I} X_i$ é um ínfimo de $\{X_i\}_{i \in I}$.

4.2 Ordinais e indução transfinita

Para o resto deste capítulo, é necessário ter algumas noções de teoria de ordinais. Nesta secção resumem-se os resultados necessários desta teoria, sem entrar em pormenores.

Assume-se como conhecida a definição dos números naturais em teoria dos conjuntos:

$$\begin{array}{lcll} 0 & = & \emptyset & \\ 1 & = & \{\emptyset\} & = \{0\} \\ 2 & = & \{\emptyset, \{\emptyset\}\} & = \{0, 1\} \\ 3 & = & \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} & = \{0, 1, 2\} \\ \vdots & & & \\ n+1 & = & \{0, 1, \ldots, n\} = n \cup \{n\} & \\ \vdots & & & \end{array}$$

Denota-se por ω a união de todos os números naturais: $\omega = \{0, 1, \ldots, n, \ldots\}$.

Continuando a sucessão anterior, é perfeitamente razoável definir $\omega + 1 = \omega \cup \{\omega\}$, e assim sucessivamente. Tendo definido $\omega + n$ para todo o n, é natural definir $\omega 2 = \{\omega + n \mid n \in \omega\}$; continuando o processo, definem-se $\omega 3$, $\omega 4$, ωn para qualquer n, e $\omega^2 = \{\omega n \mid n \in \omega\}$.

A teoria dos ordinais é o ramo da teoria de conjuntos que define formalmente e estuda as propriedades destes objectos. Em particular, é possível estender as operações usuais sobre os naturais (soma, produto, exponenciação) a ordinais genéricos, embora se percam algumas das suas propriedades. A título de exemplo, a multiplicação de ordinais não é comutativa, donde a designação de $\omega 2$ para o segundo ordinal limite e não de 2ω (aliás, mostra-se que $2\omega = \omega$). No entanto essas questões transcendem o estudo que neste momento se pretende fazer de ordinais.

Definição 4.2.1 Seja α um ordinal. Diz-se que α é:

- um ordinal *sucessor* se $\alpha = \beta + 1$ para algum ordinal β;
- um ordinal *limite* caso contrário.

Os números naturais, com excepção do 0, são todos ordinais sucessor, como o são também $\omega + 3$ ou $\omega^2 + \omega 3 + 55$. Exemplos de ordinais limite são 0, ω, $\omega 2$ ou $\omega^2 + \omega 3$.

Definição 4.2.2 Seja α um ordinal. Diz-se que α é:

- um ordinal *finito* se $\alpha \in \omega$;
- um ordinal *infinito* caso contrário.

Segue destas duas definições que o único ordinal limite finito é 0.

Definição 4.2.3 Definem-se ordenações nos ordinais por $\alpha < \beta$ sse $\alpha \in \beta$ e $\alpha \leq \beta$ sse $\alpha \subseteq \beta$.

Observe-se que $\alpha \leq \beta$ sse $\alpha < \beta$ ou $\alpha = \beta$.

Proposição 4.2.4 A relação $\leq$ na classe de todos os ordinais é uma ordem parcial em que todo o conjunto tem ínfimo.

Prova. A reflexividade, transitividade e anti-simetria de $\leq$ são consequência dessas mesmas propriedades para a relação de inclusão. Dado um conjunto $\{X_i\}_{i \in I}$ de ordinais, a sua intersecção $\bigcap_{i \in I} X_i$ é o ínfimo desse conjunto. □

Note-se que a *união* de uma classe de ordinais não tem de ser um ordinal (em particular, a união de todos os ordinais *não* é um ordinal), pelo que não se tem neste caso um reticulado completo.

Para provar uma propriedade P dos ordinais utiliza-se o seguinte *Princípio de Indução Transfinita* (PIT).

De $\forall \alpha.(\forall \beta{<}\alpha.P(\beta)) \Rightarrow P(\alpha)$ conclui-se $\forall \alpha.P(\alpha)$.

Pode-se reescrever este princípio duma forma mais aplicável observando que todos os ordinais são ordinais sucessor ou ordinais limite. Se α é um ordinal limite, tem-se que

$$\alpha = \bigcup_{\beta < \alpha} \beta;$$

é ainda conveniente tratar separadamente o caso $\alpha = 0$ (em que a união anterior é vazia), obtendo-se a seguinte variante do princípio.

De

- $P(0)$
- $\forall\alpha.P(\alpha) \Rightarrow P(\alpha+1)$
- $\forall\alpha.(\forall\beta{<}\alpha.P(\beta)) \Rightarrow P(\alpha)$ desde que α seja um ordinal limite não nulo

conclui-se $\forall\alpha.P(\alpha)$.

É a esta versão do PIT que se recorrerá no seguimento.

4.3 Pontos fixos de transformações monótonas

O objectivo principal deste capítulo é provar dois teoremas sobre pontos fixos de transformações em reticulados completos: um teorema muito geral sobre a existência de pontos fixos e uma versão mais forte que fornece um método de cálculo de pontos fixos de uma classe um pouco mais restrita de transformações.

Definição 4.3.1 Uma *transformação* num reticulado completo $\langle A, \preceq\rangle$ é uma função $T : A \to A$.

O interesse de estudar transformações em reticulados (e não simplesmente sobre conjuntos) reside na interacção entre a transformação e a relação de ordem. Assim, têm particular interesse as transformações que satisfazem a seguinte propriedade.

Definição 4.3.2 Uma transformação T num reticulado completo $\langle A, \preceq\rangle$ diz-se *monótona* se $T(a) \preceq T(b)$ sempre que $a \preceq b$.

Definição 4.3.3 Um *ponto fixo* de uma transformação T num reticulado completo $\langle A, \preceq\rangle$ é um elemento $a \in A$ tal que $T(a) = a$.

Um ponto fixo a de T diz-se o ponto fixo *mínimo* de T se $a \preceq b$ para todo o ponto fixo b de T e diz-se o ponto fixo *máximo* de T se $b \preceq a$ para todo o ponto fixo b de T.

Teorema 4.3.4 *(Tarski.)* Seja T uma transformação monótona num reticulado completo $\langle A, \preceq\rangle$. Então T tem um ponto fixo mínimo pfmin(T) e um ponto fixo máximo pfmax(T) satisfazendo às seguintes propriedades.

$$\begin{array}{rcccl} \text{pfmin}(T) & = & \inf\{x \mid T(x) = x\} & = & \inf\{x \mid T(x) \preceq x\} \\ \text{pfmax}(T) & = & \sup\{x \mid T(x) = x\} & = & \sup\{x \mid x \preceq T(x)\} \end{array}$$

Prova. Apresenta-se a parte da demonstração relativa ao ponto fixo máximo, já que o outro caso é análogo. Sejam $X = \{x \mid x \preceq T(x)\}$, $Y = \{x \mid x = T(x)\}$ e tome-se $a = \sup X$, que existe porque A é um reticulado completo.

Em primeiro lugar, mostra-se que a é ponto fixo de T. Tome-se $x \in X$; então $x \preceq T(x)$. Por definição de a, tem-se que $x \preceq a$; como T é monótona, tem-se que $T(x) \preceq T(a)$, donde $x \preceq T(a)$ por transitividade. Então $T(a)$ é um majorante de X, logo $a \preceq T(a)$. Aplicando novamente a monotonia de T, de $a \preceq T(a)$ sai que $T(a) \preceq T(T(a))$; então $T(a) \in X$, donde $T(a) \preceq a$ uma vez que a majora X. Por anti-simetria conclui-se que $T(a) = a$, ou seja, a é ponto fixo de T.

Por construção, $a = \sup X$; por outro lado, se $y \in Y$ (ou seja, $y = T(y)$) então $y \preceq T(y)$ por reflexividade de $\preceq$, donde a é majorante de Y. Uma vez que $a = T(a)$, se b for outro majorante de Y ter-se-á necessariamente $a \preceq b$, logo a também é supremo de Y.

Finalmente, seja b outro ponto fixo de T. Então $b = T(b)$, logo $b \in Y$ e portanto $b \preceq a$ uma vez que a é majorante de Y.

Logo a é o ponto fixo máximo de T. □

Corolário 4.3.5 Sejam $\langle A, \preceq \rangle$ um reticulado completo e T uma transformação monótona em A. Se $x \in A$ é tal que $x \preceq T(x)$, então existe um ponto fixo a de T tal que $a \preceq x$. Se $y \in A$ é tal que $T(y) \preceq y$, então existe um ponto fixo b de T tal que $y \preceq b$.

Prova. Basta tomar $a = \text{pfmin}(T)$ e $b = \text{pfmax}(T)$ e aplicar o Teorema de Tarski. □

Embora o Teorema de Tarski seja um resultado bastante geral e razoavelmente forte, a sua aplicabilidade prática é algo discutível: por um lado, não é fácil em geral calcular supremos ou ínfimos de conjuntos arbitrários; por outro, o próprio conjunto $\{x \mid x \preceq T(x)\}$ também está longe de ser um objecto trivial. O Teorema de Tarski é útil como ferramenta em situações em que baste saber que *existe* um ponto fixo duma transformação, mas não é preciso calculá-lo; noutras situações, são necessários resultados mais fortes, como os que se apresentam de seguida.

Definição 4.3.6 Seja T uma transformação monótona num reticulado completo $\langle A, \preceq \rangle$. Definem-se as *potências ascendentes* $T \uparrow \alpha$ e as *potências descendentes* $T \downarrow \alpha$ de T como se segue.

$$\begin{array}{ll} T \uparrow 0 = \bot & T \downarrow 0 = \top \\ T \uparrow (\alpha + 1) = T(T \uparrow \alpha) & T \downarrow (\alpha + 1) = T(T \downarrow \alpha) \\ T \uparrow \beta = \sup\{T \uparrow \alpha \mid \alpha < \beta\} & T \downarrow \beta = \inf\{T \uparrow \alpha \mid \alpha < \beta\} \end{array}$$

onde na última cláusula se assume que β é um ordinal limite.

Os seguintes resultados justifica a nomenclatura e notação utilizada para as potências ascendentes e descendentes.

Lema 4.3.7 Para todo o ordinal γ, tem-se $T \uparrow \gamma \preceq T \uparrow (\gamma + 1)$.

Prova. Por indução transfinita em γ.

- Se $\gamma = 0$, então $T \uparrow 0 = \bot \preceq T \uparrow 1$ por definição de base do reticulado.
- Suponha-se que $T \uparrow \gamma \preceq T \uparrow (\gamma + 1)$; por monotonia de T conclui-se que $T \uparrow (\gamma + 1) = T(T \uparrow \gamma) \preceq T(T \uparrow (\gamma + 1)) = T \uparrow (\gamma + 2)$.
- Suponha-se que $T \uparrow \delta \preceq T \uparrow \delta + 1$ para todo o $\delta < \gamma$. Por definição de supremo, $T \uparrow \delta \preceq T \uparrow \gamma$ para qualquer $\delta < \gamma$; por monotonia de T conclui-se como atrás que $T \uparrow (\delta + 1) \preceq T \uparrow (\gamma + 1)$ e por transitividade $T \uparrow \delta \preceq T \uparrow (\gamma + 1)$. Então $T \uparrow (\gamma + 1)$ é majorante de $\{T \uparrow \delta \mid \delta < \gamma\}$; por definição de supremo, $T \uparrow \gamma \preceq T \uparrow (\gamma + 1)$.

□

Proposição 4.3.8 Seja T uma transformação monótona num reticulado completo $\langle A, \preceq \rangle$. Se $\alpha < \beta$, então $T \uparrow \alpha \preceq T \uparrow \beta$ e $T \downarrow \beta \preceq T \downarrow \alpha$.

Prova. Prova-se que se $\alpha \leq \beta$ então $T \uparrow \alpha \preceq T \uparrow \beta$ por indução transfinita em β.

- Suponha-se que $\beta = 0$. Então $\alpha \leq \beta$ sse $\alpha = 0 = \beta$ e a tese sai por reflexividade de $\preceq$.
- Suponha-se que $\beta = \gamma + 1$. Então ou $\alpha = \gamma + 1$ ou $\alpha \leq \gamma$. No primeiro caso, $T \uparrow \alpha = T \uparrow (\gamma + 1)$ e aplica-se de novo a reflexividade de $\preceq$; no segundo, por hipótese de indução $T \uparrow \alpha \preceq T \uparrow \gamma$ e pelo lema $T \uparrow \gamma \preceq T \uparrow (\gamma + 1)$, donde por transitividade sai de novo que $T \uparrow \alpha \preceq T \uparrow (\gamma + 1)$.
- Suponha-se que β é um ordinal limite. Se $\alpha = \beta$ o resultado sai de novo por reflexividade de $\preceq$; se $\alpha < \beta$ a tese sai por definição de supremo.

O outro caso é semelhante recorrendo a uma versão análoga do lema anterior. □

Proposição 4.3.9 Seja T uma transformação monótona num reticulado completo $\langle A, \preceq \rangle$. Então $T \uparrow \alpha \preceq \text{pfmin}(T)$ e $\text{pfmax}(T) \preceq T \downarrow \alpha$ para todo o ordinal α.

Prova. Prova-se de novo apenas a primeira parte por indução transfinita em α.

- Se $\alpha = 0$, então $T \uparrow \alpha = \bot \preceq \text{pfmin}(T)$ por definição de base.
- Suponha-se que $T \uparrow \alpha \preceq \text{pfmin}(T)$. Por monotonia de T segue que $T \uparrow (\alpha + 1) \preceq T(\text{pfmin}(T)) = \text{pfmin}(T)$ por definição de ponto fixo.
- Suponha-se que $T \uparrow \gamma \preceq \text{pfmin}(T)$ para todo o $\gamma < \alpha$. Então $\text{pfmin}(T)$ é majorante de $\{T \uparrow \gamma \mid \gamma < \alpha\}$, logo $T \uparrow \alpha \leq \text{pfmin}(T)$ por definição de supremo.

□

Com base nestes resultado podemos obter uma primeira versão mais "computacional" do Teorema de Tarski.

Proposição 4.3.10 Seja T uma transformação monótona num reticulado completo $\langle A, \preceq \rangle$. Então existem um ordinal α tal que $T \uparrow \gamma = \text{pfmin}(T)$ se $\alpha \leq \gamma$ e um ordinal β tal que $T \downarrow \gamma = \text{pfmax}(T)$ de $\beta \leq \gamma$.

Prova. Em primeiro lugar, observe-se que se existirem dois ordinais $\gamma < \delta$ tais que $T \uparrow \gamma = T \uparrow \delta$, então $T \uparrow \gamma = \text{pfmin}(T)$. De facto, se $\gamma < \delta$, então pela Proposição 4.3.8 tem-se que $T \uparrow \gamma \preceq T \uparrow (\gamma + 1)$ e $T \uparrow (\gamma + 1) \preceq T \uparrow \delta = T \uparrow \gamma$; por anti-simetria de $\preceq$ conclui-se que $T \uparrow \gamma = T \uparrow (\gamma + 1)$, e portanto $T \uparrow \gamma$ é ponto fixo de T. Como $T \uparrow \gamma \preceq \text{pfmin}(T)$ pela Proposição 4.3.9, tem-se que $T \uparrow \gamma = \text{pfmin}(T)$.

Seja agora β um ordinal tal que $\#\beta > \#A$; a prova da existência de um tal ordinal transcende o âmbito desta disciplina. Suponha-se por absurdo que $T \uparrow \delta \neq \text{pfmin}(T)$ para todo o $\delta < \beta$; então a aplicação h definida por $h(\delta) = T \uparrow \delta$ é injectiva em virtude da observação acima. Mas isto é absurdo: por monotonia de T o número de elementos em $\{T \uparrow \delta \mid \delta < \beta\}$ é no máximo $\#S$, enquanto que por injectividade de h o cardinal daquele conjunto é $\#\beta$. Logo h não é injectiva, e portanto existe um ordinal α (mínimo) para o qual $T \uparrow \alpha = \text{pfmin}(T)$.

Finalmente suponha-se que $\alpha \leq \gamma$. Então $\text{pfmin}(T) = T \uparrow \alpha \leq T \uparrow \gamma \leq \text{pfmin}(T)$ atendendo às Proposições 4.3.8 e 4.3.9. Logo α é o ordinal pretendido.

De uma forma análoga provar-se-ia a existência do ordinal β nas condições do enunciado. □

Definição 4.3.11 Ao ordinal α da proposição anterior chama-se *ordinal de fecho* de T.

Embora a existência de um ordinal de fecho seja algo mais forte que o Teorema de Tarski, já que indica que o ponto fixo mínimo (máximo) pode ser obtido apenas através do cálculo de potências ascendentes (descendentes) da transformação T, computacionalmente continua a ser um resultado fraco: nada é dito sobre a cardinalidade de α, que pode não ser numerável; e mesmo que o seja, a computação de $T \uparrow (\omega^\omega + \omega^2 + 5)$ está longe de ser trivial... No seguimento restringir-se-á a atenção a operadores satisfazendo uma propriedade extra, que se mostrará satisfazerem um teorema de ponto fixo muito mais poderoso.

Em primeiro lugar, é necessário introduzir mais alguns conceitos.

Definição 4.3.12 Seja $\langle S, \preceq \rangle$ um reticulado completo. Um conjunto $X \subseteq S$ diz-se *orientado* se todo o subconjunto finito de X tem um majorante em X.

De forma equivalente: X é orientado sse, para quaisquer $x_1, \ldots, x_n \in X$ existir $y \in X$ tal que $x_i \preceq y$ para $i = 1, \ldots, n$.

