LÓGICA DE PREDICADOS

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO

1.1. INTELIGÊNCIA ARTIFICIAL, LÓGICA SIMBÓLICA E PROVA DE TEOREMAS

Desde o nascimento do primeiro computador moderno, a tecnologia dos computadores desenvolveu-se numa velocidade fantástica. Hoje vê-se computadores sendo usados, não somente para resolver problemas de alta complexidade computacional, como realizar uma transformada rápida de Fourier ou inverter uma matriz de grandes dimensões, mas também para executar tarefas que poderiam ser chamadas de inteligentes, se feitas por seres humanos. Algumas destas tarefas são: escrita de programas, responder perguntas, provar teoremas. A Inteligência Artificial é um ramo da ciência da computação que está preocupado com a execução de tais tarefas.

[MAGF1] Comentário:

A segunda metade dos anos 60 foi fenomenal para a inteligência artificial devido ao aumento no interesse na prova automática de teoremas. A disseminação deste interesse foi causada, não somente pela crescente consciência de que a habilidade de fazer deduções lógicas é uma parte integrante da inteligência humana, mas foi, talvez, um resultado do nível alcançado pelas técnicas de prova automática de teoremas ao final dos anos 60. Os fundamentos da prova automática de teoremas foram desenvolvidos por Herbrand em 1930. Seu método era impossível de ser implementado até a invenção do computador digital. E continuou assim até a publicação do fantástico artigo de J.A. Robinson em 1965, junto com o desenvolvimento do princípio da resolução, cujos maiores passos foram dados para obter os provadores de teroremas implementados em computadores. A partir deste momento, sucessivos refinamentos tem sido feitos no princípio de resolução.

Paralelamente ao progresso no aprimoramento das técnicas de prova automática de teoremas aconteceu o progresso na aplicação das técnicas de prova automática de teoremas a vários problemas de inteligência artificial. Elas foram inicialmente aplicadas a dedução (resposta de questões) e, posteriormente para solução de problemas, síntese e análise de programas entre muitas outras aplicações.

Existem muitos pontos de vista através dos quais pode-se estudar a lógica simbólica. Tradicionalmente, ela foi estudada através de orientações filosóficas e matemáticas. Aqui se está interessado em aplicações da lógica simbólica para resolução de problemas intelectualmente difíceis. Isto é, quer-se usar lógica simbólica para representar problemas e obter suas soluções.

A seguir vão ser apresentados alguns exemplos bastante simples para demonstrar como a lógica simbólica pode ser usada para representar problemas. Mesmo que não se tenha ainda discutido formalmente lógica simbólica pode-se utilizar a intuição para compreender o que segue.

Considere os seguintes fatos:

F1: Se está quente e úmido, então choverá.
F2: Se está úmido, então está quente.
F3: Está úmido agora.

A pergunta é : Vai chover ?

Os fatos acima são escritos em português. Deve-se usar símbolos para representá-los. Faça P, Q e R representar *Está quente*, *Está Úmido* e *Choverá*, respectivamente. Também são necessários alguns símbolos lógicos. Neste caso, pode-se usar $\wedge$ para representar o *E* e $\rightarrow$ para representar *Implica Em*. Então os três fatos acima podem ser representados como:

F1: $P \wedge Q \rightarrow R$
F2: $Q \rightarrow P$
F3: Q

Traduzidas as sentenças em português para fórmulas lógicas. Pode-se observar que sempre que F1,F2 e F3 são verdadeiras , a fórmula:

F4: R ,   é verdadeira.

Portanto, pode-se dizer que F4 *é conseqüência lógica* de F1, F2 e F3. Isto é, choverá.

Considere outro exemplo, assumindo os seguintes fatos:

F1: Confúcio é um homem.
F2: Todos os homens são mortais.

Para representar F1 e F2, é necessário um novo conceito, chamado de *predicado*. Pode-se fazer P(x) e Q(x) representar *x é um homem* e *x é mortal*, respectivamente. Também usamos $(\forall x)$ para representar *para todo x*.

Portanto, os fatos acima serão representados por:

F1: P(Confúcio)
F2: $(\forall x)\ (P(x) \rightarrow Q(x))$.

Novamente, poder-se-á ver que a partir de F1 e F2 pode-se deduzir logicamente que:

F3: Q(Confúcio),   significando que Confúcio é mortal.

Nos dois exemplos acima foi necessário provar que uma fórmula *é conseqüência lógica* de outras fórmulas.

Vai-se chamar de *Teorema* uma sentença na qual uma fórmula é conseqüência lógica de outras fórmulas. A demonstração de que um teorema é verdadeiro, isto é , que uma fórmula é conseqüência lógica de outras fórmulas, será chamada de *Prova do Teorema*. O problema de prova automática de teoremas é considerar métodos automáticos para encontrar provas de teoremas.

Existem vários problemas que podem ser, convenientemente, transformados em problemas de prova de teorema. Seguem alguns deles:

1. Num sistema de *resposta a perguntas*, fatos podem ser representados por fórmulas lógicas. Então, para responder uma pergunta através dos fatos, deve-se provar que a fórmula correspondente à resposta deriva das fórmulas representando os fatos.
2. Num problema de *análise de programas*, pode-se descrever a execução de um programa por uma fórmula A, e a condição de que o programa acabará, por outra fórmula B. Então, verificar se o programa acabará é equivalente a provar que a fórmula B é conseqüência lógica da fórmula A.
3. No problema de *isomorfismo de grafos*, quer-se saber se um grafo é isomórfico a um subgrafo de um outro grafo. Este problema não é meramente um problema interessante

da matemática; mas, também é um problema prático. Por exemplo, a estrutura de um composto orgânico pode ser representada por um grafo. Portanto, testar se uma subestrutura de um composto orgânico é estrutura de um outro composto orgânico é um problema de isomorfismo de grafos. Para este problema, pode-se descrever grafos através de fórmulas. Então, o problema pode ser formulado como: provar que a fórmula que representa um grafo é conseqüência lógica da fórmula que representa outro grafo.

4. No problema de *transformação de estados*, existe uma coleção de estados e uma coleção de operadores. Quando um operador é aplicado a um estado, um novo estado é obtido. Partindo de um estado inicial, tenta-se encontrar uma seqüência de operadores que transformarão o estado inicial em um estado desejado. Neste caso, pode-se descrever os estados e as regras de transição entre eles através de fórmulas lógicas. Deste modo, a transformação do estado inicial em um estado desejado pode ser tratada como a verificação de que a fórmula representando o estado desejado é conseqüência lógica da fórmula que representa ambos os estados e as regras de transição entre eles.

Uma vez que muitos problemas podem ser formulados como problemas de prova automática de teoremas, está é uma área muito importante da ciência da inteligência artificial. Graças ao esforço de muitos pesquisadores, teve-se grande avanço no uso de computadores para provar teoremas.

1.2. BASE MATEMÁTICA

Nesta seção vai-se descrever alguns conceitos matemáticos básicos que serão usados nos próximos capítulos.

Um *conjunto* é uma coleção de elementos (membros). Um conjunto que não contém nenhum elemento é chamado de *conjunto vazio* e representado por $\emptyset$. Sejam A e B dois conjuntos. $x \in A$ é usado para representar que x é um elemento de A, ou que x pertence a A.

O conjunto A é *idêntico* ao conjunto B, representando-se por A = B, se e somente se A e B tem os mesmos elementos.

O conjunto A é um *subconjunto* de B, representando-se por $A \subseteq B$, se e somente se cada elemento de A é um elemento de B.

O conjunto A é um *subconjunto próprio* do conjunto B, representado por $A \subset B$, se e somente se $A \subseteq B$ e $A \neq B$.

A *união* de dois conjuntos A e B, representado por $A \cup B$, é o conjunto formado por todos os elementos que pertencem a A ou a B.

A *intersecção* dos dois conjuntos A e B, representada por $A \cap B$, é o conjunto constituído de todos os elementos que pertencem tanto a A quanto a B.

A *diferença* entre dois conjuntos A e B, representada por A - B, é o conjunto constituído de todos os elementos que pertencem a A e não pertencem a B.

Agora, serão definidas algumas relações e funções:

Um *par ordenado* de elementos é representado por (x,y), onde x é chamado de primeira coordenada e y é chamada de segunda coordenada.

Uma *relação* é um conjunto de pares ordenados. Por exemplo, a relação de igualdade é um conjunto de pares ordenados, onde cada um deles têm a primeira coordenada igual a segunda coordenada. O *domínio* de uma relação R é o conjunto de todas as primeiras coordenadas de elementos de R, e sua imagem é o conjunto de todas as segundas coordenadas.

Uma *função* é uma relação na qual não existem dois elementos distintos que possuam a mesma primeira coordenada. Se *f* é uma função e *x* um elemento do seu domínio, então *f(x)* representa a segunda coordenada do único elemento em *f* cuja primeira coordenada é *x*. *f(x)* é chamado *valor* de *f* em *x* e dizemos que *f* atribui o valor *f(x)* para *x*.

Ao longo deste texto, usa-se muitos símbolos convencionais. Por exemplo, > significa *maior que*, 1, *maior ou igual a*; <, *menor que*, 1, *menor ou igual a*; = , *igual a*; ≠, *diferente de*; $\triangleq$, *definido como* e assim por diante. O símbolo de igualdade = será usado para vários fins. Ele poderá ser usado para representar *é definido por*, *é idêntico a*, *é equivalente a* ou *é igual a*. Isto não trará nenhuma confusão, pois o significado exato será sempre descrito no texto.

CAPÍTULO 2 - A LÓGICA PROPOSICIONAL

2.1. INTRODUÇÃO

Lógica simbólica considera as linguagens cujo principal propósito é simbolizar o raciocínio encontrado não somente na matemática, mas também na vida diária. Neste capítulo, vai-se, inicialmente, estudar a lógica simbólica mais simples - a lógica proposicional (ou cálculo proposicional). No próximo capítulo, vai-se tratar algo mais geral - a lógica de predicados (ou o cálculo de predicados de primeira ordem ou lógica de predicados).

Na lógica proposicional, se está interessado nas sentenças declarativas que podem ser tanto *verdadeiras* quanto *falsas*, mas não ambas. Uma sentença declarativa qualquer é chamada de proposição. Mais formalmente, uma *proposição* é uma sentença declarativa que é ou verdadeira ou falsa. Exemplos de proposições são: *A neve é branca*, *O açúcar é um hidrocarbono*, *Smith possui o grau de Ph.D.*. O *verdadeiro* ou *falso* atribuido a proposição é chamado de *valor* da proposição. Normalmente, representa-se *verdadeiro* por V e *falso* por F. Por conveniência, vai-se usar letras maiúsculas ou conjunto de letras para representar uma proposição. Portanto, pode-se representar as proposições acima como segue:

P $\triangleq$ A neve é branca,

Q $\triangleq$ O açúcar é um hidrocarbono,

R $\triangleq$ Smith possui o grau de Ph.D.

Os símbolos, como P, Q e R, que são usados para representar proposições são chamados de *fórmulas atômicas* ou *átomos*.

A partir das proposições pode-se construir proposições *compostas* usando os *conetivos lógicos*. Exemplos de proposições compostas são: "A neve é branca *e* o céu está claro" *e* "*Se* João não está em casa, *então* Maria está em casa". Os conetivos lógicos nas duas proposições compostas acima são "*e*" e "*se*...*então*". Na lógica proposicional, usa-se cinco conetivos lógicos: ~ (não), 2 (e), 2 (ou), 1 (se...então), e 1 (se e somente se). Estes cinco conetivos lógicos podem ser usados para construir proposições compostas a partir de proposições. Generalizando, eles podem ser usados para construir proposições compostas mais complexas a partir de outras proposições compostas aplicando-os repetidamente. Por exemplo, vai-se representar "A humidade está elevada" por *P*, "A temperatura esta elevada" por *Q*, e "Alguém se sente confortável" por *C*. Então a sentença "Se a humidade e a temperatura estão elevadas, então alguém não se sentirá confortável" pode ser representada por ((P 2 Q) 1 (~C)). Portanto, vê-se que uma proposição composta pode expressar uma idéia bastante complexa. Na lógica proposicional, uma expressão que representa uma proposição, tal como *P*, ou uma proposição composta, tal como ((P 2 Q) 1 (~C)), é chamada de fórmula.

Definição: *Fórmulas* na lógica proposicional são definidas recursivamente como o seguinte:

1. Um átomo é uma fórmula.
2. Se *G* é uma fórmula, então (~*G*) é uma fórmula.
3. Se *G* e *H* são fórmulas, então (*G* 2 *H*), (*G* 2 *H*), (*G* 1 *H*), e (*G* 1 *H*) são fórmulas.
4. Todas as fórmulas são geradas pela aplicação das regras acima.

Não é difícil ver que expressões tal como (P 1) e (P 2) não são fórmulas. Ao longo das explicações, parênteses serão suprimidos desde que isso não cause nenhuma confusão. Por exemplo, P 2 Q e P 1 Q são as fórmulas (P 2 Q) e (P 1 Q), respectivamente. Pode-se, então, omitir o uso dos parênteses. Então, P 1 Q 2 R representa (P 1 (Q 2 R)), e P 1 Q 2~R 2 S representa (P 1 (Q 2 ((~R 2 S))).

Sejam *G* e *H* duas fórmulas. Então os valores das fórmulas (~G), (G 2 H), (G 2 H). (G 1 H), e (G 1 H) são obtidos a partir dos valores de G e H pelo seguinte caminho:

1. ~G é verdadeiro quando G for falso, e é falso quando G for verdadeiro. ~G é chamado de negação de G.
2. (G 2 H) é verdadeiro se G e H são ambos verdadeiros; do contrário, (G 2 H) é falso. (G 2 H) é chamado de *conjunção* de G e H.
3. (G 2 H) é verdadeiro se pelo menos um de G e H for verdadeiro; do contrário (G 2 H) é falso. (G 2 H) é chamado de *disjunção* de G e H.
4. (G 1 H) é falso se G é verdadeiro e H é falso; do contrário, (G 1 H) é verdadeiro. (G 1 H) é lido como "*Se G, então H*" ou "*G implica em H*".
5. (G 1 H) é verdadeiro quando G e H tem o mesmo valor lógico; de outro modo (G 1 H) é falso.

As relações acima podem ser vistas adequadamente na tabela 2.1. Baseando-se nesta tabela, pode-se avaliar os valores lógicos de uma fórmula avaliando os valores dos átomos que compõem a fórmula.

G	H	~G	(G 2 H)	(G 2 H)	(G 1 H)	(G 1 H)
V	V	F	V	V	V	V
V	F	F	F	V	F	F
F	V	V	F	V	V	F
F	F	V	F	F	V	V

Tabela 2.1 - Valores Lógicos das Fórmulas

2.2 INTERPRETAÇÕES DE FÓRMULAS NA LÓGICA PROPOSICIONAL

Suponha que P e Q são dois átomos e que os valores verdade de P e Q são **V** e **F**, respectivamente. Então, de acordo com a segunda linha da tabela 2.1 com P e Q substituídos por G e H, respectivamente, notamos que os valores de (~P), (P2Q), (P2 Q), (P1Q), e (P1Q) são **F**, **F**, **V**, **F** e **F**, respectivamente. Igualmente, os valores de qualquer fórmula podem ser avaliados em termos dos valores dos átomos.

Exemplo 2.1: Considere a fórmula: G $\triangleq$ ((P2Q) 1 (R 1 (~S))

Os átomos nesta fórmula são P, Q, R e S. Suponha que os valores de P,Q, R, e S sejam **V**, **F**, **V** e **V**, respectivamente. Então (P2Q) é **F** uma vez que Q é falso; (~S) é **F** uma vez que S é **V**; (R 1 (~S)) é **F** uma vez que R é **V** e (~S) é **F**; ((P2Q)1(R 1(~S)) é **V** uma vez que (P2Q) é **F** e (R 1 (~S)) é **F**. Portanto, a fórmula G é **V** se P,Q, R, e S sejam **V**, **F**, **V** e **V**, respectivamente.