Em particular, se X é orientado, então $X \neq \emptyset$, já que X contém sempre um majorante do conjunto vazio.

Definição 4.3.13 Uma T transformação num reticulado completo $\langle S, \preceq \rangle$ diz-se *contínua* se para qualquer $X \subseteq S$ orientado se tiver $\sup(T(X)) = T(\sup(X))$.

A noção de continuidade é uma noção topológica; embora neste contexto isso não seja nada óbvio, de facto as transformações contínuas são contínuas relativamente a uma topologia em particular, a chamada *topologia de Scott* no reticulado completo $\langle S, \preceq \rangle$. Esta topologia – cujo estudo está fora do âmbito desta disciplina – é definida à custa da relação de ordem, estando ligada à noção de conjunto orientado acima apresentada.

Proposição 4.3.14 Seja T uma transformação contínua num reticulado completo $\langle S, \preceq \rangle$. Então T é monótona.

Prova. Suponha-se que T é contínua e sejam $x, y \in S$ com $x \preceq y$. O conjunto $X = \{x, y\}$ é orientado, já que y é majorante de qualquer subconjunto de X, logo $T(\sup(X)) = \sup(T(X))$. Mas por um lado $\sup(X) = y$, enquanto $T(X) = \{T(x), T(y)\}$, pelo que a igualdade anterior pode ser reescrita como $T(y) = \sup\{T(x), T(y)\}$, e em particular $T(x) \preceq T(y)$. □

É importante salientar que a implicação recíproca não é verdadeira: no conjunto $\omega + 1$, que é um reticulado completo para a relação de ordem usual, a transformação T tal que $T(n) = 0$ para $n \in \omega$ e $T(\omega) = \omega$ é uma transformação monótona que não é contínua. De facto, ω é um subconjunto orientado em $\omega + 1$ (já que qualquer conjunto finito de naturais tem um máximo), mas $\sup(T(\omega)) = \sup\{0\} = 0$ enquanto que $T(\sup(\omega)) = T(\omega) = \omega$.

Tem-se no entanto a seguinte relação.

Proposição 4.3.15 Seja T uma transformação monótona num reticulado completo $\langle S, \preceq \rangle$. Para cada conjunto $X \subseteq S$, tem-se a relação $\sup(T(X)) \preceq T(\sup(X))$.

Prova. Seja $X \subseteq S$ e tome-se $x \in X$. Então $T(x) \preceq T(\sup(X))$ por monotonia de T e definição de supremo. Por arbitrariedade de X conclui-se que $T(\sup(X))$ é majorante de $T(X)$, donde se

conclui a tese. □

O interesse das transformações contínuas vem do seguinte resultado.

Teorema 4.3.16 *(Kleene.)* Seja T uma transformação contínua num reticulado completo $\langle S, \preceq \rangle$. Então $\mathrm{pfmin}(T) = T \uparrow \omega$.

Prova. Uma vez que T é monótona, a existência de um ponto fixo mínimo de T é consequência do Teorema de Tarski (4.3.4). Pela mesma razão, a desigualdade $T \uparrow \omega \preceq \mathrm{pfmin}(T)$ sai da Proposição 4.3.9. Resta mostrar que $\mathrm{pfmin}(T) \preceq T \uparrow \omega$.

Observe-se em primeiro lugar que $\{T \uparrow n \mid n \in \omega\}$ é um conjunto orientado pela mesma razão que ω é orientado: um majorante de $T \uparrow n_1, \ldots, T \uparrow n_k$ é $T \uparrow (\max\{n_1, \ldots, n_k\})$, que claramente é um elemento daquele conjunto.

Tem-se então:

$$\begin{array}{rcll} T(T \uparrow \omega) & = & T(\sup\{T \uparrow n \mid n \in \omega\} & \text{definição de } T \uparrow \\ & = & \sup(T\{T \uparrow n \mid n \in \omega\}) & \text{continuidade} \\ & = & \sup\{T \uparrow (n+1) \mid n \in \omega\} & \text{definição de } T \uparrow \\ & = & \sup\{T \uparrow n \mid n \in \omega\} & \text{definição de } \bot = T \uparrow 0 \\ & = & T \uparrow \omega & \text{definição de } T \uparrow \end{array}$$

donde $T \uparrow \omega$ é ponto fixo de T; por definição de ponto fixo mínimo conclui-se que $\mathrm{pfmin}(T) \preceq T \uparrow \omega$, e a aplicação da anti-simetria de $\preceq$ permite concluir a prova. □

É importante observar que o Teorema de Kleene apenas se aplica a pontos fixos mínimos. Há uma razão de ser para esta assimetria: a noção de continuidade apenas diz respeito à preservação de supremos de conjuntos orientados, que são precisamente aqueles que são usados para definir as potências ascendentes de uma transformação. Para obter um resultado semelhante para pontos fixos máximos e potências descendentes, seria necessário conseguir relacionar ínfimos e supremos (em particular $T(\sup(X)) = \inf(T(X))$), o que não é em geral consequência da continuidade. Mais adiante apresentar-se-ão exemplos concretos de transformações contínuas cujo ponto fixo máximo requer $\omega 2$ ou mais iterações.

Capítulo 5

Semântica Declarativa da Programação em Lógica

Neste capítulo retoma-se a ideia expressa a seguir ao Corolário 1.4.6: para determinar se um conjunto de cláusulas é ou não contraditório, basta decidir se existe ou não um modelo de Herbrand para esse conjunto.

Para definir modelos de Herbrand de um programa determinado P recorre-se intensivamente à Proposição 1.4.3, identificando estes modelos com subconjuntos de $\mathcal{B}_P$.

Estes resultados são depois usados para provar a completude da resolução-SLD. Como foi já discutido, não se pode esperar ter um dual do Teorema de Clark no caso geral, já que existem respostas correctas que não são respostas calculadas. No entanto, mostrar-se-á que para toda a resposta correcta existe uma resposta mais geral que é calculada.

5.1 Modelo de Herbrand mínimo

Em primeiro lugar, prova-se um lema que será necessário várias vezes no seguimento.

Lema 5.1.1 Sejam Σ uma assinatura de primeira ordem, H uma estrutura de Herbrand sobre Σ, t um termo e ρ uma atribuição de valores em H. Então $[\![t\rho]\!]_H = [\![t]\!]_{H,\rho}$.

Prova. Por indução na estrutura do termo. Para constantes o resultado é imediato; se x for uma variável então $[\![x\rho]\!]_H = [\![\rho(x)]\!]_H = [\![x]\!]_{H,\rho}$, tendo em conta que H é uma estrutura de Herbrand. Para aplicações de símbolos de função a termos o resultado sai imediatamente por aplicação da hipótese de indução. □

O Corolário 1.4.6 pode ser tornado ainda mais forte; de facto, existe um modelo de Herbrand "fundamental".

Proposição 5.1.2 Sejam P um programa determinado e $\{H_i\}_{i\in I}$ um conjunto não vazio de modelos de Herbrand de P. Então $H = \bigcap_{i\in I}\{H_i\}$ é um modelo de Herbrand de P.

Prova. É imediato que H é uma estrutura de Herbrand para P. Tome-se então uma cláusula $C \equiv A \longleftarrow B_1, \ldots, B_q \in P$.

Suponha-se que H não é modelo de C. Então $[\![C]\!]_H = 0$, donde existe uma atribuição ρ tal que $[\![A \vee \neg B_1 \vee \ldots \vee \neg B_q]\!]_{H,\rho} = 0$; então $[\![A]\!]_{H,\rho} = 0$ e $[\![B_j]\!]_{H,\rho} = 1$ para $j = 1, \ldots, q$.

Pelo Lema 5.1.1, é imediato que $[\![p(t_1, \ldots, t_n)]\!]_{H,\rho} = [\![p(t_1\rho, \ldots, t_n\rho)]\!]_H$ para qualquer símbolo de predicado p. Sejam $A \equiv p_A(t_{A_1}, \ldots, t_{A_{n_A}})$ e $B_j \equiv p_j(t_{j_1}, \ldots, t_{j_{n_j}})$; então $p_A(t_{A_1}\rho, \ldots, t_{A_{n_A}}\rho) \notin H$, enquanto que $p_j(t_{j_1}\rho, \ldots, t_{j_{n_j}}\rho) \in H$. Por definição de H como intersecção de modelos, existe $i \in I$ tal que $p_A(t_{A_1}\rho, \ldots, t_{A_{n_A}}\rho) \notin H_i$, tendo-se por outro lado $p_j(t_{j_1}\rho, \ldots, t_{j_{n_j}}\rho) \in H_i$. Uma vez que ρ também é uma atribuição sobre H_i, tem-se outra vez pelo lema que $[\![A]\!]_{H_i,\rho} = 0$ e $[\![B_j]\!]_{H_i,\rho} = 1$, donde H_i não é modelo de C. □

O seguinte exemplo mostra que o resultado análogo para uniões não é válido.

Exemplo 5.1.3 Seja P o programa determinado $\{q(a) \longleftarrow p(a), p(b)\}$. Então $H_1 = \{p(a)\}$ e $H_2 = \{p(b)\}$ são modelos de Herbrand de P, mas $H_1 \cup H_2 = \{p(a), p(b)\}$ é uma estrutura de Herbrand de P que não é modelo daquele conjunto.

Proposição 5.1.4 Seja P um programa determinado. Então P tem modelo de Herbrand.

Prova. Tome-se $H = \mathcal{B}_P$. Então H é modelo de P: para qualquer cláusula $A \longleftarrow B_1, \ldots, B_q \in P$, tem-se que $[\![A]\!]_{H,\rho} = 1$ uma vez que $A\rho \in H$ e portanto H é modelo de $A \longleftarrow B_1, \ldots, B_q$. □

Proposição 5.1.5 Seja P um programa determinado. O conjunto $\wp\mathcal{B}_P$ das estruturas de Herbrand para P com a relação de inclusão é um reticulado completo.

Prova. Este exemplo é um caso particular do reticulado das partes de um conjunto (Exemplo 4.1.18). □

Definição 5.1.6 Seja P um programa determinado. O reticulado $\langle \wp\mathcal{B}_P, \subseteq \rangle$ diz-se o *reticulado de Herbrand* de P.

Definição 5.1.7 O *modelo de Herbrand mínimo* de um programa determinado P, denotado por $\mathcal{M}_P$, é a intersecção de todos os modelos de Herbrand de P.

Observe-se que $\mathcal{M}_P$ é modelo de P atendendo à Proposição 5.1.2.

Teorema 5.1.8 *(van Emden & Kowalsky I.)* Seja P um programa determinado. Então

$$\mathcal{M}_P = \{A \in \mathcal{B}_P \mid P \models A\}.$$

Prova. Seja $A \in \mathcal{B}_P$. Então:

$P \models A$	sse	$P \cup \{\neg A\}$ é contraditório	pela Proposição 1.3.13
	sse	$P \cup \{\neg A\}$ não tem modelos de Herbrand	pela Proposição 1.4.5
	sse	$[\![C]\!]_H = 0$ para algum $C \in P$ ou $[\![\neg A]\!]_H = 0$ para qualquer $H \subseteq \mathcal{B}_P$	
	sse	$[\![A]\!]_H = 1$ se H for modelo de Herbrand de P	definição de modelo
	sse	$A \in H$ qualquer H modelo de Herbrand de P	defn. estrutura de Herbrand
	sse	$A \in \bigcap\{H \mid H \subseteq \mathcal{B}_P$ é modelo de $P\}$	
	sse	$A \in \mathcal{M}_P$	definição de $\mathcal{M}_P$.

□

5.2 Transformação de Herbrand

Os resultados da secção anterior permitem restringir o problema da consequência semântica ao estudo do modelo de Herbrand mínimo. Porém, nenhuma das caracterizações apresentadas deste modelo é útil computacionalmente: a definição exige determinar *todos* os modelos de Herbrand dum dado programa (e em particular aquele em que se está interessado), enquanto que o Teorema de van Emden e Kowalsly requer que se saiba determinar se $P \models A$ para um programa P e um átomo A arbitrários – que é precisamente o problema que motivou o estudo do modelo de Herbrand mínimo em primeiro lugar. Nesta secção apresenta-se uma caracterização alternativa deste modelo enquanto ponto fixo duma transformação contínua. A utilidade desta caracterização será patente no seguimento.

Definição 5.2.1 Seja P um programa determinado. A *transformação de Herbrand* de P é a transformação $\mathcal{T}_P$ definida sobre $\wp\mathcal{B}_P$ como se segue.

$$\mathcal{T}_P(H) = \left\{ A \in \mathcal{B}_P \;\middle|\; \begin{array}{l} A \longleftarrow B_1, \ldots, B_q \text{ é instância fechada duma cláusula de } P \\ \{B_1, \ldots, B_q\} \subseteq H \end{array} \right\}$$

A transformação de Herbrand fornece um critério para decidir se uma estrutura de Herbrand é ou não modelo de um programa determinado.

Proposição 5.2.2 Sejam P um programa determinado e H uma estrutura de Herbrand para P. Então H é modelo de P sse $\mathcal{T}_P(H) \subseteq H$.

Prova.

($\longrightarrow$) Seja H um modelo de Herbrand de P e tome-se $A \in \mathcal{T}_P(H)$. Por definição de $\mathcal{T}_P$, existem uma cláusula $C \equiv A' \longleftarrow B'_1, \ldots, B'_q$ de P e uma substituição ρ tais que $A'\rho = A$ e $\{B'_1\rho, \ldots, B'_q\rho\} \subseteq H$. Então $[\![B'_i\rho]\!]_H = 1$, donde pelo Lema 5.1.1 se tem $[\![B'_i]\!]_{H,\rho} = 1$. Como H é modelo de P (e em particular de C), conclui-se que $[\![A']\!]_{H,\rho} = 1$, donde pelo mesmo lema $[\![A'\rho]\!]_H = 1$ e portanto $A = A'\rho \in H$. Logo $\mathcal{T}_P(H) \subseteq H$.

($\longleftarrow$) Suponha-se agora que $\mathcal{T}_P(H) \subseteq H$ e seja $C \equiv A \longleftarrow B_1, \ldots, B_q$ uma cláusula de H. Seja ρ uma atribuição; se $[\![B_i]\!]_{H,\rho} = 0$ para algum índice i, então trivialmente H é modelo de C. Suponha-se então que $[\![B_i]\!]_{H,\rho} = 1$ para $i = 1, \ldots, q$; pelo Lema 5.1.1 tira-se $[\![B_i\rho]\!]_H = 1$, donde $B_i\rho \in H$. Ora $A\rho$ é fechada, donde $C\rho$ é uma instância fechada duma cláusula de P com $\{B_1\rho, \ldots, B_q\rho\} \subseteq H$, donde $A\rho \in \mathcal{T}_P(H)$. Da hipótese conclui-se que $A\rho \in H$, donde segue que $[\![A\rho]\!]_H = 1$ e aplicando o lema conclui-se que $[\![A]\!]_{H,\rho} = 1$. Logo H é modelo de C. Por arbitrariedade de C conclui-se que H é modelo de P.

□

Corolário 5.2.3 Seja P um programa determinado. Então todos os pontos fixos de $\mathcal{T}_P$ são modelos de P.

Prova. Por definição de ponto fixo e reflexividade de $\subseteq$. □

Proposição 5.2.4 Seja P um programa determinado. A transformação de Herbrand de P é contínua.

Prova. Seja $\{H_i\}_{i\in I}$ um conjunto orientado (em $\wp\mathcal{B}_P$) de estruturas de Herbrand de P. É necessário provar que $\mathcal{T}_P(\sup\{H_i\}_{i\in I}) = \sup\{\mathcal{T}_P(H_i)\}_{i\in I}$. Tendo em conta que os supremos neste reticulado são dados por uniões, a igualdade anterior pode-se reescrever como $\mathcal{T}_P\left(\bigcup_{i\in I} H_i\right) = \bigcup_{i\in I}\mathcal{T}_P(H_i)$.

Seja $A \in \mathcal{B}_P$. Se $A \in \mathcal{T}_P\left(\bigcup_{i\in I} H_i\right)$ então existem uma cláusula $C' \equiv A' \longleftarrow B'_1, \ldots, B'_q$ de P e uma substituição ρ' tais que $A'\rho' = A$ e $\{B'_1\rho', \ldots, B'_q\rho'\} \subseteq \bigcup_{i\in I} H_i$. Então existem modelos $H_{i_1}, \ldots, H_{i_q}$ (não necessariamente distintos) tais que $B'_j\rho' \in H_{i_j}$. Como $\{H_i\}_{i\in I}$ é orientado, isto implica que exista $i \in I$ tal que $H_{i_j} \subseteq H_i$ e portanto $B'_j\rho' \in H_i$ para $j = 1, \ldots, q$, donde $A'\rho' = A \in \mathcal{T}_P H_i$ e portanto $A \in \bigcup_{i\in I}\mathcal{T}_P(H_i)$.

Reciprocamente, se $A \in \bigcup_{i\in I}\mathcal{T}_P(H_i)$ então $A \in \mathcal{T}_P(H_i)$ para algum $i \in I$, pelo que existem uma cláusula $C'' \equiv A'' \longleftarrow B''_1, \ldots, B''_q$ de P e uma substituição ρ'' tais que $A''\rho'' = A$ e $\{B''_1\rho'', \ldots, B''_q\rho''\} \subseteq H_i$. Então $\{B''_1\rho'', \ldots, B''_q\rho''\} \subseteq \bigcup_{i\in I} H_i$, donde $A''\rho'' = A \in \mathcal{T}_P\left(\bigcup_{i\in I} H_i\right)$. □

Este resultado permite definir potências ascendentes e descendentes de $\mathcal{T}_P$ para qualquer programa determinado P.

Proposição 5.2.5 Seja P um programa determinado. Então $\mathcal{T}_P \downarrow \alpha$ é modelo de P para qualquer ordinal α.

Prova. Como $\alpha \leq \alpha + 1$, tem-se $\mathcal{T}_P(\mathcal{T}_P \downarrow \alpha) = \mathcal{T}_P \downarrow (\alpha + 1) \subseteq \mathcal{T}_P \downarrow \alpha$ pela Proposição 4.3.8, donde $\mathcal{T}_P \downarrow \alpha$ é modelo de Herbrand de P pela Proposição 5.2.2. □

Antes de prosseguir, apresentam-se alguns exemplos de programas e das transformações de Herbrand que lhes estão associadas, interpretados à luz destes resultados.

Exemplo 5.2.6 Para um programa determinado P arbitrário, $\mathcal{T}_P(\emptyset)$ é o conjunto das instâncias fechadas de factos de P.

Exemplo 5.2.7 Seja P o seguinte programa determinado.

$$p(f(x)) \longleftarrow p(x)$$
$$q(a) \longleftarrow p(x)$$

Então $\mathcal{T}_P(J) = \{p(f(t)) \mid p(t) \in J\} \cup \{q(a) \mid p(t) \in J\}$.

- $\mathcal{T}_P(\emptyset) = \emptyset$, donde pelo Teorema de Kleene $\text{pfmin}(\mathcal{T}_P) = \emptyset$; pelo Corolário 5.2.3, $\emptyset$ é modelo de Herbrand de P.
- $\mathcal{T}_P \downarrow 1 = \mathcal{T}_P(\mathcal{B}_P) = \{q(a)\} \cup \{p(f(t)) \mid t \in \mathcal{U}_P\}$
- $\mathcal{T}_P \downarrow 2 = \{q(a)\} \cup \{p(f(f(t))) \mid t \in \mathcal{U}_P\}$
- $\mathcal{T}_P \downarrow n = \{q(a)\} \cup \{p(f^k(t)) \mid t \in \mathcal{U}_P, k \geq n\}$
- $\mathcal{T}_P \downarrow \omega = \{q(a)\}$
- $\mathcal{T}_P \downarrow (\omega + 1) = \emptyset \neq \mathcal{T}_P \downarrow \omega$
- $\text{pfmax}(\mathcal{T}_P) = \emptyset$.

Exemplo 5.2.8 Seja P o seguinte programa determinado.

$$p(f(x)) \longleftarrow p(x)$$
$$p(a) \longleftarrow$$

Então $\mathcal{T}_P(J) = \{p(f(t)) \mid p(t) \in J\} \cup \{p(a)\}$.

- $\mathcal{T}_P(\emptyset) = \{p(a)\}$
- $\mathcal{T}_P \uparrow 2 = \{p(a), p(f(a))\}$
- $\mathcal{T}_P \uparrow n = \{p(f^k(a)) \mid k < n\}$
- $\mathcal{T}_P \uparrow \omega = \{p(f^k(a)) \mid k \in \mathbb{N}\} = \mathcal{B}_P$; mais uma vez, pelo Teorema de Kleene este conjunto coincide com pfmin($\mathcal{T}_P$) e pelo Corolário 5.2.3 é um modelo de Herbrand de P.
- $\mathcal{T}_P \downarrow 1 = \mathcal{T}_P(\mathcal{B}_P) = \mathcal{B}_P$
- pfmax($\mathcal{T}_P$) $= \mathcal{B}_P$ é modelo de Herbrand de P.

Estes exemplos sugerem o seguinte resultado.

Teorema 5.2.9 *(van Emden & Kowalsky II.)* Seja P um programa determinado. Então $\mathcal{M}_P =$ pfmin($\mathcal{T}_P$).

Prova.

$$\begin{array}{rcll} \mathcal{M}_P & = & \bigcap_{i\in I}\{H \subseteq \mathcal{B}_P \mid \ H \text{ é modelo de } P\} & \text{definição de } \mathcal{M}_P \\ & = & \inf_{i\in I}\{H \subseteq \mathcal{B}_P \mid \ H \text{ é modelo de } P\} & \text{definição de ínfimo em } \wp\mathcal{B}_P \\ & = & \inf_{i\in I}\{H \subseteq \mathcal{B}_P \mid \mathcal{T}_P(H) \subseteq H\} & \text{pela Proposição 5.2.2} \\ & = & \text{pfmin}(\mathcal{T}_P) & \text{pelo Teorema de Tarski} \\ & = & \mathcal{T}_P \uparrow \omega & \text{pelo Teorema de Kleene} \end{array}$$

□

Por outro lado, impondo restrições à classe de programas é possível estabelecer um dual do Teorema de Kleene.

Proposição 5.2.10 Seja P um programa determinado cujas cláusulas não contêm nenhuma ocorrência de símbolos de função. Então pfmax$\mathcal{T}_P = \mathcal{T}_P \downarrow \omega$.

Prova. Se as cláusulas de P não contiverem nenhum símbolo de função, então $\mathcal{U}_P$ é forçosamente finito, uma vez que P é um conjunto finito de cláusulas finitas. Então $\mathcal{B}_P$ é finito, pelo que só há um número finito de estruturas de Herbrand para P. Atendendo à prova da Proposição 4.3.10, conclui-se que pfmax($\mathcal{T}_P$) $= \mathcal{T}_P \downarrow \alpha$ com $\alpha \leq \omega$. Em particular, $\mathcal{T}_P \downarrow \omega =$ pfmax$\mathcal{T}_P$. □

Conclui-se este capítulo com uma caracterização de respostas correctas em termos do modelo de Herbrand mínimo.

Proposição 5.2.11 Sejam P um programa determinado, $G \equiv \longleftarrow B_1, \ldots, B_q$ um objectivo determinado e θ uma resposta de P a G tal que $(B_1 \wedge \ldots \wedge B_q)\theta$ é fechada. Então as seguintes asserções são equivalentes.

(i) A resposta θ é uma resposta correcta de P a G.

(ii) Todo o modelo de Herbrand de P é modelo de $(B_1 \wedge \ldots \wedge B_q)\theta$.

(iii) $\mathcal{M}_P$ é modelo de $(B_1 \wedge \ldots \wedge B_q)\theta$.

Prova.

(i)$\longrightarrow$(ii) Por definição de resposta correcta.

(ii)$\longrightarrow$(iii) Consequência de $\mathcal{M}_P$ ser um modelo de Herbrand de P.

(iii)$\longrightarrow$(i) Suponha-se que $\mathcal{M}_P$ é modelo de $(B_1 \wedge ... \wedge B_q)\theta$, sendo esta fórmula fechada. Então $\mathcal{M}_P$ é modelo de $B_i\theta$ para $i = 1, ..., q$, o que é equivalente (como estas fórmulas são átomos fechados) a $B_i\theta \in \mathcal{M}_P$. Por definição de $\mathcal{M}_P$, também se tem $B_i\theta \in H$ para qualquer modelo de Herbrand H de P, donde H é modelo de $(B_1 \wedge ... \wedge B_q)\theta$. Então $P \cup \{\neg(B_1 \wedge ... \wedge B_q)\theta\}$ não tem modelos de Herbrand, donde pela Proposição 1.4.5 $P \models (B_1 \wedge ... \wedge B_q)\theta$. Como esta fórmula é fechada, isto equivale a $P \models \forall((B_1 \wedge ... \wedge B_q)\theta)$, donde θ é resposta correcta de P a G.

□

Observe-se que a exigência de $G\theta$ ser fechada é crucial.

Exemplo 5.2.12 Sejam $P = \{p(a) \longleftarrow$ e $G = \longleftarrow p(x)$. Então $\forall(p(x)\varepsilon)$ é satisfeita por $\mathcal{M}_P$, mas $P \not\models \forall(p(x)\varepsilon)$.

5.3 Completude da resolução-SLD

Nesta secção prova-se finalmente o dual (dentro do possível) do Teorema de Clark.

Em primeiro lugar, apresentam-se dois resultados relacionando refutações-SLD com o modelo de Herbrand mínimo.

Proposição 5.3.1 O conjunto de sucessos de um programa determinado P está contido no seu modelo de Herbrand mínimo $\mathcal{M}_P$.

Prova. Seja $A \in \mathcal{B}_P$ um elemento do conjunto de sucessos de P. Então existe uma refutação-SLD de P e A com resposta calculada ε (uma vez que A é fechado). Pelo Teorema de Clark, ε é resposta correcta de P a A, donde $A \models \forall(A\varepsilon)$. Como A é fechada, $\forall(A\varepsilon) \equiv A$, logo $A \in \mathcal{M}_P$ pelo Teorema de van Emden & Kowalsky I. □

Proposição 5.3.2 Sejam P um programa determinado, $G \equiv \longleftarrow A_1, ..., A_p$ um objectivo determinado e $\mathcal{S}$ uma função de selecção. Se existe uma refutação-SLD de P e G via $\mathcal{S}$ de comprimento n com sequência de substituições $\Theta = \{\theta_1, ..., \theta_n\}$, então $\bigcup_{i=1}^{p}[A_i\theta_1 ... \theta_n] \subseteq \mathcal{T}_P \uparrow n$.

Prova. Por indução no comprimento da refutação-SLD. Suponha-se que esta tem comprimento 1; então $p = 1$ e existe um facto $A \longleftarrow$ em P tal que $A_1\theta_1 = A\theta_1$. Ora $\mathcal{T}_P \uparrow 1$ contém precisamente as instâncias fechadas de factos de P, em particular as instâncias fechadas de A. Uma vez que toda a instância fechada de $A_1\theta_1 = A\theta$ é uma instância fechada de A, conclui-se que $[A_1\theta_1] \subseteq \mathcal{T}_P \uparrow 1$.

Suponha-se agora que o resultado é válido para refutações de comprimento $n-1$. Ignorando o primeiro passo da derivação, obtém-se uma refutação-SLD de P a G_1 com sequência de substituições $\{\theta_2, ..., \theta_n\}$, à qual a hipótese de indução é aplicável. Conclui-se portanto, sendo $G_1 \equiv \longleftarrow A'_1, ..., A'_q$, que $[A'_j\theta_2 ... \theta_n] \subseteq \mathcal{T}_P \uparrow (n-1)$ para $j = 1, ..., q$.

Seja A_j um sub-objectivo de G_0.

- Se $A_j \neq \mathcal{S}(G_0)$, então $A_j\theta_1 = A'_k$ para algum k e, como toda a instância fechada de $A_j\theta_1 ... \theta_n$ é instância fechada de $A_j\theta_1$, conclui-se que $[A_j\theta_1 ... \theta_n] = [A'_k\theta_2 ... \theta_n] \subseteq \mathcal{T}_P \uparrow (n-1) \subseteq \mathcal{T}_P \uparrow n$.

- Se $A_j = \mathcal{S}(G_0)$ e C_1 é um facto, então conclui-se como atrás que $[A_j\theta_1 \dots \theta_n] \subseteq [A_j\theta_1] \subseteq \mathcal{T}_P \uparrow 1$.
- Se $A_j = \mathcal{S}(G_0)$ e $C_1 \equiv A \longleftarrow B_1, \dots, B_r$, então $\{B_1, \dots, B_r\} \subseteq \{A'_1, \dots, A'_q\}$, donde toda a instância fechada de $B_i\theta_2 \dots \theta_n$ está em $\mathcal{T}_P \uparrow (n-1)$. Seja A^* uma instância fechada de $A_j\theta_1 \dots \theta_n$. Então A^* é uma instância fechada de $A_j\theta_1 = A\theta_1$; seja $A^* = A\theta_1\sigma$, com $\sigma = \theta_2 \dots \theta_n\sigma'$. Conclui-se que $A\theta_1\sigma \longleftarrow B_1\theta_1\sigma, \dots, B_r\theta_1\sigma$ é uma instância fechada duma cláusula de P (nomeadamente, de C_1) cujo antecedente está contido em $\mathcal{T}_P \uparrow (n-1)$ (uma vez que $B_i\theta_1\sigma = B_i\theta_1\theta_2 \dots \theta_n\sigma'$ é instância fechada de $B_i\theta_1\theta_2 \dots \theta_n$), donde por definição de $\mathcal{T}_P$ se conclui que $A^* \in \mathcal{T}_P \uparrow n$.

□

Os duais destes resultados não são imediatos devido mais uma vez ao facto de todas as substituições $\theta_1, \dots, \theta_n$ terem de ser UMGs. Assim, interessa estudar as consequências de levantar a restrição de escolher sempre UMGs a cada passo duma derivação-SLD.

Definição 5.3.3 Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção. Uma *derivação-SLD generalizada* de P e G é um triplo $\langle \mathcal{G}, \mathcal{C}, \Theta \rangle$ em que $\mathcal{G}$ e $\mathcal{C}$ satisfazem as propriedades de uma derivação-SLD e $\Theta = \{\theta_1, \dots, \theta_n, \dots\}$ é tal que θ_i é uma unificadora de $\mathcal{S}(G_i)$ e da cabeça de C_i.

Uma derivação-SLD generalizada de P e G de comprimento finito n tal que $G_n \equiv \square$ diz-se uma *refutação-SLD generalizada* de P a G.

A generalização que se obtém com esta definição pode ser descrita de forma precisa pelo seguinte resultado.

Lema 5.3.4 *(Lema da UMG.)* Sejam P um programa determinado e G um objectivo determinado. Para cada refutação-SLD generalizada $\langle \mathcal{G}, \mathcal{C}, \Theta \rangle$ de P e G, com $\Theta = \{\theta_1, \dots, \theta_n\}$, existe uma refutação-SLD $\langle \mathcal{G}', \mathcal{C}', \Theta' \rangle$ de P e G com o mesmo comprimento e tal que, sendo $\Theta' = \{\theta'_1, \dots, \theta'_n\}$, se tem $\theta_1 \dots \theta_n = \theta'_1 \dots \theta'_n\gamma$ para alguma substituição γ.

Prova. Por indução no comprimento n da refutação-SLD generalizada.

Se $n = 1$, pode-se tomar $\mathcal{C}' = \mathcal{C}$ e $\mathcal{G}' = \mathcal{G}$; como θ_1 é uma unificadora da cabeça de C_1 e do único átomo de G_0 e θ'_1 é uma UMG destes dois átomos, a existência de γ é consequência da definição de UMG.

Suponha-se agora que o resultado é válido para refutações-SLD generalizadas de comprimento $n-1$. Tome-se $C'_1 = C_1$ e seja θ'_1 uma UMG do átomo seleccionado na refutação-SLD generalizada em G_0 com a cabeça de C_1. Por definição de UMG, existe uma substituição δ tal que $\theta_1 = \theta'_1\delta$.

Mais, por definição de resolvente tem-se ainda a relação $G_1 = G'_1\delta$; então existe uma refutação-SLD generalizada de P e G'_1 obtida a partir de $\langle \mathcal{G}, \mathcal{C}, \Theta \rangle$ eliminando o primeiro passo e substituindo θ_2 por $\delta\theta_2$. Esta refutação-SLD generalizada tem comprimento $n-1$, donde por hipótese de indução existe uma refutação-SLD de P e G'_1 com comprimento $n-1$ e sequência de substituições $\theta'_2, \dots, \theta'_n$ tal que $\delta\theta_2 \dots \theta_n = \theta'_2 \dots \theta'_n\gamma$ para alguma substituição γ, o que permite completar a refutação-SLD de P e G_0. Finalmente, resta observar que $\theta_1\theta_2 \dots \theta_n = \theta'_1\delta\theta_2 \dots \theta_n = \theta'_1\theta'_2 \dots \theta'_n\gamma$, o que conclui a prova. □

A utilidade deste lema é imediata: doravante, para provar a existência de refutações-SLD basta construir uma refutação-SLD generalizada (que muitas vezes é mais simples) e aplicar este resultado.

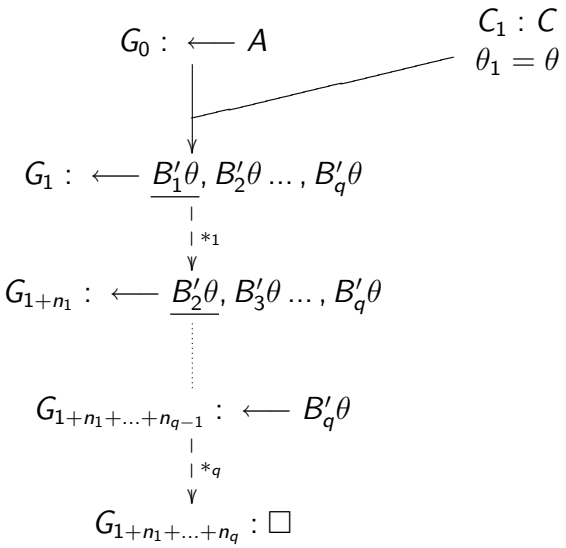
Pode-se assim obter um recíproco da Proposição 5.3.2.

Proposição 5.3.5 Sejam P um programa determinado e A um objectivo determinado. Se $A \in \mathcal{T}_P \uparrow n$, então existe uma refutação-SLD de P e $\longleftarrow A$.

Prova. Por indução em n. Se $n = 0$, não há nada a provar, já que $\mathcal{T}_P \uparrow 0 = \emptyset$.