O conjunto de valores {**V**, **F**, **V**, **V**} para {P, Q, R, S}, respectivamente, será chamado de uma *interpretação* da fórmula G. Uma vez que cada P, Q, R e S pode assumir um valor V ou F, existem 2^4 = 16 interpretações da fórmula G. Na Tabela 2.2, temos os valores da fórmula G sob todas as suas 16 interpretações.

P	Q	R	S	~ S	(P 2 Q)	(R 1 (~ S))	(P2Q)1(R1 (~S))
V	V	V	V	F	V	F	F
V	V	V	F	V	V	V	V
V	V	F	V	F	V	V	V
V	V	F	F	V	V	F	F
V	F	V	V	F	F	F	V
V	F	V	F	V	F	V	V
V	F	F	V	F	F	V	V
V	F	F	F	V	F	F	V
F	V	V	V	F	F	F	V
F	V	V	F	V	F	T	V
F	V	F	V	F	F	T	V
F	V	F	F	V	F	F	V
F	F	V	V	F	F	F	V
F	F	V	F	V	F	V	V
F	F	F	V	F	F	V	V
F	F	F	F	V	F	F	V

Tabela 2.2 - Tabela Verdade de $G \triangleq$ (P2Q) 1 (R 1 (~S))

Uma tabela , tal como a tabela 2.2, que mostra os valores da Fórmula G para todas as possíveis combinações de valores dos átomos pertencentes a G é chamada *Tabela Verdade de G*.

Agora dar-se-á uma definição formal de uma interpretação de uma fórmula proposicional.

Definição: Dada a fórmula proposicional G, seja A_1, A_2, ..., A_n os átomos que compõem a fórmula G. Então uma *interpretação* de G é uma seqüência de valores para A_1, ..., A_n no qual cada A_i possui um valor **V** ou **F**, mas não ambos.

Definição: Uma fórmula G é dita ser *verdadeira sob (ou em) uma interpretação* se e somente se G é avaliada **V** (verdadeira) na interpretação; do contrário, G é dita ser *falsa sob a interpretação.*

Se existirem *n* átomos distintos em uma fórmula, então existirão 2^n interpretações distintas para essa fórmula. Algumas vezes, se A_1, ..., A_n são átomos que compõem uma fórmula, pode ser mais conveniente representar uma interpretação por um conjunto {m_1, m_2, ..., m_n }, onde m_i é A_i ou ~A_i . Por exemplo, o conjunto {P, ~Q, ~R, S} representam uma interpretação na qual P, Q, R e S são **V**, **F**, **F** e **V**, respectivamente. Isto é, se um átomo A está em um conjunto que representa uma interpretação, então A possui valor V; enquanto que se a negação do átomo A está no conjunto, então A recebe o valor F. Essa convenção será adota daqui para frente.

2.3. VALIDADE E INCONSISTÊNCIA NA LÓGICA PROPOSICIONAL

Nesta seção, vai-se considerar fórmulas que são verdadeiras sob todas as suas interpretações possíveis e fórmulas que são falsas sob todas as suas interpretações possíveis.

Exemplo 2.2: Considere a fórmula: $G \triangleq$ ((P1Q) 2 P) 1 Q .

Os átomos nesta fórmula são P e Q. Portanto, a fórmula G tem $2^2 = 4$ interpretações. Os valores verdade de G sob todas as suas 4 interpretações são dados na tabela 2.3. Note que a fórmula G é verdadeira sob todas as suas interpretações. Esta fórmula será chamada de uma fórmula ***válida*** (ou ***tautologia***).

P	**Q**	**(P1Q)**	**(P 1 Q) 2 P**	**((P1Q) 2 P) 1 Q**
V	V	V	V	V
V	F	F	F	V
F	V	V	F	V
F	F	V	F	V

Tabela 2.3 - Tabela Verdade de ((P1Q) 2 P) 1 Q

Exemplo 2.3: Considere a fórmula: G $\triangleq$ (P 1 Q) 2 (P 2 ~Q).

A tabela verdade de G é dada na tabela 2.4. Note que G é falso sob todas as suas interpretações. Esta fórmula será chamada de uma fórmula *inconsistente* (ou uma *contradição*).

P	**Q**	**~Q**	**(P1 Q)**	**(P 2 ~Q)**	**(P 1 Q) 2 (P 2 ~Q)**
V	V	F	V	F	F
V	F	V	F	V	F
F	V	F	V	F	F
F	F	V	V	F	F

Tabela 2.4 - Tabela Verdade de (P 1 Q) 2 (P 2 ~Q).

Agora ver-se-á a definição formal de validade e inconsistência.

Definição: Uma fórmula é dita *válida* se e somente se ela for verdadeira sob todas as suas interpretações. Uma fórmula é dita *inválida* se e somente se ela não é válida.

Definição: Uma fórmula é dita ser *inconsistente* se e somente se ela for falsa sob todas as suas interpretações. Uma fórmula é dita ser *consistente* se e somente se ela não for inconsistente.

Pelas definições acima, são óbvias as seguintes observações:

1. Uma fórmula é válida se e somente se sua negação é inconsistente.
2. Uma fórmula é inconsistente se e somente se sua negação é válida.
3. Uma fórmula é inválida se e somente se existe, no mínimo, uma interpretação sob a qual a fórmula é falsa.
4. Uma fórmula é consistente se e somente se existe no mínimo uma interpretação sob a qual a fórmula é verdadeira.
5. Se uma fórmula é válida, então é consistente, mas não vice-versa.
6. Se uma fórmula é inconsistente, então ela é inválida, mas não vice-versa.

Exemplo 2.4: Usando o método das tabelas verdade, é possível estabelecer o seguinte:

a. (P 2 ~P) é inconsistente; portanto também é inválida.

b. (P 2 ~P) é válida; portanto também é consistente.

c. (P 1 ~P) é inválida, mas ainda consistente.

Se uma fórmula F é verdadeira sob uma interpretação I, então diz-se que I *satisfaz* F, ou F *é satisfeita* por I. Por outro lado, se uma fórmula F é falsa sob uma interpretação I, então diz-se que I *falsifica* F ou F *é falsificada* por I. Por exemplo, a fórmula (P 2 (~Q)) é satisfeita pela interpretação {P, ~Q}, mas é falsificada pela interpretação {P, Q}. Quando uma interpretação I satisfaz uma fórmula F, I também é chamado de modelo de F.

Será mostrado, posteriormente, que a prova de validade ou inconsistência de uma fórmula é um problema muito importante. Na Lógica Proposicional, uma vez que o número de interpretações de uma fórmula é finito, uma delas pode decidir se uma fórmula é ou não válida (ou inconsistente) testando-se exaustivamente todas as suas possíveis interpretações.

2.4 FORMAS NORMAIS NA LÓGICA PROPOSICIONAL

Como ficará claro mais a frente, muitas vezes é necessário transformar uma fórmula de uma forma para outra, especialmente para a "forma normal". Isto é feito pela substituição da fórmula dada por outra "equivalente" a ela e repetindo-se este processo até que a forma desejada seja obtida. Por "equivalente" entende-se o seguinte:

Definição: Duas fórmulas F e G são *equivalentes* (ou F é *equivalente* a G), representando-se por F = G, se e somente se, os valores verdade de F e G são os mesmos sob todas as interpretações de F e G.

Exemplo 2.5: Pode-se verificar que (P 1 Q) é equivalente a (~P 2 Q) examinando a tabela verdade (tabela 2.5).

P	**Q**	**(P1Q)**	**(~P 2 Q)**
V	V	V	V
V	F	F	F
F	V	V	V
F	F	V	V

Tabela 2.5- Tabela Verdade de (P 1 Q) e (~P 2 Q).

Serão necessários alguns recursos para se transformar uma fórmula em outra equivalente. Na lógica proposicional, o símbolo 🗁 representa uma fórmula que sempre é verdadeira e € uma fórmula que sempre é falsa. Então tem-se alguns pares úteis de fórmulas equivalentes, mostrados na tabela 2.6, onde F, G e H são fórmulas. Para simplificar, chamaremos cada uma delas de uma *lei*.

As leis na tabela 2.6 podem ser verificadas pelo uso das tabelas verdade. As leis (2.3a), (2.3b) são, geralmente, chamadas de leis comutativas; as leis (2.4a), (2.4b) de leis associativas; as leis (2.5a), (2.5b) de leis distributivas; e as leis (2.10a), (2.10b) de leis de De Morgan.

(2.1)	**F1G = (F1G) 2 (G1F)**	
(2.2)	**F1G = ~F 2 G**	
(2.3)	**(a) F 2 G = G 2 F;**	**(b) F 2 G = G 2 F (leis comutativas)**
(2.4)	**(a) (F 2 G) 2 H = F 2 (G 2 H);**	**(b) (F 2 G) 2 H = F 2 (G 2 H) (leis associativas)**
(2.5)	**(a) F 2 (G 2 H) = (F 2 G) 2 (F 2 H);**	**(b) F 2 (G 2 H) = (F 2 G) 2 (F 2 H) (leis distributivas)**
(2.6)	**(a) F 2 € = F;**	**(b) F 2 🗁 = F**
(2.7)	**(a) F 2 🗁 = 🗁;**	**(b) F 2 € = €**
(2.8)	**(a) F2 ~F = 🗁;**	**(b) F 2 ~F = €**
(2.9)	**~(~F) = F**	
(2.10)	**(a) ~(F 2 G) = ~F 2 ~G;**	**(b) ~(F 2 G) = ~F 2 ~G (Leis de De Morgan)**

Tabela 2.6 - Leis para Formação de Fórmulas Equivalentes

Devido às leis associativas, os parênteses em (F2G)2H ou F2(G2H) podem ser suprimidos. Isto é, pode-se escrever F2G2H para (F2G)2H e F2(G2H). Geralmente, pode-se escrever $F_1$2$F_2$2...2F_n sem nenhuma ambigüidade, onde F_1, F_2, ... , F_n são fórmulas. $F_1$2$F_2$2...2F_n é verdadeira sempre que pelo menos uma das fórmulas F_i , 1 1 i 1 n , for verdadeira, do contrário ela será falsa. $F_1$2$F_2$2...2F_n é chamada de *disjunção* de F_1,...,F_n.

Similarmente, pode-se escrever $F_1 2 F_2 2 ... 2 F_n$, a qual é verdadeira se todas as fórmulas F_1 , ... , F_n são verdadeiras, caso contrário, é falsa. $F_1 2 F_2 2 ... 2 F_n$ é chamada de *conjunção* de $F_1,...,F_n$.

Note que a ordem que F_i aparece na disjunção ou conjunção não importa.

Por exemplo, $F_1 2 F_2 2 F_3 = F_1 2 F_3 2 F_2 = F_2 2 F_1 2 F_3 = F_3 2 F_1 2 F_2 = F_2 2 F_3 2 F_1 = F_3 2 F_1 2 F_2$

Define-se forma normal como segue:

Definição: Um *literal* é um átomo ou a negação de um átomo.

Definição: A fórmula F é dita estar na *forma normal conjuntiva* se e somente se F tem a forma: $F \triangleq F_1 2 F_2 2 ... 2 F_n$, n 1 1, onde cada $F_1,...,F_n$ é uma disjunção de literais.

Exemplo 2.6: Sejam P, Q e R átomos. Então $F \triangleq$ (P 2 ~ Q 2 R) 2 (~P 2 Q) é uma fórmula na forma normal conjuntiva. Para essa fórmula, F_1 = (P 2 ~ Q 2 R) e F_2 = (~P 2 Q) . Claramente, F_1 é uma disjunção dos literais P, ~Q, e R e F_2 é uma disjunção dos literais ~P e Q.

Definição: A fórmula F é dita estar na *forma normal disjuntiva* se e somente se tem a forma: $F \triangleq F_1 2 F_2 2 ... 2 F_n$, n 1 1, onde cada $F_1,...,F_n$ é uma conjunção de literais.

Exemplo 2.7: Sejam P, Q e R átomos. Então $F \triangleq$ (~P 2 Q) 2 (P 2 ~Q 2 ~R) é uma fórmula na forma normal disjuntiva. Para esta fórmula,F_1 = (~P 2 Q) e F_2 = (P 2 ~Q 2 ~R). Claramente ve-se que F_1 é uma conjunção dos literais ~P e Q , e F_2 é uma conjunção dos literais P, ~Q e ~R.

Qualquer fórmula pode ser transformada em uma forma normal. Isto é obtido facilmente usando-se as leis dadas na tabela 2.6. A seguir mostramos um resumo do procedimento de transformação.

***Passo 1* :** Use as leis

(2.1) F1G = (F1G) 2 (G1F)

(2.2) F1G = ~F 2 G

para eliminar os conetivos lógicos $\leftrightarrow$ e $\rightarrow$.

***Passo 2* :** Repetidamente use a lei

(2.9) ~(~F) = F

e as leis de De Morgan

(2.10a) ~(F 2 G) = ~F 2 ~G

(2.10b) ~(F 2 G) = ~F 2 ~G

para deslocar o sinal de negação para imediatamente antes dos átomos.

***Passo 3* :** Repetidamente use as leis distributivas

(2.5a) F 2 (G 2 H) = (F 2 G) 2 (F 2 H)

(2.5b) F 2 (G 2 H) = (F 2 G) 2 (F 2 H)

e as outras leis para obter a forma normal.

Exemplo 2.8: Obtenha a forma normal disjuntiva para a fórmula (P 2 ~Q) 1 R.

Sendo (P 2 ~Q) 1 R= ~(P 2 ~Q) 2 R por (2.2)
= (~P 2 ~(~Q)) 2 R por (2.10a)
= (~P 2 Q) 2 R por (2.9).

a forma normal disjuntiva de (P 2 ~Q) 1 R é (~P 2 Q) 2 R.

Exemplo 2.9: Obtenha a forma normal conjuntiva para a fórmula (P 2 (Q 1 R)) 1 S.
Sendo (P 2 (Q 1 R)) 1 S=

= (P 2 (~Q 2 R)) 1 S	por (2.2)
= ~(P 2 (~Q 2 R)) 2 S	por (2.2)
= (~P 2 ~(~Q 2 R)) 2 S	por (2.10b)
= (~P 2 (~(~Q) 2 ~R)) 2 S	por (2.10a)
= (~P 2 (Q 2 ~R)) 2 S	por (2.9)
= ((~P 2 Q) 2 (~P 2 ~R)) 2 S	por (2.5a)
= S 2 ((~P 2 Q) 2 (~P 2 ~R))	por (2.3a)
= (S 2 (~P 2 Q)) 2 (S 2 (~P 2 ~R))	por (2.5a)
= (S 2 ~P 2 Q) 2 (S 2 ~P 2 ~R)	por (2.4a).

A forma normal conjuntiva de (P 2 (Q 1 R)) 1 S é (S 2 ~P 2 Q) 2 (S 2 ~P 2 ~R).

2.5. IMPLICAÇÃO LÓGICA

Tanto na matemática como na vida diária, geralmente é necessário decidir se uma afirmação segue de outras afirmações. Isto leva ao conceito de *Implicação Lógica*. Antes da definição formal, vai-se ver um exemplo do que se entende por implicação lógica.

Exemplo 2.10: Suponha que os preços de estocagem diminuem se a taxa de juros internacional aumenta. Suponha, também, que a maioria das pessoas ficam infelizes quando os preços de estocagem diminuem. Assuma que a taxa de juros internacional aumenta. Mostre que se pode concluir que a maioria das pessoas estão infelizes.