Suponha-se que existe refutação-SLD de P a $\longleftarrow B$ para qualquer $B \in \mathcal{T}_P \uparrow (n-1)$. Tome-se $A \in \mathcal{T}_P \uparrow n$; por definição de $\mathcal{T}_P$, existem uma cláusula $C \equiv A' \longleftarrow B'_1, \ldots, B'_q$ e uma substituição θ tais que $A'\theta = A$ e $B'_i\theta \in \mathcal{T}_P \uparrow (n-1)$ para $i = 1, \ldots, q$.

Por hipótese de indução existem refutações-SLD de P e $\longleftarrow B'_i\theta$. Pode-se então contruir uma refutação-SLD generalizada de P a $\longleftarrow A$ da seguinte forma.



Cada $*_i$ corresponde à refutação-SLD obtida a partir da refutação-SLD de P a $\longleftarrow B'_i\theta$ acrescentando $B'_{i+1}\theta, \ldots, B'_q\theta$ a todos os sub-objectivos. Note-se que, como $B'_i\theta$ é fechada, não é afectada por substituições.

Pelo Lema da UMG, existe uma refutação-SLD de P a $\longleftarrow A$. □

Observe-se que o comprimento da refutação-SLD de P e $\longleftarrow A$ é, em geral, maior que n, contrariamente ao que sucedia na Proposição 5.3.2.

Este resultado permite generalizar imediatamente a Proposição 5.3.1.

Proposição 5.3.6 O conjunto de sucessos de um programa determinado P coincide com o seu modelo de Herbrand mínimo $\mathcal{M}_P$.

Prova.

- $\subseteq$ Pela Proposição 5.3.1.
- $\supseteq$ Suponha-se que $A \in \mathcal{M}_P$. Então $A \in \mathcal{T}_P \uparrow n$ para algum $n \in \mathbb{N}$. Pela Proposição 5.3.5, existe uma refutação-SLD para P e $\longleftarrow A$, donde A está no conjunto de sucessos de P.

□

Proposição 5.3.7 Seja $\mathcal{S}$ uma função de selecção. O conjunto de sucessos de um programa determinado P via $\mathcal{S}$ coincide com $\mathcal{M}_P$.

Prova. Por definição, o conjunto de sucessos de P via $\mathcal{S}$ está contido no conjunto de sucessos de P, que pela proposição anterior coincide com $\mathcal{M}_P$. Reciprocamente, para cada refutação-SLD de P e $\longleftarrow A$ com resposta calculada θ o Teorema 3.2.3 garante a existência de uma refutação-SLD de P e $\longleftarrow A$ com resposta calculada σ tal que $A\theta$ e $A\sigma$ são variantes. Em particular, se $A \in \mathcal{M}_P$, então A é fechada, existe refutação-SLD de P e $\longleftarrow A$, $\theta = \sigma = \varepsilon$ e A está no conjunto de sucessos de P via $\mathcal{S}$. □

O Lema da UMG permite ainda generalizar o objectivo a ser refutado em situações muito gerais.

Corolário 5.3.8 *(Lema da Elevação.)* Sejam P um programa determinado, G um objectivo determinado e σ uma substituição. Para cada refutação-SLD $\langle \mathcal{G}, \mathcal{C}, \Theta \rangle$ de P e $G\sigma$, com $\Theta = \{\theta_1, \ldots, \theta_n\}$, tal que $C_1\sigma = C_1$ existe uma refutação-SLD $\langle \mathcal{G}', \mathcal{C}', \Theta' \rangle$ de P e G com o mesmo comprimento e tal que, sendo $\Theta' = \{\theta'_1, \ldots, \theta'_n\}$, se tem $\sigma\theta_1 \ldots \theta_n = \theta'_1 \ldots \theta'_n\gamma$ para alguma substituição γ.

Prova. Sejam $C_1 \equiv A \longleftarrow B_1, \ldots, B_q$ e A' o átomo seleccionado em G_0. Nas condições da hipótese, uma vez que $C_1\sigma = C_1$, tem-se $A\sigma\theta_1 = A\theta_1 = G_0\sigma\theta_1$, donde é possível construir uma refutação-SLD generalizada de P a G com comprimento n substituindo $G\sigma$ por G e θ_1 por $\sigma\theta_1$ na refutação-SLD dada. Pelo Lema da UMG existe uma refutação-SLD de P e G nas condições do enunciado. □

Os resultados seguintes são os resultados mais fortes de completude que se conseguem obter.

Teorema 5.3.9 *(Teorema de Hill.)* Sejam P um programa determinado e G um objectivo determinado. Se $P \cup \{G\}$ é contraditório, então para toda a função de selecção $\mathcal{S}$ existe uma refutação-SLD de P e G via $\mathcal{S}$.

Prova. Seja $G \equiv \longleftarrow B_1, \ldots, B_q$ e suponha-se que $P \cup \{G\}$ é contraditório. Uma vez que $\mathcal{M}_P$ é modelo de P, $[\![G]\!]_{\mathcal{M}_P,\rho} = 0$ para alguma atribuição ρ, isto é, $[\![\neg B_1 \vee \ldots \vee \neg B_q]\!]_{\mathcal{M}_P,\rho} = 0$. Pelo Lema 5.1.1, $[\![\neg B_1\rho \vee \ldots \vee \neg B_q\rho]\!]_{\mathcal{M}_P} = 0$, o que é equivalente a $[\![\neg B_i\rho]\!]_{\mathcal{M}_P} = 0$ para $i = 1, \ldots, q$ ou ainda a $[\![B_i\rho]\!]_{\mathcal{M}_P} = 1$. Como $B_i\rho \in \mathcal{B}_P$, da definição de modelo de Herbrand conclui-se que $B_i\rho \in \mathcal{M}_P$. Pela Proposição 5.3.6 existe uma refutação-SLD de P e $\longleftarrow B_i\rho$; tal como na prova da Proposição 5.3.5, podem-se combinar estas refutações-SLD para $i = 1, \ldots, q$ numa única refutação-SLD de P e $\longleftarrow B_1\rho, \ldots, B_q\rho$. Aplicando o Lema da Elevação, obtém-se uma refutação-SLD de P e $\longleftarrow B_1, \ldots, B_q$. Pelo Teorema 3.2.3 existe uma refutação-SLD de P e $\longleftarrow B_1, \ldots, B_q$ via $\mathcal{S}$. □

Lema 5.3.10 *(Lema da Resposta Vazia.)* Sejam P um programa determinado e A um átomo. Se $P \models \forall A$, então existe uma refutação-SLD de P e $\longleftarrow A$ com resposta calculada ε.

Prova. Sejam $x_1, \ldots, x_k$ os símbolos de variável que ocorrem em A e escolham-se símbolos de constante $a_1, \ldots, a_k$ que não sejam usados em P. Sejam Σ a assinatura obtida acrescentando $a_1, \ldots, a_k$ à assinatura subjacente a P e $\theta = \{x_1/a_1, \ldots, x_k/a_k\}$.

Então $P \models A\theta$; uma vez que $A\theta$ é um átomo fechado, $A\theta \in \mathcal{M}_P$ pelo Teorema de van Emden e Kowalsky I, sendo este modelo construído sobre a assinatura Σ. Pela Proposição 5.3.6, existe uma refutação-SLD de P e $A\theta$; mais, podem-se escolher os variantes nessa refutação-SLD por forma a que neles não ocorra nenhuma das variáveis $x_1, \ldots, x_k$. Seja $\Theta = \{\theta_1, \ldots, \theta_n\}$ a sequência de UMGs nesta refutação-SLD; uma vez que $x_1, \ldots, x_k$ não ocorrem na derivação, nenhuma dessas variáveis é alterada por $\theta_1, \ldots, \theta_n$.

Substituindo uniformemente toda a ocorrência de cada constante a_i pela variável x_i, obtém-se uma refutação-SLD para P e A. Uma vez que $\theta_1, \ldots, \theta_n$ não actuavam sobre a_i (por ser uma constante), as substituições na derivação-SLD obtida não actuam sobre x_i. Logo a resposta calculada por esta refutação-SLD é ε. □

Teorema 5.3.11 *(Completude da resolução-SLD.)* Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção. Se θ é uma resposta correcta de P a G, então existe uma refutação-SLD de P a G via $\mathcal{S}$ com resposta calculada σ tal que $\theta = \sigma\gamma$ para alguma substituição γ.

Prova. Seja $G \equiv \longleftarrow A_1, \ldots, A_q$ e suponha-se que θ é resposta correcta de P a G. Então $P \models \forall((A_1 \wedge \ldots \wedge A_q)\theta)$, donde em particular $P \models \forall(A_i\theta)$ para $i = 1, \ldots, q$.

Pelo Lema da Resposta Vazia, existe uma refutação-SLD de P a $A_i\theta$ com resposta calculada ε, para $i = 1, \ldots, q$. Mais uma vez, podem-se combinar estas refutações-SLD numa só pelo mecanismo atrás utilizado, obtendo-se uma refutação-SLD para P e $G\theta$ com resposta calculada ε.

Seja $\Theta = \{\theta_1, \ldots, \theta_n\}$ a sequência de UMGs desta refutação-SLD. Pelo Lema da Elevação existe uma refutação-SLD de P a G com sequência de UMGs $\Theta' = \{\theta'_1, \ldots, \theta'_n\}$ tal que $\theta\theta_1 \ldots \theta_n = \theta'_1 \ldots \theta'_n\gamma$ para alguma substituição γ. Atendendo ao Teorema 3.2.3, pode assumir-se sem perda de generalidade que esta refutação-SLD é uma refutação-SLD via $\mathcal{S}$. Restringindo às variáveis que ocorrem em G, conclui-se que $\theta = \sigma\gamma$, sendo σ a restrição de $\theta'_1 \ldots \theta'_n$ a estas (ou seja, a resposta calculada de P a G), uma vez que para qualquer i a restrição de $\theta_1 \ldots \theta_n$ às variáveis de $A_i\theta$ é ε. □

5.4 Questões práticas: pesquisa e corte

Estes resultados têm consequências na prática.

Proposição 5.4.1 Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção. Se $P \cup \{G\}$ é contraditório, então a árvore-SLD de P e G via $\mathcal{S}$ tem pelo menos um ramo bem sucedido.

Prova. Nas condições do enunciado, o Teorema de Hill garante a existência de uma refutação-SLD de P e G via $\mathcal{S}$, que corresponde a um ramo bem sucedido na árvore-SLD de P e G via $\mathcal{S}$. □

Proposição 5.4.2 Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção. Para toda a resposta correcta θ de P a G existe um ramo bem sucedido na árvore-SLD de P e G via $\mathcal{S}$ com resposta calculada σ tal que $\theta = \sigma\gamma$ para alguma substituição γ.

Prova. Nas condições do enunciado, a completude da resolução-SLD garante a existência de uma refutação-SLD de P e G via $\mathcal{S}$ com resposta calculada σ tal que $\theta = \sigma\gamma$ para alguma substituição γ. De novo, esta refutação-SLD corresponde a um ramo bem sucedido na árvore-SLD de P e G via $\mathcal{S}$. □

O problema de encontrar uma resposta calculada pode então ser visto como um problema de pesquisa: encontrar uma resposta calculada de P a G é encontrar um ramo bem sucedido n(um)a árvore-SLD de P e G.

Definição 5.4.3 Uma *regra de pesquisa* é uma estratégia de procura de ramos bem sucedidos em árvores-SLD.

Um *procedimento de refutação* é especificado por uma função de selecção e uma regra de pesquisa.

O procedimento de refutação típico é então seleccionar o átomo mais à esquerda e unificá-lo com a primeira cláusula do programa que for unificável. Isto corresponde a efectuar uma procura em profundidade primeiro na árvore-SLD... o que tem consequências graves: uma procura em profundidade primeiro bloqueia em ramos infinitos, não voltando nunca atrás para encontrar respostas correctas.

Exemplo 5.4.4 Seja P o programa determinado seguinte.

$$\begin{array}{l} p(a,b) \longleftarrow \\ p(c,b) \longleftarrow \\ p(x,z) \longleftarrow p(x,y), p(y,z) \\ p(x,y) \longleftarrow p(y,x) \end{array}$$

Considere-se o objectivo $G \equiv \longleftarrow p(a,c)$. Embora exista uma refutação-SLD para este objectivo, nenhuma pesquisa em profundidade primeiro a vai encontrar (independentemente da função de selecção escolhida).

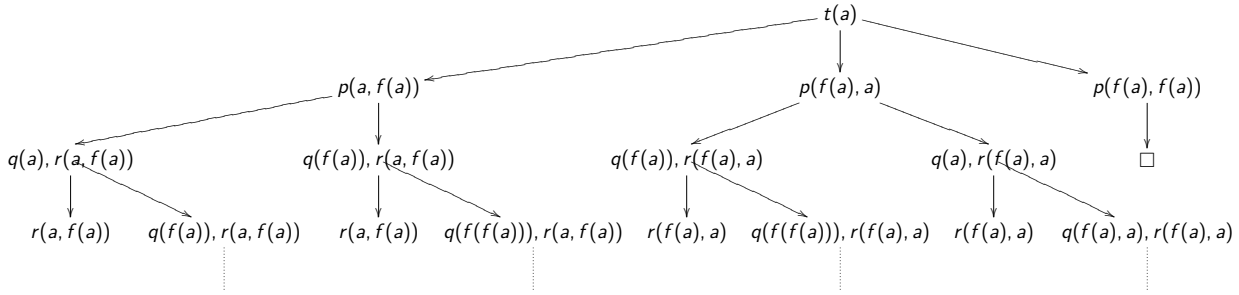
Para optimizar a pesquisa de ramos bem sucedidos na árvore-SLD, em particular evitando problemas causados pelos ramos infinitos, as implementações da resolução-SLD disponibilizam frequentemente um mecanismo de *corte*. O corte é (tipicamente) um símbolo extra-lógico que só pode ocorrer no antecedente duma cláusula; é interpretado logicamente como um átomo que é sempre verdadeiro, mas a sua verdadeira função é semântica: indica ao interpretador que a árvore-SLD a pesquisar deve ser podada.

Seja $A \longleftarrow A_1, \ldots, A_p, !, B_1, \ldots, B_q$ uma cláusula com uma ocorrência do corte (!). Suponha-se que no nó $\mathcal{N}_1$ da árvore-SLD em exploração se unificou um objectivo B com A. Se mais tarde se chegar a um nó $\mathcal{N}_2$ em que o átomo seleccionado é !, todos os descendentes ainda não explorados dos nós entre $\mathcal{N}_1$ e $\mathcal{N}_2$ (incluindo estes) são eliminados da árvore.

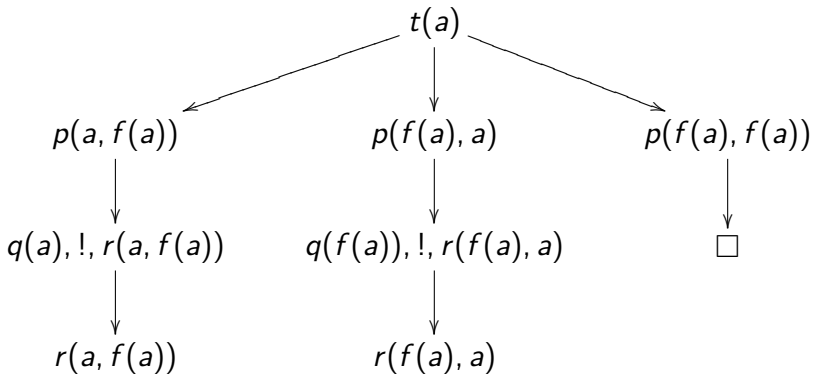
Exemplo 5.4.5 Seja P o programa determinado seguinte.

$$
\begin{array}{l}
p(f(a), f(a)) \longleftarrow \\
p(x, y) \longleftarrow q(x), !, r(x, y) \\
p(x, y) \longleftarrow q(y), r(x, y) \\
q(a) \longleftarrow \\
q(f(a)) \longleftarrow \\
q(x) \longleftarrow q(f(x)) \\
t(x) \longleftarrow p(x, f(x)) \\
t(x) \longleftarrow p(f(x), x) \\
t(x) \longleftarrow p(f(x), f(x))
\end{array}
$$

Considere-se o objectivo $G \equiv \longleftarrow t(a)$. A árvore-SLD de P e G (com a função de selecção usual), ignorando os cortes, tem o seguinte aspecto. Apenas se indicam os objectivos em cada nó; as reticências indicam ramos infinitos.



Tendo em conta os cortes, a árvore que é construída é apenas a seguinte.



Definição 5.4.6 Um corte diz-se *limpo* se nunca elimina respostas correctas de nenhuma árvore-SLD. Um corte que não é limpo diz-se *sujo*.

Os cortes do exemplo anterior eram todos limpos.

Capítulo 6

Negação

A resolução-SLD como paradigma de programação é bastante útil (até porque é completa à Turing, como se viu anteriormente), mas na prática há definições que recorrem naturalmente à negação. Por exemplo: um elemento duma ordem parcial é um mínimo se não houver nenhum elemento melhor que ele; isto poder-se-ia representar através da cláusula

$$min(x) \longleftarrow \neg r(x, y)$$

num contexto (programa) em que r correspondesse à relação de ordem.

Neste capítulo apresenta-se uma forma de tratar a negação no contexto da Programação em Lógica. Infelizmente, conforme se verá no seguimento, os resultados obtidos não são tão satisfatórios como para o fragmento clausal da Lógica de Primeira Ordem.

6.1 Negação por Falha

A motivação para a regra da Negação por Falha prende-se com a interpretação da Programação em Lógica no contexto das Bases de Dados. Um programa determinado generaliza a noção de base de dados relacional: os factos fechados correspondem aos factos da base de dados, especificando as relações que a constituem. Porém, a possibilidade de considerar factos não fechados e de definir regras torna a linguagem muito mais expressiva e o problema de obter informação da base de dados muito mais complexo e interessante.

A outra diferença entre um programa determinado e uma base de dados dedutiva (onde há possibilidade de definir regras) prende-se com as dimensões (relativas) do universo dos factos e do universo das regras. Tipicamente, uma base de dados contém milhares de instâncias específicas de tuplos relacionais e poucas regras para raciocinar sobre elas. Um programa determinado contém apenas um punhado de factos (frequentemente não fechados) e uma diversidade muito maior de cláusulas com corpo.

No entanto, estas diferenças – embora muito importantes na prática – são irrelevantes do ponto de vista teórico: do ponto de vista (por exemplo) da transformação de Herbrand associada a um programa determinado, é completamente acessório o número de factos que esse programa contém.

Ora numa base de dados não se põe a questão de como lidar com informação negativa: por defeito, todas as relações que não constam explicitamente da base de dados não se verificam. A generalização desta ideia conduz ao seguinte princípio, muito utilizado em Inteligência Artificial.

Postulado 6.1.1 *(Hipótese do Mundo Fechado.)* Sejam P um programa determinado e A um átomo fechado. Se $P \not\models A$, então $P \models \neg A$.

Como o nome indica, a motivação é assumir que o universo é fechado e que toda a informação que não está explicitamente expressa não se verifica.

O maior inconveniente da Hipótese do Mundo Fechado é ser uma regra de inferência não monótona.

Exemplo 6.1.2 Seja P o programa determinado seguinte.

$$p(a) \longleftarrow$$
$$p(b) \longleftarrow$$
$$q(c) \longleftarrow$$

Claramente, $P \not\models p(c)$, donde pela Hipótese do Mundo Fechado se infere $P \models \neg p(c)$. Seja agora Q o programa determinado que se obtém de P acrescentando a seguinte cláusula.

$$p(x) \longleftarrow q(x)$$

Tem-se agora que $Q \models p(c)$, donde $Q \not\models \neg p(c)$. Ou seja, a relação de consequência semântica não é monótona na presença da Hipótese do Mundo Fechado.

A outra questão que se põe é a de saber se com esta regra adicional se consegue decidir se $P \models \neg A$. Em geral, tal não se verifica. A "solução" para este problema, para garantir que a teoria continua a ter uma aplicação computacional útil, é impor a restrição adicional $A \in \mathcal{B}_P$ – ou seja, a Hipótese do Mundo Fechado só é aplicável a átomos fechados. Observe-se, mais uma vez, que isto é natural no contexto das Bases de Dados.

Definição 6.1.3 Sejam P um programa determinado e G um objectivo determinado. Uma árvore-SLD *finitamente falhada* para P e G é uma árvore-SLD para P e G que é finita e não contém ramos bem sucedidos.

O *conjunto de falhas finitas* de P é o conjunto dos átomos $A \in \mathcal{B}_P$ tais que existe uma árvore-SLD finitamente falhada de P e A.

É importante observar que a quantificação existencial na última definição é estritamente mais fraca que uma quantificação universal: a Independência da Regra garante que se A está no conjunto de falhas finitas de P então nenhuma árvore-SLD de P e A tem ramos bem sucedidos, mas pode haver (e em geral há) árvores-SLD de P e A com ramos infinitos.

Em termos implementacionais, para provar $\neg A$ constrói-se uma árvore-SLD de P e $\longleftarrow A$. Se esta tiver um ramo bem sucedido a prova falha; se em tempo finito se tiver explorado a árvore-SLD toda sem encontrar soluções, o objectivo é bem sucedido. Este procedimento implementa a seguinte regra de inferência.

Definição 6.1.4 A regra da Negação por Falha (NF) consiste em inferir $P \models \neg A$ quando A está no conjunto de falhas finitas de P.

Não é claro que haja maneira de, em geral, conseguir concluir operacionalmente mais informação negativa a partir de um programa...

A noção de conjunto de falhas finitas é uma noção operacional, inserida no contexto da resolução-SLD. Discute-se agora uma contrapartida denotacional desta noção, relacionada com potências *descendentes* da transformação de Herbrand.

Definição 6.1.5 Seja P um programa determinado. O conjunto $\mathcal{F}_P^d \subseteq \mathcal{B}_P$ das *falhas* de P à profundidade d define-se indutivamente como se segue.

- $A \in \mathcal{F}_P^1$ se $A \notin \mathcal{T}_P \downarrow 1$.
- $A \in \mathcal{F}_P^{d+1}$ se para cada cláusula $B \longleftarrow B_1, \ldots, B_q$ de P e para cada substituição θ tal que $B \equiv A\theta$ e $B_1\theta, \ldots, B_q\theta$ são fechadas existe k para o qual $B_k\theta \in \mathcal{F}_P^d$.

O *conjunto de falhas* de P é $\mathcal{F}_P = \bigcup_{d=1}^{\infty} \mathcal{F}_P^d$.

O resultado seguinte é muito simples de provar.

Proposição 6.1.6 Seja P um programa determinado. Então $\mathcal{F}_P = \mathcal{B}_P \setminus \mathcal{T}_P \downarrow \omega$.

Prova. Começa por se mostrar que $\mathcal{F}_P^d = \mathcal{B}_P \setminus \mathcal{T}_P \downarrow d$ por indução em d. Para $d = 1$ o resultado é imediato por definição de $\mathcal{F}_P^1$.

Suponha-se que $\mathcal{F}_P^d = \mathcal{B}_P \setminus \mathcal{T}_P \downarrow d$. Para $A \in \mathcal{B}_P$, tem-se que $A \in \mathcal{T}_P \downarrow (d+1)$ sse existirem uma cláusula $B \longleftarrow B_1, \ldots, B_q$ de P e uma substituição θ tais que $B\theta \equiv A$ e $\{B_1\theta, \ldots, B_q\theta\} \subseteq \mathcal{T}_P \downarrow d$. Então $A \notin \mathcal{T}_P \downarrow (d+1)$ sse para todas as cláusulas $B \longleftarrow B_1, \ldots, B_q$ de P e todas as substituições θ tais que $B\theta \equiv A$ e $B_1\theta, \ldots, B_q\theta$ é fechada existir k para o qual $B_k\theta \notin \mathcal{T}_P \downarrow d$. Mas por hipótese de indução a última parte da condição é equivalente a $B_k\theta \in \mathcal{F}_P^d$, donde se conclui que $A \notin \mathcal{T}_P \downarrow (d+1)$ sse $A \in \mathcal{F}_P^{d+1}$. Logo $\mathcal{F}_P^{d+1} = \mathcal{B}_P \setminus \mathcal{T}_P \downarrow (d+1)$.

Finalmente,

$$\mathcal{F}_P = \bigcup_{d=1}^{\infty} \mathcal{F}_P^d = \bigcup_{d=1}^{\infty} (\mathcal{B}_P \setminus \mathcal{T}_P \downarrow d) = \mathcal{B}_P \setminus \left(\bigcup_{d=1}^{\infty} \mathcal{T}_P \downarrow d \right) = \mathcal{B}_P \setminus \mathcal{T}_P \downarrow \omega,$$

o que conclui a prova. □

De uma forma semelhante prova-se a primeira relação entre o conjunto de falhas finitas e a transformação de Herbrand.

Lema 6.1.7 Sejam P um programa determinado e $A_1, \ldots, A_p$ átomos fechados. Se existe uma árvore-SLD finitamente falhada de profundidade n para P e $\longleftarrow A_1, \ldots, A_p$ então para algum i existe uma árvore-SLD finitamente falhada para P e $\longleftarrow A_i$ de profundidade menor ou igual a n.

Prova. Por indução em n. Se $n = 1$ então $p = 1$ e o resultado é trivial.

Suponha-se agora que o resultado é válido para árvores-SLD finitamente falhadas de profundidade menor ou igual a n e que existe uma árvore-SLD finitamente falhada de P e $\longleftarrow A_1, \ldots, A_p$ com profundidade n. Seja A_j o átomo seleccionado no primeiro passo; cada descendente da raiz dá uma árvore-SLD finitamente falhada de profundidade n. Aplicando a hipótese de indução a estas árvores-SLD, há dois casos a considerar.

- Se a partir de alguma das árvores-SLD se obtiver uma árvore-SLD finitamente falhada para P e $\longleftarrow A_i$, com $i \neq j$, então o resultado é imediatamente provado.
- Caso contrário, de cada sub-árvore-SLD obtém-se uma árvore-SLD finitamente falhada de profundidade menor ou igual a n para P e um dos átomos no resolvente de $\longleftarrow A_j$ e C_1. Pode-se então construir uma árvore-SLD finitamente falhada para P e $\longleftarrow A_j$ com profundidade menor ou igual a $n+1$ juntando todas essas árvores-SLD a um nó com o objectivo $\longleftarrow A_j$.

□

Teorema 6.1.8 *(Apt & van Emden.)* Sejam P um programa determinado e $A \in \mathcal{B}_P$. Se existe uma árvore-SLD finitamente falhada de P e $\longleftarrow A$ com profundidade (máxima) n, então $A \notin \mathcal{T}_P \downarrow n$.

Prova. Por indução em n. Se P e $\longleftarrow A$ tem uma árvore-SLD finitamente falhada de profundidade 1 então A não é unificável com a cabeça de nenhuma cláusula de P, donde $A \notin \mathcal{T}_P \downarrow 1$.

Suponha-se agora que o resultado é válido para árvores-SLD finitamente falhadas de comprimento n e suponha-se que existe uma árvore-SLD finitamente falhada de comprimento $n+1$ para P e $\longleftarrow A$. Sejam $B \longleftarrow B_1, \ldots, B_q$ uma cláusula de P e θ uma substituição tal que $B\theta \equiv A$ e $B_1\theta, \ldots, B_q\theta$ são fechadas; se γ for uma UMG de A e B, então num dos descendentes de $\longleftarrow A$ há uma sub-árvore-SLD finitamente falhada para P e $\longleftarrow B_1\gamma, \ldots, B_q\gamma$ de profundidade n.

Como γ é uma UMG de A e B e θ unifica estas duas cláusulas, um raciocínio semelhante ao da prova do Lema da Elevação permite concluir que existe uma árvore-SLD finitamente falhada para P e $\longleftarrow B_1\theta, \ldots, B_q\theta$ de profundidade n.

Pelo Lema 6.1.7, existe uma árvore-SLD finitamente falhada para P e $\longleftarrow B_i\theta$ de profundidade menor ou igual a n, para algum i. Por hipótese de indução, $B_i\theta \notin \mathcal{T}_P \downarrow n$. Uma vez que isto é válido para todas as cláusulas cuja cabeça é unificável com A, conclui-se que $A \notin \mathcal{T}_P \downarrow (n+1)$. □

Corolário 6.1.9 Sejam P um programa determinado e $A \in \mathcal{B}_P$. Se existe uma árvore-SLD finitamente falhada de P e $\longleftarrow A$ com profundidade (máxima) n, então $A \in \mathcal{F}_P^n$.

Prova. Imediato pela Proposição 6.1.6. □

Para obter um recíproco deste resultado é necessário ter algum cuidado, já que (mais uma vez) o lema da Independência da Regra não garante que todas as árvores-SLD de P e G sejam finitamente falhadas desde que uma o seja devido à possível existência de ramos infinitos. Interessa portanto caracterizar funções de selecção que encontram preferencialmente árvores-SLD finitamente falhadas.

Definição 6.1.10 Uma derivação-SLD diz-se *justa* se ou é uma derivação-SLD falhada ou todo o átomo que ocorre na derivação-SLD é escolhido ao fim de um número finito de passos.

Uma função de selecção diz-se *justa* se só produz derivações-SLD justas.

A noção de justiça ocorre muito frequentemente no estudo de propriedades de segurança de programas e destina-se por exemplo a evitar situações de *deadlock*. Um exemplo de um sistema que se pretende justo é um sistema operativo: todos os processos que requerem tempo de processador devem recebê-lo em tempo finito, evitando que um único processo impeça os outros de serem executados. Analogamente, para uma derivação-SLD justa ser infinita é porque *nenhum* átomo pode ser desenvolvido por forma a se chegar a um impasse em tempo finito.

Observe-se que a função de selecção usual (escolher sempre o átomo mais à esquerda) não é justa.

Proposição 6.1.11 Se existe uma derivação-SLD justa e não falhada para $\longleftarrow A_1, \ldots, A_q$ com sucessão de unificadoras $\Theta = \{\theta_j\}_{j=1}^{\infty}$ então para qualquer $k \in \mathbb{N}$ existe um natural n tal que $[A_i\theta_1 \ldots \theta_n] \subseteq \mathcal{T}_P \downarrow k$ para $i = 1, \ldots, q$.

Prova. Se a derivação-SLD for finita, o resultado é consequência das Proposições 5.3.2 e 4.3.9, já que $[A_i\theta_1 \ldots \theta_n] \subseteq \mathcal{T}_P \uparrow n \subseteq \mathrm{pfmin}(\mathcal{T}_P) \subseteq \mathcal{T}_P \downarrow k$.

Suponha-se agora que a derivação-SLD é infinita. O resultado prova-se por indução em k. Para $k = 0$ o resultado é trivial, já que $\mathcal{T}_P \downarrow 0 = \mathcal{B}_P$.

Se $k > 0$, escolha-se i e seja p o passo em que A_i é seleccionado; a existência de p é consequência de a derivação-SLD ser justa. Então $G_p \equiv \longleftarrow B_1, \ldots, B_q$, em que $B_j, \ldots, B_m$ são instâncias fechadas por θ_p do corpo duma cláusula de P cuja cabeça unifica por θ_p com A_i. Por hipótese de indução, para $r = j, \ldots, m$ existe s_r tal que $[B_r\theta_{p+1} \ldots \theta_{p+s_r}] \subseteq \mathcal{T}_P \downarrow (k-1)$. Tomando $s = \max\{s_j, \ldots, s_m\}$, qualquer instância fechada de $[A_i\theta_1, \ldots, \theta_s]$ é a cabeça duma instância fechada duma cláusula de P cujo corpo está contido em $\mathcal{T}_P \downarrow (k-1)$, logo essa instância está em $\mathcal{T}_P \downarrow k$. □

Como consequência prova-se que o conjunto de falhas finitas e o conjunto de falhas coincidem.

Corolário 6.1.12 Sejam P um programa determinado e $A \in \mathcal{B}_P$. As seguintes condições são equivalentes.

(i) $A \in \mathcal{F}_P$

(ii) $A \notin \mathcal{T}_P \downarrow \omega$

(iii) A está no conjunto de falhas finitas de P

(iv) Toda a árvore-SLD para P e $\longleftarrow A$ com função de selecção justa é finitamente falhada.

Prova.

$(i) \longleftrightarrow (ii)$ Proposição 6.1.6.

$(iv) \longrightarrow (iii)$ Por definição de conjunto de falhas finitas.

$(iii) \longrightarrow (ii)$ Teorema de Apt & van Emden.

$(ii) \longrightarrow (iv)$ Contra-recíproco da Proposição 6.1.11.

□

6.2 Completação de programas

O objectivo desta secção é relaxar as condições na definição de programa para considerar programas cujas cláusulas são construídas à custa de literais arbitrários. O estudo destes programas requer definir a sua completação – um conjunto de fórmulas que expressa logicamente a interpretação pretendida da negação.

Definição 6.2.1 Uma *cláusula generalizada* é uma cláusula da forma $A \longleftarrow L_1, \ldots, L_q$ em que A é um átomo e $L_1, \ldots, L_q$ são literais. Um *programa generalizado* é um conjunto finito de cláusulas generalizadas. Um *objectivo generalizado* é uma cláusula da forma $\longleftarrow L_1, \ldots, L_q$.

Observe-se que toda a cláusula com pelo menos um literal positivo pode ser escrita como uma cláusula generalizada; mais, toda a cláusula pode ser escrita como um objectivo generalizado. Contrariamente à situação até aqui, o facto de se permitir a presença de literais negativos no corpo da cláusula cria uma diferença entre a interpretação lógica das cláusulas (como disjunção de literais) e a interpretação operacional (em que a cabeça é vista como uma conclusão e o corpo como uma condição). Concretamente, as cláusulas $p(x) \longleftarrow q(x), \neg r(x)$ e $r(x) \longleftarrow q(x), \neg p(x)$ são logicamente equivalentes (correspondem a $\forall x.(p(x) \vee \neg q(x) \vee r(x))$) mas não operacionalmente, já

que a primeira é vista como parte da definição do predicado p e a segunda como parte da definição do predicado r.

O seguinte exemplo ilustra um problema fundamental dos programas generalizados.

Exemplo 6.2.2 Seja P o seguinte programa generalizado.

$$p(a) \longleftarrow \neg p(a)$$

Então $\mathcal{T}_P$ não é monótona: $\emptyset \subseteq \{p(a)\}$ e no entanto $\mathcal{T}_P(\emptyset) = \{p(a)\} \not\subseteq \emptyset = \mathcal{T}_P(\{p(a)\}) = \emptyset$.

Mais adiante mostrar-se-á que, não obstante, se consegue obter uma semântica declarativa para programas generalizados.