Para mostrar a conclusão acima, colocaremos as sentenças da seguinte forma:

$P \triangleq$ Taxa de juros internacional aumenta.
$S \triangleq$ Preços de estocagem diminuem.
$U \triangleq$ A maioria das pessoas estão infelizes.

Há quatro sentenças neste exemplo, que são:
(1) Se a taxa de juros internacionais aumenta, os preços de estocagem diminuem.
(2) Se os preços de estocagem diminuem, a maioria das pessoas ficam infelizes.
(3) A taxa de juros internacional aumenta.
(4) A maioria das pessoas estão infelizes.

Essas sentenças podem ser simbolizadas como:
(1) $P \rightarrow S$ (2) $S \rightarrow U$ (3) P (4) U

Vai-se mostrar que (4) é verdadeiro sempre que (1) $\wedge$ (2) $\wedge$ (3) é verdadeiro. Através de simbologia própria tem-se que $((P \rightarrow S) \wedge (S \rightarrow U) \wedge P)$ representa a sentença anterior. Transformando para a forma normal conjuntiva através da aplicação de diversas leis de transformação apresentadas anteriormente obtém-se:

$((P \rightarrow S) \wedge (S \rightarrow U) \wedge P)$	
$((\sim P \vee S) \wedge (\sim S \vee U) \wedge P)$	Por 2.2
$(P \wedge (\sim P \vee S) \wedge (\sim S \vee U))$	Por 2.3b (Lei comutativa)
$(((P \wedge \sim P) \vee (P \wedge S)) \wedge (\sim S \vee U))$	Por 2.5b (Lei distributiva)
$((\Box \vee (P \wedge S)) \wedge (\sim S \vee U))$	Por 2.8b (Lei do inverso)
$(P \wedge S) \wedge (\sim S \vee U)$	Por 2.6a (Lei da identidade)
$(P \wedge S \wedge \sim S) \vee (P \wedge S \wedge U)$	Por 2.5b (Lei distributiva)
$(P \wedge \Box) \vee (P \wedge S \wedge U)$	Por 2.8b (Lei do inverso)
$\Box \vee (P \wedge S \wedge U)$	Por 2.7b (Lei do elemento nulo)

$P \wedge S \wedge U$ Por 2.6a (Lei da identidade)

Portanto, se $((P \rightarrow S) \wedge (S \rightarrow U) \wedge P)$ é verdadeiro, como exposto acima, então $(P \wedge S \wedge U)$ é verdadeiro, pois são equivalentes. Uma vez que, $(P \wedge S \wedge U)$ é verdadeiro somente se P, S e U são todos verdadeiros, concluí-se que U é verdadeiro.

Devido ao fato de U ser verdadeiro quando $(P \rightarrow S)$, $(S \rightarrow U)$ e P são verdadeiros, em lógica, U é chamado de **implicação lógica** de $(P \rightarrow S)$, $(S \rightarrow U)$ e P. Mais formalmente, pode-se definir implicação lógica como segue.

Definição: Dadas as fórmulas F1, F2, F3, ..., Fn e a fórmula G, diz-se que F1,F2,...,Fn **implica logicamente** ou apenas **implica** na fórmula G, se G é verdadeiro todas as vezes que $F1 \wedge F2 \wedge F3 \wedge ... \wedge Fn$ for verdadeiro.

Indica-se da seguinte forma: $F1 \wedge F2 \wedge F3 \wedge ... \wedge Fn \rightarrow G$

Exemplo 2.11: As tabelas verdade das fórmulas $P \wedge Q$, $P \vee Q$ e P 1 Q são:

P	**Q**	**$P \wedge Q$**	**$P \vee Q$**	**P 1 Q**	
V	V	**V**	**V**	**V**	(1)
V	F	F	V	F	(2)
F	V	F	V	F	(3)
F	F	F	F	V	(4)

A fórmula $P \wedge Q$ é verdadeira (**V**) somente na linha 1 e, nesta linha, as fórmulas $P \vee Q$ e P 1 Q também são verdadeiras (V). Logo, a primeira fórmula **implica** cada uma das outras duas fórmulas, isto é:

$P \wedge Q \rightarrow P \vee Q$ e $P \wedge Q \rightarrow$ P 1 Q

Teorema 2.1 (Teorema da Dedução): Dadas as fórmulas F1,...,Fn e a fórmula G, F1,...,Fn implica logicamente em G **se e somente se** a *fórmula condicional* $((F1 \wedge F2 \wedge F3 \wedge ... \wedge Fn) \rightarrow G)$ *é válida* (*tautologia*).

Prova: (IDA) Suponha que G é implicação lógica de F1,...,Fn. Seja I uma interpretação arbitrária. Se F1,...,Fn são verdadeiros em I, então, pela definição de implicação lógica, G é verdadeiro em I. Portanto, $((F1 \wedge ... \wedge Fn) \rightarrow G)$ é verdadeiro em I. Por outro lado, se F1,..., Fn são falsos em I, então a condicional $((F1 \wedge ... \wedge Fn) \rightarrow G)$ é verdadeira em I. Dessa forma, $((F1 \wedge ... \wedge Fn) \rightarrow G)$ é verdadeira sob qualquer interpretação. Então a condicional $((F1 \wedge ... \wedge Fn) \rightarrow G)$ é uma fórmula válida (tautológica).

(VOLTA) Suponha que a condicional $((F1 \wedge ... \wedge Fn) \rightarrow G)$ é uma fórmula válida. Para alguma interpretação I, se $(F1 \wedge ... \wedge Fn)$ é verdadeira em I, então G precisa ser verdade em I. Portanto, G é implicação lógica de F1, ... , Fn.

Portanto, *a toda implicação lógica corresponde uma condicional válida (tautológica).*

Exemplo 2.12: A condicional $P \wedge \sim P \rightarrow Q$ é válida, pois a última coluna de sua tabela verdade encerra somente a letra V.

P	**Q**	**~P**	**$P \wedge \sim P$**	**$P \wedge \sim P \rightarrow Q$**
V	V	F	F	**V**
V	F	F	F	**V**
F	V	V	F	**V**
F	F	V	F	**V**

Logo, subsiste a implicação lógica $P \wedge \sim P \rightarrow Q$.

Teorema 2.2: Dadas as fórmulas F1,..., Fn e a fórmula G, G é uma implicação lógica de F1,..., Fn **se e somente se** a *fórmula* (F1 ∧ F2 ∧ ... ∧ Fn ∧ ~G) *é inconsistente (contradição).*
Prova: Pelo teorema 2.1, anteriormente demonstrado, G é uma implicação lógica de F1, ..., Fn se e somente se a fórmula ((F1 ∧ ... ∧ Fn) → G) é válida. Portanto, G é uma implicação lógica de F1, ..., Fn se e somente se a negação de ((F1 ∧ ... ∧ Fn) → G) é inconsistente (contraditória). Uma vez que,

~((F1 ∧ ... ∧ Fn) → G)	
~(~(F1 ∧ ... ∧ Fn) ∨ G)	Por 2.2
(~(~(F1 ∧ ... ∧ Fn)) ∧ ~G)	Por 2.10 (Lei de De Morgan)
(F1 ∧ ... ∧ Fn) ∧ ~G	Por 2.9 (Lei da dupla negação)
F1 ∧ ... ∧ Fn ∧ ~G,	

conclui-se que o teorema 2.2 é verdadeiro.

Os teoremas 2.1 e 2.2 são muito importantes. Eles mostram que provando que uma fórmula particular é uma implicação lógica de um conjunto finito de fórmulas é o equivalente a provar que uma certa fórmula relacionada é válida (tautológica) ou inconsistente (contraditória).

Exemplo 2.13: Considere as fórmulas F1 ≜ (P → Q), F2 ≜ ~Q e G ≜ ~P. Mostre que G é uma implicação lógica de F1 e F2, ou seja, ((P → Q) ∧ ~Q) → ~P.
Resposta: Para mostrar a implicação lógica acima pode-se usar três métodos: o primeiro é fazer a prova através de tabela verdade usando a definição, o segundo e o terceiro é usar, respectivamente, os teoremas 2.1 e 2.2 onde se quer provar se as respectivas fórmulas são válidas ou inconsistentes. Para tanto, desenvolve-se cada teorema na forma normal conjuntiva ou na forma normal disjuntiva. A escolha da fórmula equivalente é arbitrária, mas pode-se diminuir a quantidade de cálculos usando a forma normal disjuntiva para provar a validade (Teorema 2.1) e a forma normal conjuntiva para provar a inconsistência (Teorema 2.2). Isso é explicado pela Lei 2.7 (Lei do Elemento Nulo). Para se chegar à validade em uma expressão na forma normal disjuntiva é preciso apenas que durante o desenvolvimento apenas um membro seja verdadeiro:

F1 ∨ F2 ∨ F3 ∨ 🗁 ∨ F4 ∨ ... ∨ Fn = 🗁

No caso de se querer provar a inconsistência de uma expressão na forma normal conjuntiva, é necessário provar apenas que um membro seja falso para tornar toda a expressão falsa:

F1 ∧ F2 ∧ F3 ∧ € ∧ F4 ∧ ... ∧ Fn = €

***Método 1*:** Pode-se usar a técnica da tabela verdade para mostrar que G é verdade em cada modelo (interpretação) de (P → Q) ∧ ~Q. Portanto,

P	**Q**	**P → Q**	**~Q**	**(P → Q) ∧ ~Q**	**~P**	
V	V	V	F	F	F	(1)
V	F	F	V	F	F	(2)
F	V	V	F	F	V	(3)
F	F	V	V	**V**	**V**	(4)

Da tabela acima, pode-se notar que há somente uma interpretação (4) onde (P → Q) ∧ ~Q é verdadeiro. Como se vê nessa interpretação (4) ~P é também verdadeiro. Assim, pela definição de implicação lógica, concluimos que (P → Q) ∧ ~Q implicam logicamente em ~P.

Método 2**:** Aqui usa-se o teorema 2.1. Este pode ser provado extendendo a tabela verdade anterior ou desenvolvendo-se a fórmula ((P → Q) ∧ ~Q) → ~P.

a) A tabela a seguir mostra que ((P → Q) ∧ ~Q) → ~P é verdade (**V**) em todas as interpretações. Portanto ((P → Q) ∧ ~Q) → ~P é válida (tautologia) e, conforme o teorema 2.1, ~P é uma implicação lógica de (P → Q) e ~Q.

P	***Q***	***P → Q***	***~Q***	***(P → Q) ∧ ~Q***	***~P***	***((P→Q∧~Q)→~P***
V	V	V	F	F	F	**V**
V	F	F	V	F	F	**V**
F	V	V	F	F	V	**V**
F	F	V	V	V	V	**V**

b) Pode-se, também, provar a validade de ((P → Q) ∧ ~Q) → ~P pela sua transformação em formas normais conjuntivas ou disjuntivas. Como se quer provar uma validade, torna-se mais conveniente usar a forma normal disjuntiva.

~((P → Q) ∧ ~Q) ∨ ~P	Por 2.2
~((~P ∨ Q) ∧ ~Q) ∨ ~P	Por 2.2
~((~P ∧ ~Q) ∨ (Q ∧ ~Q)) ∨ ~P	Por 2.5b (Lei distributiva)
~((~P ∧ ~Q) ∨ €) ∨ ~P	Por 2.8b (Lei do inverso)
~((~P ∧ ~Q)) ∨ ~P	Por 2.6a (Lei da identidade)
(P ∨ Q) ∨ ~P	Por 2.10b (Lei de De Morgan)
(Q ∨ P) ∨ ~P	Por 2.3a (Lei comutativa)
Q ∨ (P ∨ ~P)	Por 2.4a (Lei associativa)
Q ∨ 🗁	Por 2.8a (Lei do inverso)
🗁	Por 2.7a (Lei do elemento nulo)

Deste modo, ((P → Q) ∧ ~Q) → ~P é válida.

Método 3**:** Aqui vai-se usar o teorema 2.2. Neste caso, quer-se provar que ~(((P → Q) ∧ ~Q) → ~P) = ((P → Q) ∧ ~Q) ∧ (~(~P)) = (P → Q) ∧ ~Q ∧ P é inconsistente.

a) Novamente, como no método 2, pode-se usar a técnica da tabela verdade para mostrar que ((P → Q) ∧ ~Q) → ~P é falso em cada interpretação.

P	**Q**	**P → Q**	**~Q**	**(P → Q) ∧ ~Q ∧ P**	
V	V	V	F	**F**	(1)
V	F	F	V	**F**	(2)
F	V	V	F	**F**	(3)
F	F	V	V	**F**	(4)

Da tabela acima concluí-se que (P → Q) ∧ ~Q ∧ P é inconsistente e, de acordo com o teorema 2.2, ~P é implicação lógica de (P → Q) e ~Q .

b) Pode-se, também, provar a inconsistência de (P → Q) ∧ ~Q ∧ P através de sua transformação em uma forma normal conjuntiva.

(P → Q) ∧ ~Q ∧ P	
(~P ∨ Q) ∧ ~Q ∧ P	Por 2.2
((~P ∧ ~Q) ∨ (Q ∧ ~Q)) ∧ P	Por 2.5b (Lei distributiva)
((~P ∧ ~Q) ∨ €) ∧ P	Por 2.8b (Lei do inverso)
((~P ∧ ~Q) ∧ P	Por 2.6a (Lei da identidade)

$((\sim Q \wedge \sim P) \wedge P$	Por 2.3b (Lei comutativa)
$\sim Q \wedge (\sim P \wedge P)$	Por 2.4b (Lei associativa)
$\sim Q \wedge €$	Por 2.8b (Lei do inverso)
€	Por 2.7b (Lei do elemento nulo)

Dessa forma, $(P \rightarrow Q) \wedge \sim Q \wedge P$ é inconsistente e portanto fica mostrada a implicação lógica.

2.6 APLICAÇÕES DA LÓGICA PROPOSICIONAL

Após discutir vários aspectos e conceitos nas seções anteriores, vai-se abordar agora aplicações da lógica proposicional. Elas são melhores ilustradas através de exemplos.

Exemplo 2.14: Dado que o congresso se recusa aprovar novas leis, então a greve não terminará a menos que ela dure mais que um ano e o presidente da empresa renuncie. A greve terminará se o congresso se recusar a agir e a greve está apenas iniciando?

Inicialmente transforma-se as sentenças em símbolos:

P : O congresso se recusa a agir.
Q : A greve acabou.
R : O presidente da empresa renuncia.
S : A greve dura mais que um ano.

Então os fatos dados no exemplo podem ser representados através de fórmulas:

F1 : $(P \rightarrow (\sim Q \vee (R \wedge S))) \triangleq$ se o congresso se recusa aprovar novas leis, então a greve não terminará a menos que ela dure mais que um ano e o presidente da empresa renuncie.

F2 : $P \triangleq$ o congresso se recusa a agir.

F3 : $\sim S \triangleq$ a greve apenas iniciou.

A partir dos fatos F1, F2 e F3 pode-se concluir que a greve não terminará? Isto é, pode-se mostrar que ~Q é conseqüência lógica de F1, F2 e F3? Pelo Teorema 2.1, isto é equivalente a mostrar que $((P \rightarrow (\sim Q \vee (R \wedge S))) \wedge P \wedge \sim S) \rightarrow \sim Q$ é uma fórmula válida. Os valores verdade da fórmula acima sob todas as suas interpretações são mostrados na tabela 2.7.

A partir da tabela 2.7, pode-se ver que não há nenhuma interpretação sob a qual a fórmula é falsa. Portanto a fórmula $((P \rightarrow (\sim Q \vee (R \wedge S))) \wedge P \wedge \sim S) \rightarrow \sim Q$ é uma fórmula válida. Deste modo ~Q é conseqüência lógica de F1, F2 e F3. Isto é, pode-se concluir ~Q de F1, F2 e F3. Portanto a resposta é não, A GREVE NÃO TERMINARÁ.