Definição 6.2.3 Sejam P um programa generalizado e p um símbolo de predicado que ocorra numa cláusula de P. A *completação* de p relativa a P define-se como se segue.

(i) Para cada cláusula generalizada $C \equiv p(t_1, \ldots, t_n) \longleftarrow L_1, \ldots, L_q$ pertencente à definição de p, define-se C^* como sendo a fórmula

$$p(x_1, \ldots, x_n) \longleftarrow \exists y_1 \ldots \exists y_m.(x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge L_1 \wedge \ldots \wedge L_q)$$

onde $y_1, \ldots, y_m$ são todas as variáveis que ocorrem em C e $x_1, \ldots, x_n$ são variáveis que não ocorrem em C.

(ii) Se a definição de p for constituída por cláusulas $C_1, \ldots, C_k$ com $k \geq 1$ e para cada i se tiver $C_i^* \equiv p(x_1, \ldots, x_n) \longleftarrow E_i$, então a *definição completada* de p é

$$\forall(p(x_1, \ldots, x_n) \Leftrightarrow (E_1 \vee \ldots \vee E_k))$$

escolhendo as mesmas variáveis $x_1, \ldots, x_n$ em todas as cláusulas C_i^*.

(iii) Se a definição de p for vazia, então a definição completada de p é $\forall(\neg p(x_1, \ldots, x_n))$.

Esta definição captura a ideia por detrás do conceito de definição de um predicado: $p(t_1, \ldots, t_n)$ é verdadeiro se (e somente se) houver uma instância fechada duma cláusula de P com cabeça $p(t_1, \ldots, t_n)$ cujo corpo é verdadeiro.

Repare-se que até agora as cláusulas tinham sempre sido interpretadas como condições suficientes: a completação dum programa vê-as também como condições necessárias.

O exemplo seguinte ilustra o funcionamento deste mecanismo.

Exemplo 6.2.4 Seja P o programa generalizado seguinte.

$$\begin{array}{l} p(y) \longleftarrow q(y), \neg r(a, y) \\ p(f(z)) \longleftarrow \neg q(z) \\ p(b) \longleftarrow \\ q(a) \longleftarrow \end{array}$$

As primeiras três cláusulas constituem a definição de p; uma vez que x é uma variável que não ocorre em nenhuma delas, podem-se transformar como acima descrito nas seguintes cláusulas.

$$\begin{array}{l} p(x) \longleftarrow \exists y.(x = y \wedge q(y) \wedge \neg r(a, y)) \\ p(x) \longleftarrow \exists z.(x = f(z) \wedge \neg q(z)) \\ p(x) \longleftarrow x = b \end{array}$$

Aplicando o segundo passo, chega-se à seguinte fórmula para definição completada de p.

$$\begin{aligned}\forall x.(p(x) \;\Leftrightarrow\; & ((\exists y.(x = y \wedge q(y) \wedge \neg r(a, y))) \\ & \vee(\exists z.(x = f(z) \wedge \neg q(z))) \\ & \vee(x = b)))\end{aligned}$$

Analogamente, as definições completadas de q e de r são as seguintes.

$$\forall x.(q(x) \Leftrightarrow x = a)$$
$$\forall x \forall y.\neg r(x, y)$$

Para definir a completação dum programa generalizado também é necessário especificar o predicado $=$.

Definição 6.2.5 A *teoria da igualdade* sobre uma assinatura de primeira ordem Σ é o conjunto de fórmulas que contém as seguintes fórmulas.

1. $c \neq d$ para todos os símbolos de constante c e d distintos.
2. $\forall(f(x_1, \ldots, x_m) \neq g(y_1, \ldots, y_n))$ para todos os símbolos de função f (de aridade m) e g (de aridade n) distintos.
3. $\forall(f(x_1, \ldots, x_m) \neq c)$ para todos os símbolos de constante c e de função f de aridade m.
4. $\forall(t[x] \neq x)$ para todo o termo $t[x]$ não variável que contenha x.
5. $\forall((x_1 \neq y_1 \vee \ldots \vee x_m \neq y_m) \Rightarrow f(x_1, \ldots, x_m) \neq f(y_1, \ldots, y_m))$ para todo o símbolo de função f de aridade m.
6. $\forall(x = x)$
7. $\forall((x_1 = y_1 \wedge \ldots \wedge x_m = y_m) \Rightarrow f(x_1, \ldots, x_m) = f(y_1, \ldots, y_m))$ para todo o símbolo de função f de aridade m.
8. $\forall((x_1 = y_1 \wedge \ldots \wedge x_m = y_m) \Rightarrow (p(x_1, \ldots, x_m) \Rightarrow p(y_1, \ldots, y_m)))$ para todo o símbolo de predicado f de aridade m (em particular para $=$).

As fórmulas das primeiras cinco famílias garantem essencialmente que a interpretação da igualdade é injectiva nos termos. As três restantes famílias exprimem a teoria usual da igualdade em Lógica de Primeira Ordem.

Proposição 6.2.6 Seja J um modelo da teoria da igualdade. Então $=_J$ é uma relação de equivalência.

Prova.

- *Reflexividade.* É consequência imediata do sexto axioma da lista anterior.
- *Simetria.* Tomando $=$ para p, $x_1 \equiv x_2 \equiv y_1 \equiv x$ e $y_1 \equiv y$ conclui-se do oitavo axioma que J é modelo de $\forall x \forall y.((x = y \wedge x = x) \Rightarrow (x = x \Rightarrow y = x))$; usando a reflexividade conclui-se que $=_J$ é simétrica.
- *Transitividade.* Tomando no mesmo axioma $x_1 \equiv x_2 \equiv x$, $y_1 \equiv y$ e $y_2 \equiv z$ conclui-se analogamente que J é modelo de $\forall x \forall y \forall z.((x = y \wedge x = z) \Rightarrow (x = x \Rightarrow y = z))$; usando a simetria (no antecedente) e a reflexividade (no antecedente do consequente) conclui-se que J também satisfaz $\forall x \forall y \forall z.((y = x \wedge x = z) \Rightarrow y = z)$ e portanto $=_J$ é transitiva.

□

Proposição 6.2.7 Seja H uma estrutura de Herbrand para uma assinatura Σ que é modelo da teoria da igualdade. Então $=_H$ é a relação $\Delta_{\mathcal{U}_\Sigma}$.

Prova. Pela Proposição anterior, a interpretação de $=_H$ é uma relação de equivalência, em particular reflexiva, donde $\Delta_{\mathcal{U}_\Sigma} \subseteq H$. A inclusão inversa prova-se por indução na estrutura dos elementos de $\mathcal{U}_\Sigma$: se $t_1 =_H t_2$ e t_1 for uma constante, então t_2 também o é (pelo axioma 3) e tem de ser a mesma constante (pelo axioma 1). Se $t_1 \equiv f(t'_1, \ldots, t'_m)$ então t_2 não pode ser uma constante (pelo axioma 3) e tem de ter a forma $f(u'_1, \ldots, u'_m)$ (pelo axioma 2). Pelo axioma 5 necessariamente $t'_i =_H u'_i$ para $i = 1, \ldots, m$, donde a hipótese de indução implica que $t'_i \equiv u'_i$ e portanto de novo $t_1 \equiv t_2$. □
Contrariamente ao que possa parecer, o axioma 4 (que não foi utilizado nestas provas) não é supérfluo: garante que fórmulas como $\exists x.f(x) = x$, que atendendo a esta Proposição não são satisfeitas em estruturas de Herbrand, são contraditórias quando adicionadas à teoria da igualdade.

Definição 6.2.8 Seja P um programa generalizado. A *completação de P* é o conjunto $\overline{P}$ formado pelas definições completadas de todos os símbolos de predicado que ocorrem em P e pela teoria da igualdade.

A teoria "correcta" da Programação em Lógica com programas generalizados trabalha com a *completação* dos programas em vez dos próprios programas.

É importante observar que a completação dum programa é um conjunto infinito de fórmulas desde que o programa tenha um símbolo de função: há uma infinidade de termos contendo cada variável e todos eles geram um axioma do quarto grupo. Outra observação importante é o facto de a completação de um programa poder ser um conjunto contraditório.

Exemplo 6.2.9 Seja P o programa generalizado seguinte.

$$p(a) \longleftarrow \neg p(a)$$

Então $\overline{P}$ contém a fórmula

$$\forall x.\,(p(x) \Leftrightarrow (x = a \wedge \neg p(a)))$$

que é contraditória com a teoria da igualdade.

Note-se que a completação de um programa generaliza o programa original.

Proposição 6.2.10 Seja P um programa generalizado. Então $\overline{P} \models P$.

Prova. Seja M um modelo de $\overline{P}$ e seja $C \equiv p(t_1, \ldots, t_n) \longleftarrow L_1, \ldots, L_q$ uma cláusula generalizada de P.

Seja ρ uma atribuição. Se $[\![L_i]\!]_{M,\rho} = 0$ para algum i então $[\![C]\!]_{M,\rho} = 1$, portanto suponha-se que $[\![L_i]\!]_{M,\rho} = 1$ para $i = 1, \ldots, q$.

Seja E a fórmula existencial na definição completada de p que proveio de C. Então E é da forma $\exists y_1 \ldots \exists y_k.(x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge L_1 \wedge \ldots L_q)$; tomando $\rho' = \rho\left[{}^{x_1}/_{t_1\rho}\right] \ldots \left[{}^{x_n}/_{t_n\rho}\right]$, tem-se que $[\![x_i = t_i]\!]_{M,\rho'} = 1$ para $i = 1, \ldots, n$ e $[\![L_j]\!]_{M,\rho'} = 1$ para $j = 1, \ldots, q$ (pois os x_is não ocorrem em L_j). Logo $[\![E]\!]_{M,\rho} = 1$, donde M satisfaz um dos elementos da disjunção à direita na definição

completada de p e portanto $[\![p(x_1, \ldots, x_n)]\!]_{M,\rho'} = 1$. Finalmente, como $[\![x_i]\!]_{M,\rho'} = [\![t_i]\!]_{M,\rho}$, conclui-se que $[\![p(t_1, \ldots, t_n)]\!]_{M,\rho} = 1$ e portanto M é modelo de C. □

Corolário 6.2.11 Sejam P um programa generalizado e F uma fórmula. Se $P \models F$ então $\overline{P} \models F$.

Prova. Pela Proposição 6.2.10, todo o modelo de $\overline{P}$ é modelo de P, logo se $P \models F$ então todo o modelo de $\overline{P}$ é ainda modelo de F. □

Definição 6.2.12 Sejam P um programa generalizado e $G \equiv \longleftarrow L_1, \ldots, L_q$ um objectivo generalizado. Uma *resposta correcta* de $\overline{P}$ a G é uma substituição θ tal que $\overline{P} \models \forall((L_1 \wedge \ldots L_q)\theta)$.

Corolário 6.2.13 Sejam P um programa generalizado e G um objectivo generalizado. Se θ é uma resposta correcta de P a G então θ é resposta correcta de $\overline{P}$ a G.

Prova. Caso particular do Corolário 6.2.11. □

6.3 Resolução-SLDNF

O mecanismo da resolução-SLD pode ser estendido para incluir o tratamento de literais negativos – por outras palavras, por forma a ser aplicável a programas e objectivos generalizados. Nesta secção descreve-se a semântica operacional que se obtém acrescentando à resolução-SLD a regra da Negação por Falha; ver-se-á mais adiante que as condições impostas sobre as funções de selecção não são facilmente eliminâveis.

Definição 6.3.1 Uma função de selecção $\mathcal{S}$ diz-se *segura* se sempre que $\mathcal{S}(\longleftarrow L_1, \ldots, L_q) = L_i$ e L_i for um literal negativo, então L_i é fechado.

Definição 6.3.2 Sejam P um programa generalizado, G_0 um objectivo generalizado e $\mathcal{S}$ uma função de selecção segura. Uma *derivação-SLDNF* de P e G_0 via $\mathcal{S}$ é um triplo $\langle \mathcal{G}, \mathcal{C}, \Theta \rangle$ satisfazendo as seguintes condições.

- $\mathcal{G}$ é uma sequência $G_0, G_1, \ldots$ de objectivos generalizados.
- $\mathcal{C}$ é uma sequência $C_1, C_2, \ldots$ de variantes de cláusulas generalizadas de P.
- Θ é uma sequência $\theta_1, \theta_2, \ldots$ de substituições.
- Para cada i, se $\mathcal{S}(G_i)$ é um literal positivo, então G_{i+1} é um resolvente de G_i e C_{i+1} usando θ_{i+1}.
- Para cada i, se $\neg A \equiv \mathcal{S}(G_i)$ é um literal negativo fechado, seja $\mathcal{A}$ a árvore-SLDNF de P e $\longleftarrow A$. Se $\mathcal{A}$ for finitamente falhada, então G_{i+1} é G_i sem o objectivo $\neg A$ e $\theta_{i+1} = \varepsilon$; se $\mathcal{A}$ tiver um ramo bem sucedido então a derivação termina.
- Para cada i, C_i é escolhido por forma a não conter nenhuma variável que já tenha ocorrido em $G_0, \ldots, G_i, C_1, \ldots, C_{i-1}$.
- As sequências G, C e Θ ou são infinitas ou terminam em G_n, C_n e θ_n respectivamente, com $G_n = \square$ ou $\mathcal{S}(G_n)$ não unificável com a cabeça de nenhuma cláusula de P ou raíz de uma árvore-SLDNF com ramos bem sucedidos.

Uma derivação-SLDNF finita que termina com a cláusula vazia diz-se uma *refutação-SLDNF* de P e G; o natural n tal que $G_n = \square$ diz-se o *comprimento* da refutação-SLDNF. Neste caso, a substituição θ obtida restringindo a composição $\theta_1 ... \theta_n$ às variáveis que ocorrem em G_0 diz-se uma *resposta calculada* de P a G_0 via $\mathcal{S}$.

É importante observar que os literais negativos nunca podem criar instanciações de variáveis. Na Secção 6.5 ver-se-á que esta restrição traz limitações ao poder computacional da resolução-SLDNF, mas é infelizmente necessária. Em particular, se θ for uma resposta calculada de P a G então $L\theta$ é fechado para todo o subobjectivo negativo L de G.

Tal como anteriormente, podem-se juntar todas as derivações-SLDNF possíveis numa única árvore.

Definição 6.3.3 Sejam P um programa generalizado, G um objectivo generalizado e $\mathcal{S}$ uma função de selecção. A *árvore-SLDNF* de P e G via $\mathcal{S}$ é uma árvore etiquetada construída da seguinte forma:

- a etiqueta de cada nó é um objectivo;
- a etiqueta da raiz é G;
- para cada nó com etiqueta $G' \equiv \longleftarrow L_1, ..., L_p$, se $\mathcal{S}(\longleftarrow L_1, ..., L_p) = L_m$ for um literal positivo o nó tem um descendente por cada cláusula $C \equiv L \longleftarrow L'_1, ..., L'_q$ de P cuja cabeça seja unificável com L_m, sendo cada descendente etiquetado pelo resolvente de C e G' usando uma UMG de L_m e L;
- para cada nó com etiqueta $G' \equiv \longleftarrow L_1, ..., L_p$, se $\mathcal{S}(\longleftarrow L_1, ..., L_p) = L_m \equiv \neg A$ for um literal negativo fechado o nó tem um único descendente etiquetado com $\longleftarrow L_1, ..., L_{m-1}, L_{m+1}, ..., L_p$ se a árvore-SLDNF de P e $\longleftarrow A$ for finitamente falhada; caso contrário não tem descendentes.

Tal como anteriormente, um ramo cuja folha está etiquetada pela cláusula vazia $\square$ diz-se *bem sucedido*, um ramo com uma folha etiquetada com outra cláusula diz-se *falhado* e os restantes ramos dizem-se ramos *infinitos*. Uma árvore-SLDNF cujos ramos são todos finitos e falhados diz-se uma árvore-SLDNF *finitamente falhada*.

A correcção da resolução-SLDNF é relativamente simples de provar, requerendo no entanto alguns resultados prévios.

Lema 6.3.4 Sejam Σ uma assinatura de primeira ordem e $p(s_1, ..., s_n)$ e $p(t_1, ..., t_n)$ átomos sobre essa assinatura.

(i) Se $p(s_1, ..., s_n)$ e $p(t_1, ..., t_n)$ não são unificáveis, então a fórmula

$$\neg\exists(s_1 = t_1 \wedge ... \wedge s_n = t_n)$$

é consequência da teoria da igualdade.

(ii) Se $p(s_1, ..., s_n)$ e $p(t_1, ..., t_n)$ são unificáveis com UMG $\theta = \{x_1/r_1, ..., x_k/r_k\}$ encontrada pelo algoritmo de unificação, então a fórmula

$$\forall((s_1 = t_1 \wedge ... \wedge s_n = t_n) \Leftrightarrow (x_1 = r_1 \wedge ... \wedge x_k = r_k))$$

é consequência da teoria da igualdade.

Prova. Em primeiro lugar, observe-se que por definição de unificadora se θ unifica $p(s_1, \ldots, s_n)$ e $p(t_1, \ldots, t_n)$ então tem-se

$$\forall((s_1 = t_1 \wedge \ldots \wedge s_n = t_n) \Leftarrow (x_1 = r_1 \wedge \ldots \wedge x_k = r_k))$$

usando o facto de $=$ ser reflexiva e uma congruência para as funções e predicados (prova por indução na estrutura de $p(t_1, \ldots, t_n)\theta$).

As outras fórmulas provam-se por indução no número de vezes que o ciclo do algoritmo de unificação é executado; observe-se que, caso o algoritmo termine com sucesso, este número é precisamente k.

Sejam r e r' elementos do conjunto de conflitos de $p(s_1, \ldots, s_n)$ e $p(t_1, \ldots, t_n)$. Usando o contra-recíproco do axioma 5 da teoria da igualdade conclui-se da definição de conjunto de conflitos que $\forall((s_1 = t_1 \wedge \ldots \wedge s_n = t_n) \Rightarrow r = r')$. Em particular, se $k = 1$ e $\{x_1/r_1\}$ for uma UMG de $p(s_1, \ldots, s_n)$ e $p(t_1, \ldots, t_n)$ encontrada pelo algoritmo, então conclui-se $\forall((s_1 = t_1 \wedge \ldots \wedge s_n = t_n) \Rightarrow x_1 = t_1)$. De modo semelhante, se o algoritmo falhar prova-se $\forall(r \neq r')$ usando os quatro primeiros axiomas, donde se conclui $\neg\exists(s_1 = t_1 \wedge \ldots \wedge s_n = t_n)$.

Suponha-se agora que o resultado é válido quando o algoritmo termina ao fim de m passos e seja $k = m + 1$. Seja $\theta_1 = \{x_1/r'_1\}$ a primeira substituição construída pelo algoritmo; então a hipótese de indução é aplicável a $p(s_1, \ldots, s_n)\theta_1$ e $p(t_1, \ldots, t_n)\theta_1$, ou seja, ao resto da execução do algoritmo.

Se o algoritmo falhar, então a fórmula $\neg\exists(s_1\theta_1 = t_1\theta_1 \wedge \ldots \wedge s_n\theta_1 = t_n\theta_1)$ é consequência da teoria da igualdade, donde se conclui que $\neg\exists(s_1 = t_1 \wedge \ldots \wedge s_n = t_n)$ também o é (se ρ for uma atribuição que satisfaz a conjunção da última fórmula, então ρ' tal que $\rho'(x) = \rho(x\theta_1)$ satisfaria a conjunção da fórmula anterior).

Se o algoritmo terminar com sucesso então por hipótese de indução a teoria da igualdade permite deduzir

$$\forall((s_1\theta_1 = t_1\theta_1 \wedge \ldots \wedge s_n\theta_1 = t_n\theta_1) \Rightarrow (x_2 = r_2 \wedge \ldots \wedge x_k = r_k));$$

pela observação do início da prova tem-se também

$$\forall((s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge x = r'_1) \Rightarrow (s_1\theta_1 = t_1\theta_1 \wedge \ldots \wedge s_n\theta_1 = t_n\theta_1)).$$

Por outro lado, da definição de conjunto de conflitos tem-se como atrás que $\forall((s_1 = t_1 \wedge \ldots \wedge s_n = t_n) \Rightarrow x_1 = r'_1)$ é consequência da teoria da igualdade. Sendo $\gamma = \{x_2/r_2, \ldots, x_k/r_k\}$, prova-se que $r_1 = r'_1\gamma$, donde a teoria da igualdade prova ainda $\forall((x_2 = r_2 \wedge \ldots \wedge x_k = r_k) \Rightarrow r_1 = r'_1)$ (por indução na estrutura de r_1). De todas estas fórmulas conclui-se finalmente que

$$\forall((s_1 = t_1 \wedge \ldots \wedge s_n = t_n) \Rightarrow (x_1 = r_1 \wedge \ldots \wedge x_k = r_k))$$

é consequência da teoria da igualdade. □

Lema 6.3.5 Sejam P um programa generalizado e G um objectivo generalizado.

(i) Se o literal seleccionado em G é positivo e a raiz da árvore-SLDNF de P e G não tem descendentes, então $\overline{P} \models G$.

(ii) Se o literal seleccionado em G é positivo e a raiz da árvore-SLDNF de P e G tem descendentes etiquetados por $G_1, \ldots, G_q$ então $\overline{P} \models G \Leftrightarrow (G_1 \wedge \ldots \wedge G_q)$.

Prova. Observe-se em primeiro lugar que se $G \equiv \longleftarrow L_1, \ldots, L_q$ então por definição G é a fórmula $\forall(\neg L_1 \vee \ldots \vee \neg L_q)$.

Seja $p(s_1, \ldots, s_n)$ o literal positivo seleccionado em G. Há três casos a considerar.

1. Se não há cláusulas em P cuja cabeça utiliza o símbolo de predicado p, então o caso (i) aplica-se. Ora nesta situação a definição completada de p é $\forall(\neg p(x_1, \ldots, x_n))$, donde $\overline{P} \models G$.
2. Se há cláusulas em P cuja cabeça utiliza o símbolo de predicado p mas nenhuma dessas cabeças unifica com $p(s_1, \ldots, s_n)$, aplica-se ainda o caso (i). Nesta situação a definição completada de p é $\forall(p(x_1, \ldots, x_n) \Leftrightarrow E_1 \vee \ldots \vee E_k)$ em que cada E_j tem a forma

$$\exists y_{j,1} \ldots \exists y_{j,m_j}. \left(x_1 = t_{j,1} \wedge \ldots x_n = t_{j,n} \wedge L_{j,1} \wedge \ldots \wedge L_{j,p_j}\right)$$

   e $p(t_{j,1}, \ldots, t_{j,n}) \longleftarrow L_{j,1}, \ldots, L_{j,p_j}$ é uma cláusula de P. Uma vez que $p(t_{j,1}, \ldots, t_{j,n})$ não é unificável com $p(s_1, \ldots, s_n)$, o Lema 6.3.4 permite concluir que $\overline{P} \models \neg\exists(s_1 = t_{j,1} \wedge \ldots s_n = t_{j,n})$ e portanto nenhum dos E_j é satisfeito quando cada x_i está instanciado por s_i. Logo também neste caso $\overline{P} \models \forall(\neg p(s_1, \ldots, s_n))$, donde $\overline{P} \models G$.
3. Finalmente, usando a notação do caso anterior, suponha-se que $p(s_1, \ldots, s_n)$ unifica com $p(t_{j,1}, \ldots, t_{j,n})$ com UMG $\theta = \{x_1/r_1, \ldots, x_k/r_k\}$. Pelo Lema 6.3.4, a teoria da igualdade implica $\forall((s_1 = t_{j,1} \wedge \ldots \wedge s_n = t_{j,n}) \Leftrightarrow (x_1 = r_1 \wedge \ldots \wedge x_k = r_k))$. Ora $p(s_1, \ldots, s_n)$ é equivalente à disjunção dos E_js com x_i substituído por s_i; a fórmula anterior diz que se pode substituir E_j por $\exists(L_{j,1}\theta \wedge \ldots \wedge L_{j,p_j}\theta)$. Um raciocínio semelhante ao do caso anterior permite eliminar os E_js provenientes de cláusulas cujas cabeças não são unificáveis com $p(s_1, \ldots, s_n)$. É trivial concluir que $\overline{P} \models G \Leftrightarrow (G_1 \wedge \ldots \wedge G_q)$.

□

Proposição 6.3.6 *(Correcção da Negação por Falha.)* Sejam P um programa generalizado, G um objectivo generalizado e $\mathcal{S}$ uma função de selecção segura. Se existe uma árvore-SLDNF finitamente falhada para P e G via $\mathcal{S}$, então $\overline{P} \models G$.

Prova. Por indução no número de literais negativos que são seleccionados durante a construção da árvore-SLDNF para P e G. Se este número for 0, então a prova é uma consequência imediata do Lema 6.3.5 por indução na profundidade da árvore-SLDNF.

Suponha-se que o resultado é válido quando são seleccionados n literais negativos e que a construção da árvore-SLDNF finitamente falhada para P e G via $\mathcal{S}$ requer escolher $n+1$. Seja $\longleftarrow M_1, \ldots, M_q$ a etiqueta do nó mais acima na árvore-SLDNF onde é seleccionado um literal negativo. Por iteração do Lema 6.3.5, $\overline{P} \models G \Leftrightarrow (G_1 \wedge \ldots \wedge G_p)$, onde $G_1, \ldots, G_p$ são as etiquetas dos nós à altura de $\longleftarrow M_1, \ldots, M_q \equiv G_i$. Aplicando a hipótese de indução aos outros nós (cujas sub-árvores não podem requerer a selecção de mais do que n literais negativos) conclui-se que $\overline{P} \models G \Leftrightarrow \neg\exists(M_1 \wedge \ldots \wedge M_q)$. Resta então mostrar que $\overline{P} \models \neg\exists(M_1 \wedge \ldots \wedge M_q)$.

Seja $M_j \equiv \neg A$ o literal negativo fechado seleccionado neste objectivo. Há dois casos possíveis.

- O nó tem um descendente etiquetado por $\longleftarrow M_1, \ldots, M_{j-1}, M_{j+1}, \ldots, M_q$. Por hipótese de indução $G \models \neg\exists(M_1 \wedge \ldots M_{j-1} \wedge M_{j+1} \wedge \ldots \wedge M_q)$, donde $G \models \neg\exists(M_1 \wedge \ldots \wedge M_q)$.
- O nó não tem descendentes. Então existe uma refutação-SLDNF de P e $\longleftarrow A$ e esta tem no máximo n passos em que um literal negativo é seleccionado. Observe-se que A é fechado, donde $\exists A \equiv A$, e basta portanto mostrar que $\overline{P} \models A$ (pois $\exists A$ é equivalente a $\neg\exists M_j$). Este

facto prova-se mostrando que $\overline{P} \models (L_1 \wedge ... \wedge L_k) \Rightarrow A$, em que $\longleftarrow L_1, ..., L_k$ é qualquer objectivo que ocorra na refutação-SLDNF. Isto prova-se por indução na profundidade a que o objectivo ocorre.

O objectivo inicial é $\longleftarrow A$, tendo-se claramente $\overline{P} \models A \Rightarrow A$.

Suponha-se que $\overline{P} \models (L_1 \wedge ... \wedge L_k) \Rightarrow A$ e seja L_i o literal seleccionado. Se L_i é um literal negativo, então $\overline{P} \models L_i$ por hipótese de indução (da Proposição) donde $\overline{P} \models (L_1 \wedge ... \wedge L_k) \Leftrightarrow (L_1 \wedge ... \wedge L_{i-1} \wedge L_{i+1} \wedge ... \wedge L_k)$ e o resultado segue trivialmente. Se o objectivo seleccionado em L_i é um literal positivo e $C \equiv L_i \longleftarrow L'_1, ..., L'_m$ for a instância da cláusula seleccionada nesse passo, é imediato verificar que $\{C, L'_1, ..., L'_k\} \models L_i$, donde $P \models (L_1 \wedge ... \wedge L_{i-1} \wedge L'_1 \wedge ... \wedge L'_m \wedge L_{i+1} \wedge ... \wedge L_k) \Rightarrow (L_1 \wedge ... \wedge L_k)$ e pelo Corolário 6.2.11 a mesma fórmula é consequência semântica de $\overline{P}$, donde o resultado segue outra vez trivialmente.

Uma vez que toda a refutação-SLDNF termina com a cláusula vazia, conclui-se que $\overline{P} \models A$, já que a conjunção de zero átomos é trivialmente satisfeita.

□

Corolário 6.3.7 Seja P um programa determinado. Se $A \in \mathcal{F}_P$ então $\overline{P} \models \neg A$.

Prova. Pelo Corolário 6.1.12, se $A \in \mathcal{F}_P$ então toda a árvore-SLD com função de selecção justa para P e A é finitamente falhada. Uma vez que uma função de selecção para um programa determinado é sempre segura (pois não ocorrem literais negativos em nenhuma cláusula), a Proposição anterior garante que $\overline{P} \models \neg A$. □

Teorema 6.3.8 *(Correcção da resolução-SLDNF.)* Sejam P um programa generalizado, G um objectivo generalizado e $\mathcal{S}$ uma função de selecção justa. Toda a resposta calculada de P a G via $\mathcal{S}$ é uma resposta correcta de $\overline{P}$ a G.

Prova. Por indução no comprimento da refutação-SLDNF de P e G que calcula a resposta θ. Utilizar-se-á implicitamente o Corolário 6.2.11.

Suponha-se que o comprimento da refutação é 1. Então $G \equiv \longleftarrow L$ e há dois casos a considerar.

- Se L é positivo então $L\theta$ é uma instância de um facto de P, donde $P \models \forall(L\theta)$ e portanto $\overline{P} \models \forall(L\theta)$.
- Se L é negativo então $L \equiv \neg A$ é fechado, $\theta = \varepsilon$ e existe uma árvore-SLDNF de P e $\longleftarrow A$ via $\mathcal{S}$ que é finitamente falhada; pela correcção da Negação por Falha, $\overline{P} \models \neg A \equiv \forall(L\theta)$.

Suponha-se agora que o resultado é válido para refutações-SLDNF de comprimento n e que a refutação-SLDNF de P a G com resposta calculada θ tem comprimento $n+1$. Sejam $G \equiv \longleftarrow L_1, ..., L_q$, $\Theta = \{\theta_1, ..., \theta_{n+1}\}$ a sequência de UMGs e L_m o literal seleccionado no primeiro passo. Há de novo dois casos.

- Se L_m é positivo e $C_1 \equiv A \longleftarrow L'_1, ..., L'_p$, então por hipótese de indução tem-se que $\overline{P} \models \forall((L_1 \wedge ... \wedge L_{m-1} \wedge L'_1 \wedge ... \wedge L'_p \wedge L_{m+1} \wedge ... \wedge L_q)\theta)$ e portanto $\overline{P} \models \forall((L'_1 \wedge ... \wedge L'_p)\theta)$, donde (como C_1 é consequência de $\overline{P}$) também se tem $\overline{P} \models L_m\theta$. Então $\overline{P} \models \forall((L_1 \wedge ... \wedge L_q)\theta)$ e θ é resposta correcta de $\overline{P}$ a G.

- Se L_m é negativo, um raciocínio análogo ao do caso base permite concluir que $\theta_1 = \varepsilon$ e $\overline{P} \models L_m$. Como $\mathcal{S}$ é segura, L_m é fechado, donde $\forall(L_m\theta) \equiv L_m$. Por hipótese de indução $\overline{P} \models \forall((L_1 \wedge ... \wedge L_{m-1} \wedge L_{m+1} \wedge ... \wedge L_q)\theta)$, logo mais uma vez $\overline{P} \models \forall((L_1 \wedge ... \wedge L_q)\theta)$ e θ é resposta correcta de $\overline{P}$ a G.

□

Apresentam-se agora alguns exemplos que ilustram a necessidade de só considerar funções de selecção seguras.

Exemplo 6.3.9 Considere-se o seguinte programa generalizado P.

$$p(a) \longleftarrow \neg q(x)$$
$$q(a) \longleftarrow$$

Com uma função de selecção não segura, é possível construir uma árvore-SLDNF finitamente falhada para $\longleftarrow p(a)$. Porém, é igualmente claro que $\overline{P} \not\models p(a)$.

Exemplo 6.3.10 Considere-se o seguinte programa generalizado P.

$$p(a) \longleftarrow$$
$$q(b) \longleftarrow$$

A função de selecção "sub-objectivo mais à esquerda", que não é segura, não encontra refutação-SLDNF para $\longleftarrow \neg p(x), q(x)$, já que a árvore-SLDNF para P e $\longleftarrow \neg p(x)$ não é finitamente falhada. Porém, a resposta $\{x/b\}$ é uma resposta correcta de P àquele objectivo.

Infelizmente, esta função de selecção é a que está usualmente implementada. Nalguns sistemas há o cuidado de adiar a selecção de literais negativos não instanciados para mais tarde; esses sistemas encontrariam a resposta $\{x/b\}$ no exemplo anterior. Porém, essa situação é a excepção e não a regra.

6.4 Completude da Negação por Falha

Nesta secção mostra-se que a regra da Negação por Falha é completa para o universo dos programas determinados. Um exemplo simples mostra que um resultado mais geral não é razoável.

Exemplo 6.4.1 Seja P o programa generalizado seguinte.

$$q(a) \longleftarrow \neg r(a)$$
$$r(a) \longleftarrow p(a)$$
$$r(a) \longleftarrow \neg p(a)$$
$$p(x) \longleftarrow p(f(x))$$

É fácil verificar que $P \models r(a)$, já que em qualquer estrutura de interpretação J se tem $p_J(a_J) = 1$ ou $p_J(a_J) = 0$, e portanto $P \models \neg q(a)$. Pelo Corolário 6.2.11 tem-se ainda $\overline{P} \models \neg q(a)$. No entanto não existe nenhuma árvore-SLDNF finitamente falhada para P e $\longleftarrow q(a)$.

Assim, ao longo desta secção considerar-se-ão apenas programas determinados. Em primeiro lugar, é necessário mostrar que as respostas correctas de $\overline{P}$ a G são precisamente as respostas correctas de P a G quando P é um programa determinado e G é um objectivo determinado. Uma das implicações já está estabelecida pelo Corolário 6.2.13; para mostrar a implicação inversa é necessário generalizar a noção de transformação de Herbrand associada a um programa determinado.

Definição 6.4.2 Uma *pré-interpretação* para uma assinatura de primeira ordem Σ é um par $I = \langle D, \cdot_I \rangle$ em que D é um conjunto (o *domínio* da pré-interpretação) e $\cdot_I$ um mapa de interpretação das constantes e símbolos de função em Σ.