P	**Q**	**R**	**S**	**F1**	**F2**	**F3**	**~Q**	**$(F1 \wedge F2 \wedge F3) \rightarrow \sim Q$**
V	V	V	V	V	V	F	F	V
V	V	V	F	F	V	V	F	V
V	V	F	V	F	V	F	F	V
V	V	F	F	F	V	V	F	V
V	F	V	V	V	V	F	V	V
V	F	V	F	V	V	V	V	V
V	F	F	V	V	V	F	V	V
V	F	F	F	V	V	V	V	V
F	V	V	V	V	F	F	F	V
F	V	V	F	V	F	V	F	V
F	V	F	V	V	F	F	F	V

F	V	F	F	V	F	V	F	V
F	F	V	V	V	F	F	V	V
F	F	V	F	V	F	V	V	V
F	F	F	V	V	F	F	V	V
F	F	F	F	V	F	V	V	V

Tabela 2.7 Tabela da Verdade de $(F1 \wedge F2 \wedge F3) \rightarrow \sim Q$

Exemplo 2.15: (Problema da Síntese Química)

Suponha que se tenha as seguintes reações químicas.

$$MgO + H_2 \rightarrow Mg + H_2O$$
$$C + O_2 \rightarrow CO_2$$
$$CO_2 + H_2O \rightarrow H_2CO_3$$

Suponha, também, que se tenha alguma quantidade de MgO, H_2, O_2 e C. Mostre que se pode fazer H_2CO_3.

Para este problema, deve-se considerar MgO, H_2, O_2 e C, como fórmulas atômicas. Então as reações químicas acima podem ser representadas pelas seguintes fórmulas:

A_1: $(MgO \wedge H_2) \rightarrow (Mg \wedge H_2O)$

A_2: $(C \wedge O_2) \rightarrow CO_2$

A_3: $(CO_2 \wedge H_2O) \rightarrow H_2CO_3$

Uma vez que temos MgO, H_2, O_2 e C, estes fatos podem ser representados pelas seguintes fórmulas:

A_4: MgO

A_5: H_2

A_6: O_2

A_7: C

Agora, o problema pode ser tratado como a prova de que H_2CO_3 é conseqüência lógica de $A_1, \ldots, A_7$, que, pelo teorema 2.2, é verdadeiro, se $(A_1 \wedge \ldots \wedge A_7) \wedge \sim H_2CO_3$ é inconsistente. Prova-se isto, transformando a fórmula $A_1 \wedge \ldots \wedge A_7 \wedge \sim H_2CO_3$ em sua forma normal disjuntiva.

$$\begin{aligned}
A_1 \wedge \ldots \wedge A_7 \wedge \sim H_2CO_3 = \;& ((MgO \wedge H_2) \rightarrow (Mg \wedge H_2O)) \wedge ((C \wedge O_2) \rightarrow CO_2) \wedge \\
& ((CO_2 \wedge H_2O) \rightarrow H_2CO_3) \wedge MgO \wedge H_2 \wedge O_2 \wedge C \wedge \sim H_2CO_3 \\
= \;& (\sim MgO \vee \sim H_2 \vee Mg) \wedge (\sim MgO \vee \sim H_2 \vee H_2O) \wedge \\
& (\sim C \vee \sim O_2 \vee CO_2) \wedge (\sim CO_2 \vee \sim H_2O \vee H_2CO_3) \wedge \\
& MgO \wedge H_2 \wedge O_2 \wedge C \wedge \sim H_2CO_3 \\
= \;& (\sim MgO \vee \sim H_2 \vee Mg) \wedge (\sim MgO \vee \sim H_2 \vee H_2O) \wedge MgO \wedge H_2 \wedge \\
& (\sim C \vee \sim O_2 \vee CO_2)\; C \wedge O_2 \wedge (\sim CO_2 \vee \sim H_2O \vee H_2CO_3) \wedge \\
& \wedge \sim H_2CO_3 \\
= \;& Mg \wedge H_2O \wedge MgO \wedge H_2 \wedge CO_2 \wedge C \wedge O_2 \wedge \\
& (\sim CO_2 \vee \sim H_2O) \wedge \sim H_2CO_3 \\
= \;& (\sim CO_2 \vee \sim H_2O) \wedge H_2O \wedge CO_2 \wedge Mg \wedge MgO \wedge H_2 \wedge C \wedge O_2 \wedge \\
& \sim H_2CO_3 \\
= \;& \square \wedge Mg \wedge MgO \wedge H_2 \wedge C \wedge O_2 \wedge \sim H_2CO_3 \\
= \;& \square
\end{aligned}$$

Uma vez que $\square$ é sempre falso, a fórmula $(A_1 \wedge \ldots \wedge A_7 \wedge \sim H_2CO_3)$ é inconsistente. Portanto, H_2CO_3 é conseqüência lógica de $A_1 \wedge \ldots \wedge A_7$. Isto é, pode-se obter H_2CO_3 a partir de MgO, H_2, O_2 e C. O procedimento acima de provar que uma fórmula é inconsistente pela sua transformação em $\square$ é, alguma vezes, chamado de *método multiplicativo*, pois o processo de transformação é muito semelhante a multiplicação por zero em uma expressão aritmética.

O exemplo 2.15 é apenas um simples exemplo de síntese química. Na realidade existem centenas de reações químicas. De maneira a provar de forma eficiente estes teoremas, necessita-se de métodos eficientes, que serão discutidos nos capítulos seguintes.

Nos exemplos anteriores, mostrou-se que lógica proposicional pode ser aplicada na solução de vários problemas. O método é, inicialmente, simbolizar os problemas através de fórmulas e, então, provar que estas fórmulas são válidas ou inconsistentes. Usou-se o método da tabela da verdade e o método multiplicativo para provar que uma fórmula é válida (inconsistente). Nos capítulos seguintes, métodos mais eficientes serão vistos para provar a validade e inconsistência de fórmulas.

2.7 EXERCÍCIOS

SEÇÃO 2.1

1. Simbolize as seguintes proposições por fórmulas.

(a) Uma relação é uma relação de equivalência se e somente se ela é reflexiva, simétrica e transitiva.

(b) Se a umidade está elevada, choverá esta tarde ou esta noite.

(c) O câncer não será curado a não ser que sua causa seja determinada e uma nova droga para o câncer seja encontrada.

(d) É necessário coragem e habilidade para escalar uma montanha.

(e) Se ele é um homem que trabalha arduamente em sua campanha, ele provavelmente será eleito.

2. Seja,

P $\triangleq$ Ele precisa de um doutor.　　Q $\triangleq$ Ele precisa de um advogado.
R $\triangleq$ Ele teve um acidente.　　S $\triangleq$ Ele está doente.
U $\triangleq$ Ele está ferido.

Coloque as seguintes fórmulas em Português.

(a) (S 1 P) 2 (R 1 Q)　　(b) P 1 (S 2 U)
(c) (P 2 Q) 1 R　　(d) (P 2 Q) 1 (S 2 U)
(e) ~(S 2 U) 1 ~P

SEÇÃO 2.2

3. Complete a tabela verdade da seguinte fórmula:

(~P 2 Q) 2 (~(P 2 ~Q))

P	**Q**	**~P**	**~Q**	**~P 2 Q**	**P 2 ~Q**	**~(P 2 ~Q)**	**(~P 2 Q) 2 (~(P 2 ~Q))**
V	V						
V	F						
F	V						
F	F						

SEÇÃO 2.3

4. Para cada uma das seguintes fórmulas, determine se é válida, inválida, inconsistente, consistente ou alguma combinação dessas dessas.

(a) ~(~P) 1 P　　(b) P 1 (P 2 Q)
(c) ~(P 2 Q) 2 ~Q　　(d) (P 2 Q) 1 P
(e) (P 1 Q) 1 (~Q 1 ~P)　　(f) (P 1 Q) 1 (Q 1 P)

(g) P 2 (P 1 Q)
(h) (P 2 (Q 1 P)) 1 P
(i) P 2 (Q 1 ~P)
(j) (P 2 ~Q) 2 (~P 2 Q)
(k) ~P 2 (~(P 1 Q))
(l) P 1 ~P
(m) ~P 1 P

5. Considere a seguinte proposição:

Se o congresso se recusa a promulgar novas leis, então a greve não terminará a menos que dure mais que um ano e o presidente da firma renuncie, e se o congresso promulga novas leis ou a greve não termina então a greve durará mais que um ano.
A proposição acima é contraditória? Explique.

SEÇÃO 2.4

6. Transforme as seguintes fórmulas em formas normais disjuntivas.

(a) (~P 2 Q) 1 R
(b) P 1 ((Q 2 R) 1 S)
(c) ~(P 2 ~Q) 2 (S 1 T)
(d) (P 1 Q) 1 R
(e) ~(P 2 Q) 2 (P 2 Q)

7. Transforme as seguintes fórmulas em formas normais conjuntivas.

(a) P 2 (~P 2 Q 2 R)
(b) ~(P 1 Q) 2 (P 2 Q)
(c) ~(P 1 Q)
(d) (P 1 Q) 1 R
(e) (~P 2 Q) 2 (P 2 ~Q)

8. É possível ter uma fórmula que é forma normal conjuntiva e também forma normal disjuntiva? Caso exista, dê um exemplo.

9. Verifique cada um dos seguintes pares de fórmulas equivalentes através de transformações em ambos os lados na mesma forma normal.

(a) P 2 P = P e P 2 P = P
(b) (P 1 Q) 2 (P 1 R) = (P 1 (Q 2 R))
(c) (P 1 Q) 1 (P 2 Q) = (~P 1 Q) 2 (Q 1 P)
(d) P 2 Q 2 (~P 2 ~Q) = ~P 2 ~Q 2 (P 2 Q)
(e) P 2 (P 1 (P 2 Q)) = ~P 2 ~Q 2 (P 2 Q)

SEÇÕES 2.5 E 2.6

10. Prove que (~Q 1 ~P) é uma implicação lógica de (P 1 Q).

11. Se o congresso se recusa a promulga novas leis, então a greve não terminará a não ser que ela dure mais que um ano e o presidente da firma renuncie. Suponha que o congresso se recusa a agir, a greve termine e o presidente da firma não renuncie. A greve durou mais que um ano?

12. Considere as seguintes proposições:

F1 $\triangleq$ Tom não pode se um bom estudante a não ser que seja dedicado e seu pai o financie.

F2 $\triangleq$ Tom é um bom estudante se seu pai o financia.

Mostre que F2 é uma implicação lógica de F1.

13. Mostre que para as seguintes proposições, F2 é uma implicação lógica de F1.

F1 $\triangleq$ Se o presidente não tem a autoridade apropriada ou se ele não quer assumir a responsabilidade, então nem a ordem será restaurada e nem os disturbios pararão de se expandir a menos que os desordeiros tornem-se cansados da desordem e as autoridades locais comecem a tomar medidas conciliatórias.

F2 $\triangleq$ Se o presidente não quer assumir a responsabilidade e os desordeiros não estão cansados da desordem, então os desordeiros se expandirão.

14. Mostre que Q é uma implicação lógica de (P 1 Q) e P. Esta é a chamada regra ***modus ponens***.

CAPÍTULO 3 - LÓGICA DE PREDICADOS

3.1 INTRODUÇÃO

Na lógica proposicional, a maioria dos elementos básicos são átomos. Através dos átomos se constrõem fórmulas. Então, usa-se as fórmulas para expressar várias idéias complexas. Como discutido no Capítulo 2, nesta lógica simples, um átomo representa uma sentença declarativa que pode ser V ou F, mas não ambos. Um átomo é tratado como uma entidade única. Seus atributos e componentes são desprezados. Entretanto, existem muitas idéias que não podem ser tratadas desta maneira simplista. Por exemplo, considere a seguinte declaração: Todo homem é mortal.

Uma vez que Confúcio é um homem, ele é mortal.

O raciocínio acima é intuitivamente correto, porém se for aplicada a seguinte representação: P: Todo homem é mortal,

Q: Confúcio é um homem,

R: Confúcio é mortal,

então R não é conseqüência lógica de P e Q dentro do contexto da lógica proposicional. Isto acontece porque os atributos de P, Q e R não são utilizados na lógica proposicional. Neste Capítulo, será introduzida a lógica de predicados, que apresenta mais três conceitos lógicos, chamados: termos, predicados e quantificadores. Será visto na seqüência que a maioria da linguagem, tanto corrente (Português) quanto matemática, pode ser simbolizada como lógica de predicados.

Da mesma forma que na lógica proposicional, inicia-se definindo átomos na lógica de predicados. Antes de formalizar a definição de átomo, observe alguns exemplos.

Suponha que se deseja representar "*x é maior que 3*". Inicialmente define-se um predicado MAIOR(x,y) significando "*x é maior que y*" (observe que um predicado é uma relação). Então a sentença "*x é maior que 3*" é representada por MAIOR(x,3).

De maneira semelhante, pode representar "*x ama y*" pelo predicado AMA(x,y). Então a frase "*João ama Maria*" pode ser representada por AMA(João, Maria).

Também se pode usar funções na lógica de predicados. Por exemplo, pode usar adição(x,y) para representar "*x + y*" e pai(x) significando "*o pai de x*". As sentenças "*x + 1 é maior que x*" e "*O pai de João o ama*" podem ser simbolizadas por MAIOR(adição(x,1),x) e AMA(pai(João),João), respectivamente.

Nos exemplos anteriores, MAIOR(x,3), AMA(João, Maria), MAIOR(adição(x,1),x) e AMA(pai(João),João) são todos átomos na lógica de predicados, onde MAIOR e AMA são predicados; x é uma variável; 3, João e Maria são constantes; e pai e adição são funções.

Em geral, é permitido o uso de quatro tipos de símbolos na construção de um átomo:

i. Constantes: são, geralmente, nomes de objetos, tais como: 3, João e Maria.
ii. Variáveis: são escritos, normalmente, em letras minúsculas ou letras sublinhadas: x, y, z,...
iii. Funções: são escritos, normalmente, em letras minúsculas, f, g, h, ... ou cadeias significativas como pai e adição.
iv. Predicados: são escritos, normalmente, em letras maiúsculas, P, Q, R, ... ou cadeias significativas como MAIOR e AMA.

Qualquer função ou predicado necessita de um número especificado de argumentos. Se a função f necessita de n argumentos, f é chamada de função n-arg. Observe que uma constante pode ser considerada uma função sem argumentos. Do mesmo modo, se um predicado P necessita de n argumentos, então P é chamado de predicado n-arg. Por exemplo, pai é uma função 1-arg e MAIOR e AMA são predicados 2-arg.

Uma função é um mapeamento de uma lista de constantes para uma constante. Por exemplo, pai(João) é uma função que mapeia a pessoa chamada João para uma pessoa que é o pai de João. Portanto pai(João) representa uma pessoa, mesmo sem se conhecer seu nome. Chama-se pai(João) um termo na lógica de predicados. Formalmente, tem-se a seguinte definição:

Definição: Termos são definidos, recursivamente, como:

i. Uma constante é um termo.
ii. Uma variável é um termo.
iii. Se f é uma função n-arg e t1, t2, ... , tn são termos, então f(t1,t2, ... ,tn) é um termo.
iv. Todos os termos são gerados pela aplicação das regras acima.

Exemplo 3.1: Sendo x e 1 termos, e adição uma função 2-arg, adição(x,1) é um termo pela definição acima.

Além disso, pode-se ver que adição(adição(x,1),x) e pai(pai(João)) também são termos, onde o primeiro representa (x + 1) + x e o seguinte representa o avô de João.

Um predicado é um mapeamento de uma lista de constantes para V ou F. Por exemplo, MAIOR é um predicado, MAIOR(5,3) é V, mas MAIOR(2,3) é F. Tendo definido termos, pode-se definir formalmente um átomo na lógica de predicados.

Definição: Se P é um predicado n-arg, e t1, t2, ... , tn são termos, então P(t1,t2, ... ,tn) é um átomo.

Estando os átomos definidos, pode-se usar os mesmos cinco conectivos lógicos (~, 2 , 2 , 1 , 1) vistos no Capítulo 2 para construir fórmulas. Além disso, uma vez que foram introduzidas variáveis. Os símbolos 3 e 3 são chamados, respectivamente, de quantificador existencial e quantificador universal. Se x é uma variável, então (3x) é lido como "*para todo x*" e (3x) é lido como "*existe um x*", "*para algum x*", "*para no mínimo um x*". Vai-se exemplificar para melhor enxergar como os quantificadores podem ser usados.

Exemplo 3.2: Simbolize as seguintes sentenças:

(a) Todo número racional é um número real.
(b) Existe um número que é primo.
(c) Para todo número x, existe um número y tal que x < y.

Representando "*x é um número primo*" por P(x), "*x é um número racional*" por Q(x), "*x é um número real*" por R(x) e "*x é menor que y*" por MENOR(x,y), as sentenças acima podem ser escritas como:

(a') (3x) (Q(x) 1 R(x))
(b') (3x) P(x)
(c') (3x)(3y) MENOR(x,y)

Cada uma das expressões (a'), (b') e (c') é chamada fórmula. Antes da definição de fórmula, vai-se ver a distinção entre variáveis dependentes e variáveis livres. Para isto, inicialmente, define-se o escopo (área de abrangência) do quantificador como sendo a fórmula na qual ele ocorre, ou seja, a fórmula na qual o quantificador é aplicado. Por exemplo, o escopo tanto do quantificador universal quanto do quantificador existencial na fórmula (c') é MENOR(x,y). O escopo do quantificador universal na fórmula (a') é (Q(x) 1 R(x)).

Definição: A ocorrência de uma variável em uma fórmula é dependente se e somente se sua ocorrência está dentro do escopo de um quantificador utilizando a variável ou a ocorrência é no próprio quantificador. A ocorrência de uma variável em uma fórmula é livre se e somente se a ocorrência não é dependente.

Definição: Uma variável é livre na fórmula se, no mínimo, uma ocorrência dela é livre na fórmula. Uma variável é dependente se, no mínimo, uma ocorrência dela é dependente.

Na fórmula (3x) P(x,y), uma vez que ambas as ocorrências de x são dependentes, a variável x é dependente. Entretanto, a variável y é livre, uma vez que sua única ocorrência é livre. Observe que uma variável pode ser livre e dependente em uma mesma fórmula. Por exemplo, y é livre e dependente na fórmula (3x) P(x,y) 2 (3y) Q(y).

Pode-se agora definir, formalmente, uma fórmula, usando átomos, conectivos lógicos e quantificadores.

Definição: Fórmulas bem formadas (fbf) ou, resumidamente, fórmulas, na lógica de predicados são definidas, recursivamente como segue:

i. Um átomo é uma fórmula (observe que átomo é uma abreviatura para fórmula atômica).
ii. Se F e G são fórmulas, então ~(F), (F2G), (F2G), (F1G), (F1G) são fórmulas.
iii. Se F é uma fórmula e x é uma variável livre em F, então (3x) F e (3x) F são fórmulas.
iv. Fórmulas são geradas somente pela aplicação, um número finito de vezes de i, ii e iii.

Como no Capítulo 2, os parênteses poderão ser omitidos quando não cause confusão. As regras vistas anteriormente podem ser ampliadas considerando que os quantificadores tenham as áreas de abrangência restritas. Por exemplo, (3x) A 2 B, representa (((3x) A) 2 B).

Exemplo 3.3: Transforme a sentença "*Todo homem é mortal. Confúcio é um homem. Portanto Confúcio é mortal*" em uma fórmula. Representando "*x é um homem*" por HOMEM(x) e "*x é mortal*" por MORTAL(x). Então:

"*Todo homem é mortal*" pode ser representado por: (3x) (HOMEM(x)1MORTAL(x)),
"*Confúcio é um homem*" pode ser representado por: HOMEM(Confúcio) e
"*Confúcio é mortal*" pode ser representado por: MORTAL(Confúcio), portanto a sentença completa pode ser representada por:

(3x) (HOMEM(x)1MORTAL(x)) 2 HOMEM(Confúcio) 1 MORTAL(Confúcio)

Exemplo 3.4: Os axiomas básicos dos números naturais são os seguintes:
A1: Para todo número, existe um e somente um sucessor imediato.
A2: Não existe nenhum número para o qual 0 seja o seu sucessor imediato.
A3: Para todo número diferente de 0, existe um e somente um predecessor imediato.

Fazendo f(x) e g(x) representar o sucessor imediato de x e o predecessor imediato de x, respectivamente e I(x,y) representar "*x é igual a y*". Então os axiomas podem ser representados pelas seguintes fórmulas:
A1': (3x)(3y) (I(y,f(x)) 2 (3z) (I(z,f(x)) 1 I(y,z)))
A2': ~((3x) I(0,f(x)))
A3': (3x) (~I(x,0) 1 ((3y) (I(y,g(x)) 2 (3z) (I(z,g(x)) 1 I(y,z)))))

3.2 INTERPRETAÇÃO DE FÓRMULAS NA LÓGICA DE PREDICADOS

Ná lógica proposicional, uma interpretação é uma atribuição de valores verdade aos átomos. Na lógica de predicados, uma vez que existem variáveis envolvidas, tem-se algo mais. Para definir uma interpretação para uma fórmula na lógica de predicados, tem-se que especificar duas coisas: o domínio e uma atribuição de valores para as constantes, funções e predicados ocorrendo na fórmula. A seguir tem-se a definição formal de uma interpretação de uma fórmula na lógica de predicados.

Definição: Uma interpretação de uma fórmula F na lógica de predicados, consiste de um domínio D, não vazio e uma atribuição de valores para cada constante, função e predicado ocorrendo na fórmula, como segue:

1. Para cada constante, atribui-se um elemento em D.
2. Para cada função n-arg, atribui-se um mapeamento de D^n para D. (Observe que D^n = {(x1, ... , xn) | x1 2D, ... , xn 2D}).
3. Para cada predicado n-arg, atribui-se um mapeamento de D^n para { V , F }.

Algumas vezes, para enfatizar o domínio D, fala-se de uma interpretação de uma fórmula sobre D. Quando se avalia o valor verdade de uma fórmula em uma interpretação sobre o domínio D, (3x) será interpretado como "*para todos os elementos de D*" e (3x) como "*existe um elemento em D*".

Para toda interpretação de uma fórmula sobre um domínio D, a fórmula pode ser avaliada V ou F de acordo com as seguintes regras:

1. Se os valores verdade das fórmulas G e H são avaliados, então os valores verdade das fórmulas ~(G), (G2H) (G2H), (G1H), (G1H) são avaliados utilizando-se a tabela 2.1 do Capítulo 2.
2. (3x) G é avaliado V se o valor de G é avaliado V para todo x em D, caso contrário é avaliado F.
3. (3x) G é avaliado V se o valor de G é V para, no mínimo, um x em D, caso contrário é avaliado F.

Note que qualquer fórmula contendo variáveis livres não pode ser avaliada. A partir deste ponto, vai-se assumir que, ou as fórmulas não contém variáveis livres ou as variáveis livres são tratadas como constantes.

Exemplo 3.5: Considere as fórmulas: (3x) P(x) e (3x) ~P(x) e uma interpretação como segue:

Domínio: D = { 1 , 2 }

Atribuição para P:

P(1)	P(2)
V	F

Fica fácil confirmar que (3x) P(x) é F nessa interpretação, pois P(x) não é V para x = 2. Por outro lado, uma vez que ~P(2) é V nessa interpretação (3x) ~P(x) também é V.

Exemplo 3.6: Considere a fórmula: (3x)(3y) P(x,y) e uma interpretação como segue:

Domínio: D = { 1 , 2 }

Atribuição para P:

P(1,1)	P(1,2)	P(2,1)	P(2,2)

V	F	F	V

Se x = 1, pode-se ver que existe um y, de valor 1, tal que P(1,y) é V.
Se x = 2, também existe um y, de valor 2, tal que P(2,y) é V.
Portanto, na interpretação acima, para todo x em D, existe um y tal que P(x,y) é V, isto é, (3x)(3y) P(x,y) é V nessa interpretação.

Exemplo 3.7: Considere a fórmula G: (3x) (P(x) 1 Q(f(x),a)). Existe uma constante a, uma função f, 1-arg, um predicado P, 1-arg e um predicado Q, 2-arg. Uma interpretação I de G pode ser descrita como segue:

Domínio: D = { 1 , 2 }

Atribuição para a:

a
1

Atribuição para f:

f(1)	f(2)
2	1

Atribuição para P e Q:

P(1)	P(2)	Q(1,1)	Q(1,2)	Q(2,1)	Q(2,2)
F	V	V	V	F	V

Se x = 1, então: P(x) 1 Q(f(x),a) = P(1) 1 Q(f(1),a)
= P(1) 1 Q(2,1) = F 1 F = **V**
Se x = 2, então: P(x) 1 Q(f(x),a) = P(2) 1 Q(f(2),a)
= P(2) 1 Q(1,1) = V 1 V = **V**

Uma vez que P(x) 1 Q(f(x),a) é V para todos os valores de x no domínio D, a fórmula (3x) (P(x) 1 Q(f(x),a)) é verdadeira na interpretação I.

Exemplo 3.8: Encontre os valores verdade para as seguintes fórmulas, considerando a interpretação dada no exemplo 3.7.

(a) (3x) (P(f(x)) 2 Q(x,f(a)))
Se x = 1, então: P(f(x)) 2 Q(x,f(a)) = P(f(1)) 2 Q(1,f(a))
= P(2) 2 Q(1,f(1)) = P(2) 2 Q(1,2) = V 2 V = **V**
Se x = 2, então: P(f(x)) 2 Q(x,f(a)) = P(f(2)) 2 Q(2,f(a))
= P(1) 2 Q(2,f(1)) = P(1) 2 Q(2,2) = F 2 V = **F**
Uma vez que existe um elemento no domínio D, x = 1, tal que P(f(x)) 2 Q(x,f(a)) é V, então o valor da fórmula (3x) (P(f(x)) 2 Q(x,f(a))) é V sobre a interpretação I.

(b) (3x) (P(x) 2 Q(x,a))
Se x = 1, então: P(x) 2 Q(x,a) = P(1) 2 Q(1,1) = F 2 V = **F**
Se x = 2, então: P(x) 2 Q(x,a) = P(2) 2 Q(2,1) = V 2 F = **F**
Uma vez que não existe nenhum elemento no domínio D, tal que P(x) 2 Q(x,a) seja V, a fórmula (3x) (P(x) 2 Q(x,a)) é F sobre a interpretação I.

(c) (3x)(3y) (P(x) 2 Q(x,y))

Se x = 1, então: P(x) = P(1) = **F**, portanto P(x) 2 Q(x,y) é F para y = 1 e y = 2, uma vez que existe um x, x = 1, tal que (3y) (P(x) 2 Q(x,y)) é F, portanto a fórmula (3x)(3y) (P(x)2Q(x,y)) é falsa sobre a interpretação I, isto é, a fórmula é falsificada por I.

Uma vez que as interpretações são definidas, todos os conceitos, tais como: validade, inconsistência e conseqüência lógica definidos no Capítulo 2, podem ser definidos de forma análoga para as fórmulas da lógica de predicados.

Definição: Uma fórmula G é consistente se e somente se existe uma interpretação I tal que G é avaliada V em I. Se a fórmula G é V na interpretação I, diz-se que I é um modelo de G e I satisfaz G.

Definição: Uma fórmula G é inconsistente se e somente se não existe nenhuma interpretação que satisfaça G.

Definição: Uma fórmula G é válida se e somente se toda interpretação de G satisfaz G.

Definição: Uma fórmula G é conseqüência lógica das fórmulas F1, ... , Fn se e somente se para toda interpretação: se F1 2 F2 2 ... 2 Fn é V em I, G também é V em I.

As relações entre validade (inconsistência) e conseqüência lógica, como mostrado nos teoremas 2.1 e 2.2 são também verdadeiras para a lógica de predicados. De fato, a lógica de predicados pode ser considerada uma extensão da lógica proposicional. Quando uma fórmula da lógica de predicados não contém variáveis, nem quantificadores, ela pode ser tratada como uma fórmula da lógica proposicional.

Exemplo 3.9: Os seguintes exemplos vão ser deixados para que sejam provados por você:
(1) (3x) P(x) 2 (3y) ~P(y) é inconsistente.
(2) (3x) P(x) 1 (3y) P(y) é válida.
(3) P(a) 1 ~((3x) P(x)) é consistente.
(2) (3x) P(x) 2 ((3y) ~P(y)) é válida.

Exemplo 3.10: Considere as fórmulas: F1: (3x) (P(x) 1 Q(x))
F2: P(a)

Quer-se provar que a fórmula Q(a) é conseqüência lógica de F1 e F2.
Considere qualquer interpretação I que satisfaça (3x) (P(x) 1 Q(x)) 2 P(a). Certamente, nessa interpretação P(a) é V. Assuma que Q(a) não é V nessa interpretação; então ~P(a) 2 Q(a), isto é P(a) 1 Q(a) é F em I. Isto significa que (3x) (P(x) 1 Q(x)) é F em I, o que é impossível. Portanto, Q(a) deve ser V em toda a interpretação que satisfaz (3x) (P(x) 1 Q(x)) 2 P(a). Isto significa que Q(a) é conseqüência lógica de F1 e F2.

Na lógica de predicados, uma vez que há um número infinito de domínios, em geral, existe um número infinito de interpretações de uma fórmula. Portanto, de maneira diferente da lógica proposicional, não é possível verificar a validade ou inconsistência de uma fórmula avaliando-a sobre todas as suas possíveis interpretações. Nos próximos capítulos, vai-se ver procedimentos para verificar inconsistências na lógica de predicados.

3.3 FORMAS NORMAIS PRENEX NA LÓGICA DE PREDICADOS

Na lógica proposicional viu-se duas formas normais - a forma normal conjuntiva e a forma norma disjuntiva. Na lógica de predicados existe também uma forma normal chamada de "*forma normal prenex*". A razão para utilizar uma forma normal prenex de uma fórmula é para simplificar os métodos de prova, que serão vistos mais adiante.

Definição: Uma fórmula *F* na lógica de predicados é dita estar em uma *forma normal prenex* se e somente se a fórmula *F* está na forma de: $(Q_1 x_1) \ldots (Q_n x_n)(M)$,
onde todo $(Q_i\ x_i)$, i = 1, ..., n, é $(\forall x_i)$ ou $(\exists x_i)$, e *M* é uma fórmula que não possui quantificadores. $(Q_1\ x_1) \ldots (Q_n x_n)$ é chamada de *prefixo* e *M* é chamada de *matriz* da fórmula *F*.

Abaixo são mostradas algumas fórmulas na forma normal prenex:

i. $(\forall x)\ (\forall y)\ (P(x,y) \wedge Q(y))$
ii. $(\forall x)\ (\forall y)\ (\sim P(x,y) \rightarrow Q(y))$
iii. $(\forall x)\ (\forall y)\ (\exists z)\ (Q(x,y) \rightarrow R(z))$

Dada uma fórmula, será visto agora um método para transformá-la na forma normal prenex. Isto é obtido considerando alguns pares de fórmulas equivalentes na lógica de predicados. Lembrando que duas fórmulas F e G são *equivalentes*, representados por F = G, se e somente se os valores de F e G são os mesmos sob todas as interpretações. Os pares básicos de fórmulas equivalentes dados na tabela 2.6 no Capítulo 2 ainda são válidos para a lógica de predicados. Existem ainda outros pares de fórmulas equivalentes que contém quantificadores. Serão vistos agora esses pares adicionais de fórmulas equivalentes.