Uma estrutura de interpretação J diz-se *baseada* em I se os domínios de I e J coincidirem e se tiver $c_I = c_J$ e $f_I = f_J$ para cada símbolo de constante c e de função f.

Proposição 6.4.3 Seja I uma pré-interpretação para uma assinatura de primeira ordem Σ.

(i) O conjunto das estruturas de interpretação baseadas em I está em bijecção com o conjunto dos subconjuntos de átomos fechados sobre D.

(ii) O conjunto das estruturas de interpretação baseadas em I com a relação de inclusão forma um reticulado completo.

Prova.

(i) Análogo à prova da Proposição 1.4.3, observando que o conjunto dos átomos fechados sobre $\mathcal{U}_\Sigma$ é precisamente $\mathcal{B}_\Sigma$.

(ii) Análogo à prova da Proposição 5.1.5.

□

Definição 6.4.4 Seja I uma pré-interpretação para um programa determinado P. A *transformação de Herbrand sobre I associada a P*, denotada por $\mathcal{T}_P^I$, é a transformação definida no reticulado das interpretações baseadas I por

$$\mathcal{T}_P^I(J) = \left\{ p([\![t_1]\!]_{J,\rho}, \ldots, [\![t_n]\!]_{J,\rho}) \;\middle|\; \begin{array}{l} p(t_1, \ldots, t_n) \longleftarrow B_1, \ldots, B_q \text{ é uma cláusula de } P \\ [\![B_1]\!]_{J,\rho} = \ldots = [\![B_q]\!]_{J,\rho} = 1 \end{array} \right\}.$$

Observe-se que a transformação de Herbrand anteriormente definida é um caso particular desta, em que I é a pré-interpretação das estruturas de Herbrand de P.

A prova do resultado seguinte é completamente análoga à da Proposição 5.2.2.

Proposição 6.4.5 Sejam P um programa determinado, I uma pré-interpretação para P e J uma estrutura de interpretação baseada em I. Então J é modelo de P sse $\mathcal{T}_P^I(J) \subseteq J$.

Através da transformação de Herbrand generalizada conseguem-se ainda caracterizar os modelos da completação de P.

Proposição 6.4.6 Sejam P um programa determinado, I uma pré-interpretação para P e J uma estrutura de interpretação baseada em I. Suponha-se ainda que $=_J$ é a identidade no domínio de I e que J é modelo da teoria da igualdade. Então J é modelo de P sse J é ponto fixo de $\mathcal{T}_P^I$.

Prova.

($\longleftarrow$) Suponha-se que J é ponto fixo de $\mathcal{T}_P^I$. Por hipótese $=_J$ é a identidade no domínio de I (e de J), pelo que J é modelo da teoria da igualdade. Resta mostrar que para cada símbolo de predicado p de P a definição completada de p é válida em J.

Suponha-se que a completação de p é $\forall(\neg p(x_1, \ldots, x_n))$. Então não há cláusulas de P cuja cabeça contenha o símbolo de predicado p, pelo que $p_J(d_1, \ldots, d_n) = 0$ para quaisquer $d_1, \ldots, d_n$ atendendo a que $J = \mathcal{T}_P^I(J)$; logo J é modelo da completação de p.

Suponha-se que a completação de p é $\forall(p(x_1, \ldots, x_n) \Leftrightarrow (E_1 \vee \ldots \vee E_q))$. Se J satisfizer E_i para algum $i = 1, \ldots, q$ e alguma atribuição ρ, então $\mathcal{T}_P^I(J)$ contém $p_J(d_1, \ldots, d_n)$, em que $d_1, \ldots, d_n$ são os valores dados a $x_1, \ldots, x_n$ por ρ (por definição de $\mathcal{T}_P^I$) e portanto $J = \mathcal{T}_P^I(J)$ é modelo de $\forall(p(x_1, \ldots, x_n) \Leftarrow (E_1 \vee \ldots \vee E_q))$. Reciprocamente, se J contiver $p_J(d_1, \ldots, d_n)$, como $J = \mathcal{T}_P^I(J)$ existe uma cláusula de P cujo corpo é satisfeito por J para alguma atribuição ρ, pelo que J e ρ satisfazem algum dos E_is (com $i = 1, \ldots, q$) e por conseguinte J é modelo de $\forall(p(x_1, \ldots, x_n) \Rightarrow (E_1 \vee \ldots \vee E_q))$.

($\longrightarrow$) Suponha-se agora que J é modelo de P. Em particular J é modelo da completação de cada símbolo de predicado p da forma $\forall(p(x_1, \ldots, x_n) \Leftrightarrow (E_1 \vee \ldots \vee E_q))$. Se J satisfizer uma instanciação do corpo duma cláusula C de P, então J é modelo de um dos E_is na definição do predicado ocorrente na cabeça de C para uma dada instanciação das variáveis que ocorrem em E_i, donde (como J é modelo da completação desse predicado) J satisfaz p com os argumentos correspondentemente instanciados – que é precisamente a cabeça de C. Conclui-se então que $\mathcal{T}_P^I(J) \subseteq J$. Reciprocamente, se $p_J(d_1, \ldots, d_n) = 1$ então como J é modelo da completação de p existirá um E_i que é satisfeito; a esse E_i corresponde um corpo duma cláusula C de P que também é satisfeito por alguma atribuição às variáveis, tendo-se ainda que a mesma atribuição aplicada à cabeça de C corresponde a $p_J(d_1, \ldots, d_n)$. Logo $J \subseteq \mathcal{T}_P^I(J)$.

□

Proposição 6.4.7 Sejam P um programa determinado e $A_1, \ldots, A_q$ átomos. Se $\overline{P} \models \forall(A_1 \wedge \ldots \wedge A_q)$ então $P \models \forall(A_1 \wedge \ldots \wedge A_q)$.

Prova. Pela Proposição 1.3.13, basta mostrar que, nas condições da hipótese, $P \cup \{\neg\forall(A_1 \wedge \ldots \wedge A_q)\}$ não tem modelos. Substituindo as variáveis livres em $A_1, \ldots, A_q$ por símbolos de constante que não ocorram em P, verifica-se facilmente que isto é ainda equivalente a mostrar que $S = P \cup \{\neg A'_1 \vee \ldots \vee \neg A'_q\}$ não tem modelos. Como S é um conjunto de cláusulas, basta mostrar que não tem modelo de Herbrand atendendo ao Corolário 1.4.6.

Seja H uma estrutura de Herbrand para S. Observe-se que H não é necessariamente uma estrutura de Herbrand para P (só o é se $A_1, \ldots, A_q$ forem fórmulas fechadas). Se H for modelo de P, então $\mathcal{T}_P^I(H) \subseteq H$, onde I é a pré-interpretação para P que se obtém de H ignorando a interpretação das constantes que não ocorrem em P. Como $\mathcal{T}_P^I$ é monótona, pelo Corolário 4.3.5 existe um ponto fixo J de $\mathcal{T}_P^I$ com $H \supseteq J$. Ora pela Proposição 6.4.6 J é modelo de $\overline{P}$, donde em particular J não é modelo de $\neg A'_1 \vee \ldots \vee \neg A'_q$. Logo H também não é modelo desta fórmula.

Por arbitrariedade de H conclui-se que S é contraditório, donde $P \models \forall(A_1 \wedge \ldots \wedge A_q)$. □

Observe-se que o resultado anterior não é válido para programas generalizados: não só a prova não se aplica por $\mathcal{T}_P^I$ não ser monótona, como é fácil arranjar contra-exemplos.

Exemplo 6.4.8 Seja P o programa generalizado seguinte.

$$p(a) \longleftarrow \neg p(a)$$

Então $\overline{P}$ é contraditório, donde em particular $\overline{P} \models \Box$, mas P tem modelos (por exemplo $\mathcal{B}_P$) e portanto $P \not\models \Box$.

Teorema 6.4.9 Sejam P um programa determinado e G um objectivo determinado. Uma substituição θ é uma resposta correcta de P a G sse θ é uma resposta correcta de $\overline{P}$ a G.

Prova. Claramente as condições sobre as variáveis no domínio de θ são equivalentes; as condições sobre consequência semântica são consequência do Corolário 6.2.13 e da Proposição 6.4.7. □

O resultado seguinte é um auxiliar relevante para a discussão na Secção 6.5.

Proposição 6.4.10 Sejam P um programa determinado e H uma estrutura de Herbrand para P. Então $H \cup \Delta_{\mathcal{U}_P}$ é modelo de $\overline{P}$ sse H é ponto fixo de $\mathcal{T}_P$.

Prova. O resultado é consequência imediata da Proposição 6.4.6 e do facto de $I \cup \Delta_{\mathcal{U}_P}$ ser um modelo da teoria da igualdade. □

Corolário 6.4.11 Sejam P um programa determinado e $A \in \mathcal{B}_P$. Então $A \notin \text{pfmax}(\mathcal{T}_P)$ sse $\overline{P} \cup \{A\}$ não tem modelo de Herbrand.

Prova.

($\longleftarrow$) Suponha-se que $A \in \text{pfmax}(\mathcal{T}_P)$. Então $\text{pfmax}(\mathcal{T}_P) \cup \Delta_{\mathcal{U}_P}$ é modelo de $\overline{P} \cup \{A\}$ pela proposição anterior.

($\longrightarrow$) Suponha-se que H é um modelo de Herbrand de $\overline{P} \cup \{A\}$. Pela Proposição 6.2.7, a interpretação da igualdade é $\Delta_{\mathcal{U}_P}$, logo H é da forma $H' \cup \Delta_{\mathcal{U}_P}$ com H' um modelo de Herbrand de P. Pela proposição anterior, H' é ponto fixo de $\mathcal{T}_P$, donde $H' \subseteq \text{pfmax}(\mathcal{T}_P)$ pelo Corolário 4.3.5 e portanto $A \in \text{pfmax}(\mathcal{T}_P)$.

□

Teorema 6.4.12 *(Completude da Negação por Falha.)* Sejam P um programa determinado, G um objectivo determinado e $\mathcal{S}$ uma função de selecção justa. Se $\overline{P} \models G$ então a árvore-SLDNF de P e $\longleftarrow G$ via $\mathcal{S}$ é finitamente falhada.

Prova. A prova é por contra-recíproco. Seja $G \equiv \longleftarrow A_1, \ldots, A_q$; a partir de um ramo não falhado duma árvore-SLDNF de P e G via $\mathcal{S}$ constrói-se um modelo de $P \cup \{\exists(A_1 \wedge \ldots \wedge A_q)\}$, o que estabelece a negação da hipótese.

Seja $\mathcal{R}$ um ramo não falhado da árvore-SLDNF de P e G via $\mathcal{S}$. Então $\mathcal{R}$ corresponde a uma derivação-SLDNF $\langle \mathcal{G}, \mathcal{C}, \Theta \rangle$ com $\mathcal{G} = \{G \equiv G_0, G_1, \ldots, G_n, \ldots\}$, $\mathcal{C} = \{C_1, \ldots, C_n, \ldots\}$ e $\Theta = \{\theta_1, \ldots, \theta_n, \ldots\}$. Observe-se que $\mathcal{R}$ pode ser finito ou infinito; a prova é independente deste facto.

Defina-se uma pré-interpretação para P como a seguir se descreve. Seja Σ a assinatura de primeira ordem subjacente a P e defina-se uma relação binária $*$ sobre os termos de Σ por $t * t'$

sse $t\theta_1 \dots \theta_k = t'\theta_1 \dots \theta_k$ para algum $k \in \mathbb{N}$. Uma vez que $*$ é uma relação de equivalência, pode-se tomar o conjunto D das classes de equivalência de $*$. Este conjunto é o domínio da pré-interpretação I. Finalmente, define-se $c_I = [c]_*$ e f_I por $f_I([t_1]_*, \dots, [t_n]_*) = [f(t_1, \dots, t_n)]_*$. Verifica-se facilmente que f_I está bem definida.

Seja agora J a estrutura de interpretação definida sobre I como se segue.

$$J = \{p([t_1]_*, \dots, [t_n]_*) \mid p(t_1, \dots, t_n) \text{ ocorre em } G_i \text{ para algum } i \in \mathbb{N}\}$$

Mostra-se de seguida que $\mathcal{T}_P^I(J) \subseteq J$. Se $p([t_1]_*, \dots, [t_n]_*) \in J$, então $p(t_1, \dots, t_n)$ ocorre em G_i para algum i. Como $\mathcal{S}$ é uma função de selecção justa, existe um objectivo G_j em que $p(t'_1, \dots, t'_n)$ é seleccionado, com $j \geq i$ e $t'_k = t_k\theta_{i+1} \dots \theta_j$. Observe-se também que $[t_k]_* = [t'_k]_*$.

Como $\mathcal{R}$ é um ramo não falhado, existe uma cláusula $C_j \equiv A \longleftarrow B_1, \dots, B_p$ de P tal que $A\theta_{j+1} \equiv p(t'_1, \dots, t'_n)\theta_{j+1}$. Por definição de J, para cada m tem-se que $[B_m\theta_{j+1}]_* \in J$ (em que $[B_m\theta_{j+1}]_*$ tem o significado sugerido pela notação); observe-se ainda que para cada k se tem $t_k\theta_{i+1} \dots \theta_{j+1} = t'_k\theta_{i+1} \dots \theta_{j+1} = t'_k\theta_{j+1}$ atendendo a que $\theta_{i+1}, \dots, \theta_j$ não afectam as variáveis de C_j. Então $p([t_1]_*, \dots, [t_n]_*) = p([t'_1\theta_{j+1}]_*, \dots, [t'_n\theta_{j+1}]_*) \in \mathcal{T}_P^I(J)$.

Seja $J' \supseteq J$ um ponto fixo de $\mathcal{T}_P^I$, que existe pelo Corolário 4.3.5. Então $J' \cup \Delta_\Sigma$ é modelo de $\overline{P} \cup \{\exists(A_1, \dots, A_q)\}$, já que J é modelo de $\exists(A_1, \dots, A_q)$ por construção, $J \subseteq J'$ e J' é modelo de $\overline{P}$ pela Proposição 6.4.6. □

Corolário 6.4.13 Sejam P um programa determinado e $A \in \mathcal{B}_P$. Se $\overline{P} \models \neg A$ então $A \in \mathcal{F}_P$.

Prova. Pela completude da Negação por Falha aplicando o Corolário 6.1.12. □

O modelo construído na prova da completude da Negação por Falha não é um modelo de Herbrand. Em geral, a existência de um tal modelo não é necessária.

Exemplo 6.4.14 Seja P o programa determinado seguinte.

$$p(f(y)) \longleftarrow p(y)$$
$$q(a) \longleftarrow p(y)$$

É fácil verificar que $q(a) \notin \mathcal{F}_P$. Por outro lado, como $\text{pfmax}(\mathcal{T}_P^I) = \emptyset$ e $q(a) \notin \emptyset$, o Corolário 6.4.11 mostra que não existem modelos de Herbrand para $\overline{P} \cup \{q(a)\}$.

Os dois teoremas seguintes sumarizam os resultados deste capítulo.

Teorema 6.4.15 Sejam P um programa determinado e $A \in \mathcal{B}_P$. As condições seguintes são equivalentes.

(i) $A \in \mathcal{F}_P$

(ii) $A \notin \mathcal{T}_P \downarrow \omega$

(iii) A está no conjunto de falhas finitas de P

(iv) A árvore-SLD(NF) de P e $\longleftarrow A$ via $\mathcal{S}$ é finitamente falhada para qualquer função de selecção justa $\mathcal{S}$.

(v) $\overline{P} \models \neg A$.

Prova. A equivalência das quatro primeiras condições foi provada no Corolário 6.1.12. A quinta condição é equivalente à primeira pelo Corolário anterior. □

Teorema 6.4.16 Sejam P um programa determinado e G um objectivo determinado. Então $\overline{P} \models G$ sse existe uma árvore-SLD(NF) finitamente falhada de P e G.

Prova. Pela completude da Negação por Falha aplicando o Corolário 6.2.11. □

6.5 Incompletude da resolução-SLDNF

Nesta secção discute-se a questão da completude da resolução-SLDNF, ilustrando-se com exemplos os problemas que surgem.

Exemplo 6.5.1 Seja P o programa determinado seguinte.

$$p(x) \longleftarrow$$
$$q(a) \longleftarrow$$
$$r(b) \longleftarrow$$

Então $\{x/b\}$ é uma resposta correcta de $\overline{P}$ a $\longleftarrow p(x), \neg q(x)$, mas nenhuma função de selecção a consegue calcular.

Exemplo 6.5.2 Seja Q o programa generalizado seguinte.

$$r(a) \longleftarrow p(a)$$
$$r(a) \longleftarrow \neg p(a)$$
$$p(x) \longleftarrow p(f(x))$$

Então ε é uma resposta correcta de $\overline{Q}$ a $\longleftarrow r(a)$, mas nenhuma função de selecção a consegue calcular.

O primeiro exemplo (programa P) mostra o inconveniente de a Negação por Falha funcionar apenas como um teste e não permitir instanciação: se se pudesse instanciar x por b ao construir a árvore-SLDNF para P e $\longleftarrow q(x)$ encontrar-se-ia a resposta correcta. Infelizmente, por razões discutidas anteriormente, tal não é aconselhável...

O segundo exemplo (programa Q) ilustra um problema bem mais profundo. Claramente qualquer modelo de Q satisfaz $r(a)$, mas não é nada claro como o raciocínio subjacente à prova deste facto pode ser implementado por meio de resolução-SLDNF.

Índice