Seja F uma fórmula contendo uma variável x. Para salientar que a variável x está em F, representa-se F por F[x]. Seja G uma fórmula que não contém a variável x. Então tem-se os seguintes pares de fórmulas equivalentes, onde Q é $\forall$ ou $\exists$. Para simplificar, chama-se cada par abaixo de "*lei*".

(3.1a) $(Qx)\ F[x] \vee G = (Qx)\ (F[x] \vee G)$
(3.1b) $(Qx)\ F[x] \wedge G = (Qx)\ (F[x] \wedge G)$
(3.2a) $\sim((\forall x)\ F[x]) = (\exists x)\ (\sim F[x])$
(3.2b) $\sim((\exists x)\ F[x]) = (\forall x)\ (\sim F[x])$

As leis (3.1a) e (3.1 b) são obviamente verdadeiras, desde que G não contenha x e portanto possa ser trazido para dentro do escopo do quantificador Q. As leis (3.2a) e (3.2b) não são difíceis de provar. Seja I uma interpretação arbitrária qualquer sobre um domínio D. Se ~ (($\forall$x) F[x]) é verdadeiro em I, então ($\forall$x) F[x] é falso em I. Isto significa que existe um elemento *e* em D tal que F[*e*] é falso, isto é, ~F[*e*] é verdadeiro em I. Portanto, (3x)(~F[x]) é verdadeiro em I. Por outro lado, se ~((3x) F[x]) é falso em I, então (3x) F[x] é verdadeiro em I. Isso significa que F[x] é verdadeira para todo elemento x em D, isto é, ~F[x] é falso para todo elemento x em D. Portanto, (3x)(~F[x]) é falso em I. Desde que ~((3x) F[x]) e (3x)(~F[x]) sempre assume o mesmo valor para qualquer interpretação arbitrária, por definição, ~((3x) F[x]) = (3x)(~F[x]). Por consequência, a lei (3.2a) fica provada. Do mesmo modo, pode-se provar a lei (3.2b).

Supondo que F[x] e H[x] são duas fórmulas contendo x. Existem duas outras leis:

(3.3a) (3x) F[x] 2 (3x) H[x] = (3x) (F[x] 2 H[x])
(3.3b) (3x) F[x] 2 (3x) H[x] = (3x) (F[x] 2 H[x]).

Isto é, o quantificador universal 3 e o quantificador existencial 3 podem distribuir-se sobre 2 e 2, respectivamente.

As provas (3.3a) e (3.3b) não são difíceis, portanto ficarão a cargo do estudante. Porém, o quantificador universal 3 e o quantificador existencial 3 *não podem* distribuir-se sobre 2, e 2, respectivamente. Isto é,

(3x) F[x] 2 (3x) H[x] 1 (3x) (F[x] 2 H[x]) e
(3x) F[x] 2 (3x) H[x] 1 (3x) (F[x] 2 H[x]).

Para este caso, faz-se algo especial. Desde que todas as variáveis dependentes na fórmula podem ser consideradas como uma variável fantasma, todas as variáveis dependentes x podem ser renomeadas para z, e a fórmula (3x) H[x] torna-se (3z) H[z] ; isto é, (3x) H[x] = (3z) H[z]. Supodo que se escolha a variável z que não aparece em F[x]. Então,

(3x) F[x] 2 (3x)H[x] = (3x)F[x] 2 (3z) H[z]
(renomeando todos os x's ocorrentes em (3x)H[x] para z)
= (3x)(3z)(F[x] 2 H[z]) (por 3.1a)

Da mesma forma, pode-se ter

(3x) F[x] 2 (3x) H[x] = (3x)F[x] 2 (3z) H[z]
(renomeando todos os x's ocorrentes em (3x) H[x] para z)
= (3x) (3z) (F[x] 2 H[z]) (por 3.1b)

Portanto, para esses dois casos, pode-se trazer todos os quantificadores para a esquerda da fórmula. Em geral, tem-se:

(3.4a) (Q_1x) F[x] 2 (Q_2x) H[x] = (Q_1x)(Q_2z) (F[x] 2 H[z])
(3.4b) (Q_3x) F[x] 2 (Q_4x) H[x] = (Q_3x)(Q_4z) (F[x] 2 H[z])

onde Q_1 , Q_2, Q_3 e Q_4 são 3 ou 3, e z não aparece em F[x]. É claro que, se Q_1 = Q_2 = 3 e Q_3 = Q_4 = 3 , então não é necessário renomear os x's em Q_2x H[x] ou Q_4x H[x]. Pode-se usar (3.3) diretamente.

Usando as leis (2.1) - (2.10) e as leis (3.1) - (3.4), pode-se sempre transformar uma fórmula na forma normal prenex. A seguir apresenta-se um resumo do procedimento de transformação.

Transformando Fórmulas na Forma Normal Prenex

Passo 1: Usar as leis F 1 G = (F 1 G) 2 (G 1 F) (2.1)
F 1 G = ~ F 2 G (2.2)
para eliminar os conetivos lógicos 1 e 1.

Passo 2: Repetidamente usar a lei ~ (~F) = F (2.9),
as leis de De Morgan ~ (F 2 G) = ~F 2 ~G (2.10a),
~ (F 2 G) = ~F 2 ~G (2.10b),
e as leis ~ ((3x) F[x]) = (3x) (~F[x]) (3.2a),
~ ((3x) F[x]) = (3x) (~F[x]) (3.2b),

para trazer o sinal de negação para a imediatamente antes dos termos.

Passo 3: Renomear as variáveis dependentes se necessário.

Passo 4: Usar as leis (Qx) F[x] ∨ G = (Qx) (F[x] ∨ G) (3.1a),
(Qx) F[x] ^ G = (Qx) (F[x] ^ G) (3.1b),
(3x) F[x] 2 (3x) H[x] = (3x) (F[x] 2 H[x]) (3.3a),
(3x) F[x] 2 (3x) H[x] = (3x) (F[x] 2 H[x]) (3.3b),
(Q_1x) F[x] 2 (Q_2x) H[x] = (Q_1x)(Q_2z) (F[x] 2 H[z]) (3.4a),
(Q_3x) F[x] 2 (Q_4x) H[x] = (Q_3x)(Q_4z) (F[x] 2 H[z]) (3.4b)

para mover os quantificadores para a esquerda de toda fórmula para obter uma forma normal prenex.

Exemplo 3.11: Transformar a fórmula (3x) P(x) 1 (3x) Q(x) na forma normal prenex.

(3x) P(x) 1 (3x) Q(x) = ~ ((3x) P(x)) 2 (3x) Q(x) por (2.2)
= (3x) (~P(x)) 2 (3x) Q(x) por (3.2a)
= (3x) (~P(x) 2 Q(x)) por (3.3b).

Portanto, a forma normal prenex de (3x) P(x) 1 (3x) Q(x) é (3x) (~P(x) 2 Q(x)).

Exemplo 3.12: Obter a forma normal prenex para a fórmula:
(3x)(3y) ((3z) (P(x,z) 2 P (y,z)) 1 (3u) Q(x,y,u)).
= (3x)(3y) (~((3z) (P(x,z) 2 P(y,z))) 2 (3u) Q(x,y,u)) por (2.2)
= (3x)(3y)((3z) (~P(x,z) 2 ~P(y,z)) 2 (3u) Q(x,y,u)) por (3.2b) e (2.10b)
= (3x)(3y)(3z)(3u) (~P(x,z) 2 ~P(y,z) 2 Q(x,y,u)), usando (3.1a), move-se os quantificadores para a esquerda, obtendo por fim a forma normal prenex da primeira fórmula.

3.4 APLICAÇÕES DA LÓGICA DE PREDICADOS

Nesta seção serão vistos alguns exemplos para ilustrar algumas aplicações da lógica de predicados na solução de problemas. Assim como na lógica proposicional, a abordagem mais comum é primeiro simbolizar problemas por fórmulas e então provar que as fórmulas são válidas ou inconsistentes.

Exemplo 3.13: Considerando o exemplo 3.3. Existem dois axiomas:
A_1 : (3x) (HOMEM(x) 1 MORTAL(x)).
A_2 : HOMEM(Confúcio).

De A_1 e A_2, mostra-se que Confúcioé mortal. Isto é, mostra-se que MORTAL(Confúcio) é uma consequência lógica de A_1 e A_2. Tem-se:

A_1 2 A_2 : (3x) (HOMEM(x) 1 MORTAL(x)) 2 HOMEM(Confúcio).

Se (A_1 2 A_2) é verdadeiro em uma interpretação I , então ambos A_1 e A_2 são verdadeiros em I . Desde que (HOMEM(x) 1 MORTAL(x)) é verdadeiro para todo x, quando x é substituído por "Confúcio", (HOMEM(Confúcio) 1 MORTAL(Confúcio)) é verdadeiro em I. Isto é, ~HOMEM(Confúcio) 2 MORTAL(Confúcio) é verdadeiro em I. No entanto, ~HOMEM(Confúcio) é falso em I desde que HOMEM(Confúcio) for verdadeiro em I. Portanto, MORTAL(Confúcio) deve ser verdadeiro em I. Tem-se portanto mostrado que

MORTAL(Confúcio) é verdadeiro em I quando (A_1 2 A_2) é verdadeiro em I. Por definição, MORTAL(Confúcio) é uma consequência lógica de A_1 e A_2 .

Exemplo 3.14: Nenhum vendedor de carros usados compra uma carro usado para a sua família. Algumas pessoas que compram carros usados para sua família são totalmente desonestas. Conclui-se que algumas pessoas totalmente desonestas não são vendedores de carros usados.

Fazendo U(x), B(x), e D(x) representar "*x é um vendedor de carros usados*", "*x compra um carro usado para sua família*", e "*x é totalmente desonesto*", respectivamente. Então tem-se: A_1: (3x) (U(x) 1 ~B(x))

A_2: (3x) (B(x) 2 D(x))

E precisa-se mostrar que: A_3 : (3x) (D(x) 2 ~U(x)) é uma consequência lógica de A_1 e A_2.

Assume-se que A_1 e A_2 são verdadeiros em uma interpretação I sob o domínio D. Desde que A_2 é verdadeiro em I, existe um x em D, chamado de a, tal que B(a) 2 D(a) é verdadeiro em I. Portanto B(a) é verdadeiro em I, isto é, ~B(a) é falso em I. A_1 pode ser escrito da seguinte forma: A_1: (3x) (~U(x) 2 ~B(x)).

Desde que A_1 é verdadeiro em I e ~B(a) é falso em I, ~U(a) deve ser verdadeiro em I. Porém, desde que B(a) 2 D(a) é verdadeiro em I, D(a) é verdadeiro em I. Portanto, D(a) 2 ~U(a) é verdadeiro em I. Então, A_3 , isto é , (3x) (D(x) 2 ~U(x)), é verdadeiro em I. Por consequência, A_3 é uma consequência lógica de A_1 e A_2 .

Exemplo 3.15: Alguns pacientes gostam de todos os médicos. Nenhum paciente gosta de charlatões. Portanto, nenhum médico é um charlatão.

Representa-se: P(x): x é um paciente,
D(x): x é um médico,
Q(x): x é um charlatão,
L(x,y): x gosta de y.

Então os fatos e as conculusões podem ser representados por:

F_1: (3x) (P(x) 2 (3y)(D(y) 1 L(x,y)))

F_2: (3x) (P(x) 1 (3y)(Q(y) 1 ~L(x,y)))

G : (3x) (D(x) 1 ~(Q(x)).

Agora será mostrado que G é uma consequencia lógica de F_1e F_2. Seja I uma interpretação arbitrária sobre o domínio D .Suponha que F_1e F_2 são verdadeiros em I. Desde que F_1, isto é, (3x) (P(x) 2 (3y)(D(y) 1 L(x,y))) é verdadeiro em I, existe algum elemento, chamado de *e*, em D talque (P(*e*) 2 (3y) (D(y) 1 L(*e*,y))) é verdadeiro em I.

Isto é, ambos P(*e*) e (3y) (D(y) 1 L(*e*,y)) são verdadeiros em I. Por outro lado, desde que (P(x) 1 (3y)(Q(y) 1 ~L(x,y))) é verdadeiro em I para todos os elementos de x em D, certamente (P(*e*) 1 (3y)(Q(y) 1 ~L(*e*,y))) é verdadeiro em I. Desde que P(*e*) é verdadeiro em I, (3y)(Q(y) 1 ~L(e,y)) deve ser ser verdadeiro em I. Por isso sabe-se que para todo elemento y em D, tanto (D(y) 1 L(e,y)) e (Q(y) 1 ~L(e,y)) são verdadeiros em I.

Se D(y) é falso em I, então (D(y) 1 ~(Q(y)) é verdadeiro em I. Se D(y) é verdadeiro em I, então L(e,y) deve ser verdadeiro em I, desde que (D(y) 1 L(e,y)) é verdadeiro em I. Portanto, Q(y) deve ser falso em I, desde que (Q(y) 1 L(e,y)) é verdadeiro em I. Consequentemente, (D(y) 1 ~(Q(y)) é verdadeiro em I. Portanto, (D(y) 1 ~(Q(y)) é verdadeiro para todo y em D;

isto é , (3y) (D(y) 1 ~(Q(y)) é verdadeiro em I. Por isso, mostra-se que se F_1 e F_1 são verdadeiros em I, (3y) (D(y) 1 ~(Q(y)) é verdadeiro em I. Isso mostra que G é uma consequência lógica de F_1 e F_2 .

Nos exemplos acima, pode-se mostrar que as conclusões seguem dos fatos dados. Uma demonstração que uma conclusão segue de axiomas é chamada de *prova*. Um procedimento para chegar às provas é chamado de *método de prova*. Do capítulo 4 a 7, serão dados métodos de provas que *mecanicamente* usam regras de inferência para encontrar uma prova. Esses métodos de prova são muito eficientes. Poder-se-á notar que os dois exemplos anteriores tornam-se fáceis de provar se forem usados os métodos de prova.

3.5 EXERCÍCIOS

Seção 3.1

1. Fazendo P(x) e Q(x) representar "*x é um número racional*" e "*x é um número real*", respectivamente. Simbolizar as seguintes sentenças:

1.1 Todo número racional é um número real.
1.2 Alguns números reais são números racionais.
1.3 Nem todo número real é um número racional.

2. Fazendo C(x) significar "*x é um vendedor de carros usados*", e H(x) significar "*x é honesto*". Traduzir as linhas seguintes para português:

2.1 (3x) C(x)
2.2 (3x) H(x)
2.3 (3x) (C(x) 1 ~H(x))
2.4 (3x) (C(x) 2 H(x))
2.5 (3x) (H(x) 1 C(x)).

3. Fazendo P(x), L(x), R(x,y,z), e E(x,y) representar "*x é um ponto*", "*x é uma linha*", "*z passa através de x e y*", e "*x = y*", respectivamente. Traduzir a seguinte frase: Para todos dois pontos, existe uma e somente uma linha que passa através desses dois pontos.

4. Um grupo Abelian é um conjunto A com um operador binário + que tem certas propriedades. Faça P(x,y,z) e E(x,y) representar x + y = z e x = y, respectivamente. Expressar os seguintes axiomas para o grupo Abelian simbolicamente.

(a) Para todo x e y em A, existe um z em A tal que x + y = z (fechamento).
(b) Se x+y = z e x + y = w, então z = w (unicidade).
(c) (x + y) + z = x + (y + z) (associatividade).
(d) x + y = y + x (simetria).
(e) Para todo x e y em A, existe um z tal que x + z = y (solução correta).

Seção 3.2

5. Para a seguinte interpretação (D = {a,b}),

P(a,a)	P(a,b)	P(b,a)	P(b,b)
V	F	F	V

determinar a tabela verdade das seguintes fórmulas:

(a) (3x) (3y) P(x,y) (b) (3x) (3y) P(x,y)

(c) (3x) (3y) P(x,y)   (d) (3y) ~ P(a,y)
(e) (3x) (3y) (P(x,y) 1 P(y,x))   (f) (3x) P(x,x)

6. Considerar a seguinte fórmula:
A: (3x) P(x) 1 (3x) P(x).
(a) Provar que esta fórmula é sempre verdadeira se o domínio D contém somente um elemento.
(b) Seja D = {a,b}. Achar a interpretação sobre D na qual A possui valor F (falso).

7. Considerar a seguite interpretação:
Domínio: D = {1,2}.
Valor das constantes a e b:

a	b
1	2

valor para a função f:

f(1)	f(2)
2	1

valor para o predicado P:

P(1,1)	P(1,2)	P(2,1)	P(2,2)
V	V	F	F

Avaliar o valor lógico das seguintes fórmulas na interpretação acima:
(1) P(a, f(a)) 2 P(b, f(b))
(2) (3x) (3y) P(y,x)
(3) (3x) (3y) (P(x,y) 1 P(f(x), F(y))).

8. Sejam F_1 e F_2 as seguintes fórmulas:
F_1 : (3x) (P(x) 1 Q(x))
F_2: ~ Q(a).
Provar que ~ P(a) é uma conseqüência lógica de F_1 e F_2 .

Seção 3.3
9. Transformar as seguintes fórmulas na forma normal prenex:
(1) (3x) (P(x) 1 (3y) Q(x,y))
(2) (3x) (~ ((3y) P(x,y)) 1 ((3z) Q(z) 1 R(x)))
(3) (3x) (3y) ((3z) P (x,y,z) 2 ((3u) Q(x,u) 1 (3v) Q(y,v))).

Seção 3.4
10. Considerar as seguintes sentenças:
F_1 : Todo estudante é honesto.
F_2 : João não é honesto.
Para as sentenças acima, provar que João não é um estudante.

11. Considerar as seguintes premissas:
(1) Todo atleta é forte.
(2) Todo aquele que for inteligente e forte terá sucesso em sua carreira.
(3) Peter é um atleta.
(4) Peter é inteligente.

Tente concluir que Peter terá sucesso em sua carreira.

12. Assumir que São Francisco é amado por todos aqueles que amam alguém. Assumir também que não há ninguém que não ame ninguém. Deduza que São Francisco é amado por todos.

CAPÍTULO 4 - TEOREMA DE HERBRAND

4.1 INTRODUÇÃO

Nos capítulos anteriores, foi discutido como resolver problemas através da prova de teoremas. Neste e nos capítulos seguintes, serão estudados os procedimentos de prova. Encontrar um procedimento geral de decisão para verificar a validade (ou inconsistência) de uma fórmula estava sendo usado a muito tempo. A primeira tentativa foi feita por Leibniz (1646-1716) , novamente por Peano na virada do século e pela escola de Hilbert na década de 20. Aconteceu até que Church [1936] e Turing[1936] provassem que isso era impossível. Church e Turing independentemente mostraram que não existe um procedimento geral de decisão para verificar a validade de fórmulas da lógica de primeira ordem. No entanto, existem métodos de prova que podem verificar se uma fórmula é válida se realmente ela for válida. Para fórmulas inválidas, esses procedimentos em geral nunca terminarão. Na visão do resultado de Church e Turing, isso é o melhor que se pode ter para se obter um método de prova.

Uma abordagem muito importante para a prova mecânica de teoremas foi dada por Herbrand em 1930. Por definição, uma fórmula válida é uma fórmula que é verdadeira sob todas as suas interpretações. Herbrand desenvolveu um algoritmo para encontrar uma interpretação que pode invalidar uma fórmula dada. No entanto, se uma certa fórmula é realmente válida, nenhuma dessas interpretações podem existir e seu algoritmo termina após um número finito de tentativas. O método de Herbrand é a base para muitos métodos modernos de prova automática.

Gilmore [1960] foi uma das primeiras pessoas a implementar o método de Herbrand em um computador. Desde que uma fórmula é válida se e somente se sua negação for inconsistente, seu programa foi feito para encontrar a inconsistência da negação de uma fórmula dada. Durante a execução do programa, são geradas fórmulas proposicionais que são testadas sobre sua inconsistência. Se a negação de uma certa fórmula é inconsistente, o programa detecta esta situação. O programa de Gilmore foi usado para provar fórmulas bem simples, porém encontrou sérias dificuldades com muitas outras fórmulas da lógica de primeira ordem. Estudos minuciosos de seu algoritmo revelaram que seu método de testar a inconsistência de uma fórmula proposicional era ineficiente. O método de Gilmore foi melhorado por Davis e Putnam [1960], meses depois de seu resultado ser publicado. Entretanto, tal melhoria não foi suficiente. Muitas fórmula válidas da lógica de primeira ordem não poderiam ser provadas pelos computadores em um tempo razoável.

A maior modificação foi feita por Robinson [1965], que introduziu o tão famoso *princípio de resolução*. O método de Resolução é muito mais eficiente que os anteriores. Desde o início do princípio de resolução, muitas melhorias foram sugeridas para aumentar sua eficiência. Algumas dessas melhorias são: a resolução semântica, a resolução de bloqueio, a resolução linear, a resolução unitária (unit resolution) e a estratégia de conjunto de apoio (set-of-support strategy).

Neste capítulo, será primeiro provado o teorema de Herbrand. O princípio de resolução e suas melhorias serão vistos nos capítulos seguintes.

4.2 FORMAS PADRÃO DE SKOLEM

O método de Herbrand e o método de prova de resolução citados anteriormente são procedimentos de refutação. Isto é, ao invés de provar que uma fórmula é válida, prova-se que a negação de uma fórmula é inconsistente. Isso é uma questão de conveniência. Não existe nenhuma perda de generalidade em usar-se procedimentos de refutação. Entretanto, esses procedimentos são aplicados a uma "forma padrão" de uma fórmula. Esta forma padrão foi introduzida por Davis e Putnam e será usada ao longo deste texto. Essencialmente eles queriam expressar as seguintes idéias:

1) Uma fórmula da lógica de primeira ordem pode ser transformada em uma forma normal prenex onde a matriz não contém quantificadores e o prefixo é uma sequencia de quantificadores.
2) A matriz, a qual não contém quantificadores, pode ser transformada numa forma normal conjuntiva.
3) Sem afetar a propriedade de inconsistência, os quantificadores existenciais no prefixo podem ser eliminados usando-se funções Skolem.

No capítulo 3, já foi estudado como transformar uma fórmula na forma normal prenex. Pelas técnicas dadas no capítulo 2, pode-se saber como transformar a matriz em uma forma normal conjuntiva. Agora será visto como eliminar os quantificadores existenciais.

Seja uma fórmula F já na forma normal prenex (Q_1x_1) ... $(Q_nx_n)M$, onde M está expressa na forma normal conjuntiva. Suponha-se que Qr é um quantificador existencial no prefixo (Q_1x_1) ... (Q_nx_n), 1 1 r 1 n . Se nenhum quantificador universal aparece antes de Q_r, pode-se escolher uma nova constante *c* diferente das outras constantes que aparecem em M, substituir todos os x_r de M por c, e apagar (Q_rx_r) do prefixo.

Se Q_{s1}, ..., Q_{sm} são todos quantificadores universais que aparecem antes de Q_r, 1 1 s1 < s2 ... < sm < r, pode-se escolher um novo símbolo f n-arg diferente dos outros símbolos da função, substituir todos os xr em M por f(xs1, xs2, ..., xsm), e apagar (Q_rx_r) do prefixo. Após o processo anterior ser aplicado para todos os quantificadores existenciais no prefixo, a fórmula final obtida é uma "forma padrão Skolem" (ou somente forma padrão) da fórmula F. As constantes e funções usadas para substituir os quantificadores existenciais são chamados de funções Skolem.

Exemplo 4.1: Obter a forma padrão da fómula: $(\exists x)(\forall y)(\forall z)(\exists u)(\forall v)(\exists w)\ P(x, y, z, u, v, w)$.

Na fórmula acima, $(\exists x)$ não é precidido por nenhum quantificador universal, $(\exists u)$ é precedido por $(\forall y)$ e $(\forall z)$, e $(\exists w)$ por $(\forall y)$, $(\forall z)$ e $(\forall v)$. Portanto, substitui-se a variável existencial x por uma constante *a*, *u* por uma função de 2 variáveis f(y,z), e w por uma função de três parâmetros g(y, z, v). Então, obtém-se a seguinte forma padrão da fórmula:

$$(\forall y)(\forall z)(\forall v)\ P(a, y, z, f(y,z), v, g(y, z, v)).$$

Exemplo 4.2: Obter a forma padrão da fómula: $(\forall x)(\exists y)(\exists z)\ ((\sim P(x,y) \wedge Q(x,z)) \vee R(x,y,z))$.
Primeiro, a matriz é transformada em uma forma normal conjuntiva:

$$(\forall x)(\exists y)(\exists z)\ ((\sim P(x, y) \vee R(x, y, z)) \wedge (Q(x, z) \vee R(x, y, z))).$$

Então, desde que $(\exists y)$ e $(\exists z)$ são ambas precedidas por $(\forall x)$, as variáveis exitenciais y e z são substituídas, respectivamente, por funções de uma variável f(x) e g(x). Então, pode-se obter a seguinte forma padrão para a fórmula:

$$(\forall x)\ ((\sim P(x, f(x)) \vee R(x, f(x), g(x))) \wedge (Q(x, g(x)) \vee R(x, f(x), g(x)))).$$

Definição: Uma cláusula é uma disjunção de literais.

Quando for conveniente, pode-se ver um conjunto de literais como um sinônimo de uma cláusula. Por exemplo, $P \vee Q \vee \sim R = \{ P, Q, \sim R\}$. Uma cláusula composta de *r* literais é chamada uma cláusula *r-literal* . Uma cláusula 1-literal é chamada uma cláusula unitária. Quando uma cláusula não contém literais, pode-se chamar de cláusula *vazia*. Desde que uma cláusula vazia não possui literais os quais podem ser satisfeitas por uma interpretação, uma cláusula vazia é sempre falsa. Normalmente, representa-se cláusulas vazias pelo símbolo € .

As disjunções $\sim P(x, f(x)) \vee R(x, f(x), g(x))$ e $Q(x, g(x)) \vee R(x, f(x), g(x))$ na forma padrão no Exemplo 4.2 são cláusulas. *Um conjunto S de cláusulas é considerado como uma conjunção de todas as cláusulas em S, onde toda variável em S é considerada regida por um quantificador universal*. Por convenção, uma forma padrão pode ser simplesmente representada por um conjunto de cláusulas. Por exemplo, a forma padrão do exemplo 4.2 pode ser representada pelo conjunto :

$$\{ \sim P(x, f(x)) \vee R(x, f(x), g(x)),\quad Q(x, g(x)) \vee R(x, f(x), g(x)) \} .$$

Como foi citado na seção inicial, pode-se eliminar os quantificadores existenciais sem afetar a propriedade de inconsistência. Isso é mostrado no seguinte teorema.

Teorema 4.1: Seja S um conjunto de cláusulas que representa uma forma padrão de uma fórmula F. Então F é inconsistente se e somente se S é inconsistente.

Prova: Sem perder a generalidade, pode-se assumir que F está na forma normal prenex, isto é, $F = (Q_1x_1) \ldots (Q_nx_n)\ M\ [x_1, \ldots, x_n]$. (Usa-se $M\ [x_1, \ldots, x_n]$ para representar que a matriz M contém variáveis $x_1, \ldots, x_n$). Faça-se Q_r ser o primeiro quantificador existencial. Seja $F_1 = (\forall x_1) \ldots (\forall x_{r-1})\ (Q_{r+1}x_{r+1}) \ldots (Q_nx_n)\ M[x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n]$, onde *f* é uma função Skolem correspondente a x_r, 1 1 r 1 n . Quer-se mostrar que F é inconsistente se e somente se F_1 é inconsistente. Suponha-se que F é inconsistente. Se F_1 é consistente, então existe uma interpretação *I* tal que F_1 é verdadeira em *I* . Isto é, para todo $x_1, \ldots, x_{r-1}$ existe no mínimo um elemento, o qual é $f(x_1, \ldots, x_{r-1})$, tal que $(Q_{r+1}x_{r+1}) \ldots (Q_nx_n)\ M[x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n]$ é verdadeiro em *I*. Então, F é verdadeiro em *I*, o que contradiz a idéia de que F é inconsistente. Portanto F_1 deve ser inconsistente. Por outro lado, supõe-se que F_1 é inconsistente. Se F é consistente, então existe uma interpretação *I* sobre um domínio *D* tal que F é verdadeira em I. Isto é, para todo $x_1, \ldots, x_{r-1}$ existe um elemento x_r tal que $(Q_{r+1}x_{r+1}) \ldots (Q_nx_n)\ M[x_1, \ldots, x_{r-1}, x_r, x_{r+1}, \ldots, x_n]$ é verdadeira em *I*. Extende-se a interpretação *I* para incluir uma função f que mapeia $(x_1, \ldots, x_{r-1})$ para x_r para todo $x_1, \ldots, x_{r-1}$ em D, isto é, $f(x_1, \ldots, x_{r-1}) = x_r$. Seja essa extensão de I representada por I'. Então, claramente, para todo $x_1, \ldots, x_{r-1}$, $(Q_{r+1}x_{r+1}) \ldots (Q_nx_n)\ M[x_1, \ldots, x_{r-1}, f(x_1, \ldots, x_{r-1}), x_{r+1}, \ldots, x_n]$ é verdadeiro em I'. Ou seja, F_1 é verdadeira em I', o que contradiz a idéia de que F_1 é inconsistente. Portanto F deve ser inconsistente. Assume-se que existem *m* quantificadores existenciais em F. Seja F_k obtido de F_{k-1} pela substituiçãodo primeiro quantificador existencial em F_{k-1} por uma função Skolem, k = 1, ..., m. Claramente, $S = F_m$. Usando os mesmos argumentos dados acima, pode-se mostrar que F_{k-1} é inconsistente se e somente se F_k é inconsistente para k = 1, ..., m. Portanto, conclui-se que F é inconsistente se e somente se S é inconsistente.

Seja S uma forma padrão de uma fórmula F. Se F é inconsistente, então pelo Teorema 4.1, F = S. Se F não é inconsistente, nota-se que, em geral, F não é equivalente a S. Por exemplo, seja $F \triangleq (\exists x)\ P(x)$ e $S \triangleq P(a)$. Claramente, S é uma forma padrão de F. Entretanto, faça-se I ser a interpretação definida abaixo:

Domínio: D = {1, 2}.

Valores para *a*:

a

Valores para *P*:

	1
P(1)	P(2)
F	**V**

Então, claramente, F é verdadeiro em I, mas S é falso em I. Portanto $F \neq S$.

Nota-se que uma fórmula pode ter mais de uma forma padrão. Para maior simplicidade, quando se transforma uma fórmula F em uma forma padrão S, poder-se-ia substituir os quantificadores existenciais por funções Skolem as quais devem ser as mais simples possíveis. Isto é, poder-se-ia usar funções Skolem com o menor número de argumentos possível. Isso significa que deve-se mover os quantificadores existenciais o mais à esquerda possível. Além disso, se tem-se $F = F_1 \wedge ... \wedge F_n$, pode-se obter separadamente um conjunto S_i de cláusulas, onde cada S_i representa uma forma padrão de F_i, i = 1, ..., n. Então, seja $S = S_1 \cup ... \cup S_n$. Por argumentos semelhantes aos dados na prova do Teorema 4.1, não é difícil ver que F é inconsistente se e somente se S é inconsistente.

Exemplo 4.3: Neste exemplo, mostra-se como expressar o seguinte teorema em uma forma padrão: Se $x \cdot x = e$ para todo x no grupo G, onde $\cdot$ é um operador binário e e a sua identidade em G, então G é comutativo.

Primeiro simboliza-se o teorema acima junto com alguns axiomas básicos na teoria dos grupos e então representa-se a negação do teorema por um conjunto de clásulas.

A_1: $x, y \in G$ implica que $x \cdot y \in G$ (propriedade de fechamento);

A_2: $x, y, z \in G$ implica que $x \cdot (y \cdot z) = (x \cdot y) \cdot z$ (propriedade da associatividade);

A_3: $x \cdot e = e \cdot x = x$ para todo $x \in G$ (propriedade da identidade);

A_4: para todo $x \in G$ existe um elemento $x^{-1} \in G$ tal que $x \cdot x^{-1} = x^{-1} \cdot x = e$ (propriedade do inverso).

Seja $x \cdot y = z$ representado por P(x, y, z) e x^{-1} representado por i(x). Então os axiomas acima podem ser representados por:

A_1': $(\forall x)(\forall y)(\forall z)\ P(x, y, z)$

A_2': $(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)\ (P(x, y, u) \wedge P(y, z, v) \wedge P(u, z, w) \rightarrow P(x, v, w)) \wedge (\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)\ (P(x, y, u) \wedge P(y, z, v) \wedge P(x, v, w) \rightarrow P(u, z, w))$

A_3': $(\forall x)\ P(x, e, x) \wedge (\forall x)\ P(e, x, x)$

A_4': $(\forall x)\ P(x, i(x), e) \wedge (\forall x)\ P(i(x), x, e)$.

A conclusão do teorema é:

B: Se $x \cdot x = e$ para todo $x \in G$, então G é comutativo, ou seja, $u \cdot v = v \cdot u\ (\forall\, u, v) \in G$.

B pode ser representado por:

B': $(\forall x)\ P(x, x, e) \rightarrow ((\forall u)(\forall v)(\forall w)\ P(u, v, w) \rightarrow P(v, u, w)))$.

Agora, o teorema completo é representado pela fórmula $F = A_1' \wedge ... \wedge A_4' \rightarrow B'$. Então, $\sim F = A_1' \wedge A_2' \wedge A_3' \wedge A_4' \wedge \sim B'$. Para obter-se um conjunto S de cláusulas para ~ F, obtém-se um consjuto S_i de cláusulas para cada axioma A_i', i = 1, 2, 3, 4 , como consta abaixo:

S_1: $\{ P(x, y, f(x, y))\}$

S_2: $\{\sim P(x, y, u) \vee \sim P(y, z, v) \vee P(u, z\ w), \sim P(x, v, w) \vee \sim P(x, y ,u) \vee \sim P(y, z, v) \vee \sim P(x, v, w) \vee P(u, z, w)\}$

S_3: $\{P(x, e, x), P(e, x, x)\}$.

S_4: { P(x, i(x), e), P(i(x), x, e)}.

Sendo ~B' = ~((∀x) P(x, x, e) → ((∀u)(∀v)(∀w)(P(u, v, w) → P(v, u, w))))

= ~ (~(∀x) P(x, x, e) ∨ ((∀u)(∀v)(∀w)(~ P(u, v, w) ∨ P(v, u, w))))

= (∀x) P(x, x, e) ∧ ~ ((∀u)(∀v)(∀w) (~ P(u, v, w) ∨ P(v, u, w)))

= (∀x) P(x, x, e) ∧ (∃u)(∃v)(∃w) P(u, v, w) ∧ ~ P(v, u, w)),

o conjunto de cláusulas para ~ B' é mostrado abaixo.

T: {P(x, x, e),
P(a, b, c),
~P(b, a, c)}.

Então, o conjunto S = $S_1 \cup S_2 \cup S_3 \cup S_4 \cup T$ é o conjunto das seguintes cláusulas:

(1) P(x, y, f(x, y))
(2) ~ P(x, y, u) ∨ ~ P(y, z, v) ∨ ~ P(u, z, w) ∨ P(x, v, w)
(3) ~ P(x, y, u) ∨ ~ P(y, z, v) ∨ ~ P(x, v, w) ∨ P(u, z, w)
(4) P(x, e, x)
(5) P(e, x, x)
(6) P(x, i(x), e)
(7) P(i(x), x, e)
(8) P(x, x, e)
(9) P(a, b, c)
(10) ~P(b, a, c).

No exemplo 4.3, mostrou-se como obter um conjunto S de cláusulas para a fórmula ~F. Pelos teoremas 2.2 e 4.1, sabe-se que F é válida se e somente se S é inconsistente. Como foi dito no início desta seção, usam-se procedimentos de refutação para provar teoremas. Então, daqui para frente, assume-se que a entrada do procedimento de refutação é sempre um conjunto de cláusulas, tal como o conjunto S obtido no exemplo acima. Portanto, será usado "insatisfatível" ("satisfatível"), ao invés de "inconsistente" ("consistente"), para conjuntos de cláusulas.

4.3 O UNIVERSO DE HERBRAND DE UM CONJUNTO DE CLÁUSULAS

Por definição, um conjunto S de cláusulas é insatisfatível se e somente se ele falso sob todas as interpretações sobre todos os domínios. Uma vez que é inconveniente e impossível considerar todas as interpretações sobre todos os domínios, poderia ser interessante buscar-se um domínio especial H, tal que S é insatisfatível se e somente se S é falso sob todas as interpretações sobre este domínio. Felizmente, existe um domínio, que se chama de *universo de Herbrand* de S, definido como o seguinte:

Definição: Sendo H_o o conjunto de constantes que aparecem em S. Se nenhuma constante aparece em S, então H_o é formado por uma única constante, representado por H_o = {a}. Para i = 0, 1, 2, ...,H_{i+1} é a união de H_i com o conjunto de todos os termos da forma $f^n(t_1, ..., t_n)$ para todas as funções n-arg f^n que aparecem em S, onde t_j, j = 1, ..., n, são membros do conjunto H_i . Então cada H_i é chamado de conjunto de constantes i-nível de S, e H_1 é chamado de Universo de Herbrand de S.

Exemplo 4.4: Seja S = {P(a), ~P(x) ∨ P(f(x))}. Então:

H_o = {a}

H_1 = {a, f(a)}
H_2 = {a, f(a), f(f(a))}
.
.
.
H_1 = {a, f(a), f(f(a)), f(f(f(a))), ...}.

Exemplo 4.5: Seja S = { P(x) $\vee$ q(x), R(z), T(y) $\vee$ ~W(y) }. Uma vez que não existe nenhuma constante em S, H_o = { a }. Não existe nenhum símbolo de função em S , portanto H = H_o = H_1= ... = H_1 = { a }.

Exemplo 4.6: Seja S = { P(f(x), a, g(y), b) }. Então
H_o = { a, b}
H_1 = { a, b, f(a), f(b), g(a), g(b) }
H_2 = {a,b,f(a),f(b),g(a), g(b),f(f(a)),f(f(b)),f(g(a)),f(g(b)),g(f(a)),g(f(b)), g(g(a)), g(g(b))}
.
.
.

Na sequência, entende-se por expressão um termo, um conjunto de termos, um átomo, um conjunto de átomos, um literal, uma cláusula, ou um conjunto de cláusulas. Quando nenhuma variável aparecer na expressão, dá-se o nome de expressão base para enfatizar esse fato. Então é possível usar um termo base, um átomo base, um literal base, e um cláusula base significando que não aparece nenhuma variável na respectiva expressão. Portanto, uma *subexpressão* de uma expressão E é uma expressão que aparece em E.

Definição: Seja um conjunto de cláusulas. O conjunto de átomos base da forma P^n(t_1, ... ,t_n) para todos os predicados n-arg P^n existentes em S, onde t_1, ... ,t_n são elementos de um universo Herbrand de S, é chamado de *conjunto de átomos* ou *base Herbrand* de S.

Definição: Uma instância base de uma cláusula C de um conjunto S de cláusulas é uma cláusula obtida pela substituição das variáveis em C por membros do universo Herbrand de S.

Exemplo 4.7: Seja S = { P(x), Q(f(y)) $\vee$ R(y)}. C = P(x) é uma cláusula em S e H = {a, f(a), f(f(a)), ...} é o universo Herbrand de S. Então P(a) e P(f(f(a))) são ambas instâncias base de C.

Serão consideradas agora interpretações sobre o universo Herbrand. Seja S um conjunto de cláusulas. Como foi visto no Capítulo 3, uma interpretação sobre o universo Herbrand de S é uma atribuição de valores às constantes, funções e predicados que aparecem em S. Na sequência, define-se uma interpretação especial sobre o universo Herbrand de S, chamada de H-interpretação de S.

Definição: Seja S um conjunto de cláusulas; H, o universo Herbrand de S; e I, uma interpretação de S sobre H. I é chamado de uma H-interpretação de S se satisfaz as seguintes condições:
1. I mapeia todas as constantes em S para elas mesmas.
2. Seja f uma função n-arg e h_1, ..., h_n elementos de H. Em I, f é uma função que mapeia (h_1, ..., h_n) (um elemento em H^n) para f(h_1, ..., h_n) (um elemento em H).

Não existem restrições para atribuições a qualquer predicado n-arg em S. Seja A = $\{A_1, A_2, ..., A_n, ... \}$ um conjunto de átomos de S. Uma H-interpretação I pode ser convenientemente representada por um conjunto: I = $\{ m_1, m_2, ..., m_n, ... \}$ onde m_j é A_j ou $\sim A_j$ para j = 1, 2, ... O significado desse conjunto é que se m_j é A_j, então A_j é "verdadeiro"; do contrário, A_j é "falso".

Exemplo 4.8: Considere o conjunto S = { P(x) $\vee$ Q(x), R(f(y)) }. O universo Herbrand H de S é H = { a, f(a), f(f(a)), ... }. Existem três símbolos de predicados: P, Q e R. Portanto o conjunto de átomos de S é:

A = { P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), ... }.

Algumas H-interpretações de S são as seguintes :

I_1 = { P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), ... }.

I_2 = { ~P(a), ~Q(a), ~R(a), ~P(f(a)), ~Q(f(a)), ~R(f(a)), ... }.

I_3 = { P(a), Q(a), ~R(a), P(f(a)), Q(f(a)), ~R(f(a)), ... }.

Uma interpretação de um conjunto S de cláusulas não precisa necessariamente ser definido sobre o universo Herbrand de S. Então uma interpretação pode não ser uma H-interpretação. Por exemplo, seja S = { P(x), Q(y, f(y,a)) }. Se o domínio é D = {1, 2}, então a seguinte interpretação é uma interpretação de S.

D = { 1, 2}

a	f(1,1)	f(1,2)	f(2,1)	f(2,2)
2	1	2	2	1

P(1)	P(2)	Q(1,1)	Q(1,2)	Q(2,1)	Q(2,2)
V	F	F	V	F	V

Para uma interpretação tal como a apresentada acima, pode-se definir uma H-interpretação I^* correspondente a I. Usa-se o exemplo acima para ilustrar. Primeiro, acha-se o conjunto de átomos S:

A = { P(a), Q(a,a), P(f(a,a)), Q(a, f(a,a)), Q(f(a,a), a), Q(f(a,a), f(a,a), ... }.

Depois, avalia-se cada membro de A usando a tabela acima:

P(a) = P(2) = F

Q(a, a) = Q(2, 2) = T

P(f(a, a)) = P(f(2, 2) = P(1) = T

Q(a, f(a, a)) = Q(2, f(2, 2)) = Q(2, 1) = F

Q(f(a, a), a) = Q(f(2,2), 2) = Q(1, 2) = T

Q(f(a, a), f(a, a)) = Q(f(2, 2), f(2, 2)) = Q(1, 1) = F

.

.

.

Portanto, a H-interpretação I^* correspondente a I é

I^* = { ~P(a), Q(a,a), P(f(a,a), ~Q(a, f(a,a)), Q(f(a,a), a), ~Q(f(a, a), f(a,a)), ... }.

No caso de não existirem constantes em S, o elemento *a*, que é usado para iniciar o universo Herbrand de S, pode ser mapeado para qualquer elemento do domínio D. Neste caso, se existe mais de um elemento em D, então existe mais de uma H-interpretação correspondente a I. Por exemplo, seja S = { P(x), Q(y, f(y, z)) } e seja uma interpretação I para S como segue:

D = { 1, 2}.

f(1,1)	f(1,2)	f(2,1)	f(2,2)
1	2	2	1

P(1)	P(2)	Q(1,1)	Q(1,2)	Q(2,1)	Q(2,2)
V	F	F	V	F	V

Então as duas H-iterpretações correspondentes a I são:
I^* = { P(a), ~Q(a,a), P(f(a,a), ~Q(a, f(a,a)), ~Q(f(a,a), a), ~Q(f(a, a), f(a,a)), ...}, se a= 1.
I^* = { ~P(a), Q(a,a), P(f(a,a), ~Q(a, f(a,a)), Q(f(a,a), a), ~Q(f(a, a), f(a,a)), ... }, se a = 2.

Pode-se formalizar os conceitos vistos acima da seguinte forma:

Definição: Dada uma interpretação I sobre um domínio D, uma H-interpretação I^* correspondente a I é uma H-interpretação que satisfaz a seguinte condição:

Seja h_1, ..., h_n elementos de H (o universo Herbrand de S). Sendo cada h_i mapeado para algum d_i em D. Se é atribuído a P(d_1, ... , d_n) V(F) por I, então para P(h_1, ... , h_n) também é atribuído V(F) em I^*.

A prova do seguinte lema não é difícil e fica como exercício.

Lema 4.1: Se uma interpretação I sobre algum domínio D satisfaz um conjunto de cláusulas S, então qualquer uma das H-interpretações I^* correspondentes a I também satisfaz S.

Teorema 4.2: Um conjunto S de cláusulas é insatisfatível se e somente se S é falso sob todas as H-interpretações de S.
<u>Prova</u>: (IDA) A primeira metade do teorema acima é óbvia uma vez que, por definição, S é insatisfatível se e somente se S é falso sob todas as interpretações sobre qualquer domínio.

(VOLTA) Para provar a segunda metade do teorema acima, assume-se que S é falso sob todas as H-interpretações de S. Suponha-se que S é não insastisfatível. Então existe uma interpretação I sobre algum domínio D tal que S é verdadeira sob I. Seja I^* uma H-interpretação correspondente a I. De acordo com o Lema 4.1, S é verdadeira sob I^*. Isso contradiz a afirmação de que S é falsa sob todas as H-interpretações de S. Portanto, S deve ser insatisfatível.

Então chega-se ao objetivo do início desta seção. Isto é, precisa-se considerar somente interpretações sobre o universo Herbrand, ou mais estritamente, H-interpretações, para verificar que um conjunto de cláusulas é ou não insatisfatível. Devido Teorema 4.2, daqui para diante sempre que for mencionada uma interpretação estar-se-á falando de uma H-interpretação.

Sendo $\varnothing$ representando o conjunto vazio. Cada uma das seguintes observações é óbvia. As provas serão deixadas como exercício.

1. Uma instância base C' de uma cláusula C é satisfeita por uma interpretação I se e somente se existe um literal base L' que também está em I, isto é $C' \cap I \neq \varnothing$.

2. Uma cláusula C é satisfeita por uma interpretação I se e somente se toda instância base de C é satisfeita por I.
3. Uma cláusula C é falsificada por uma interpretação I se e somente se existe no mínimo uma instância base C' de C tal que C' não é satisfeita por I.
4. Um conjunto de cláusulas S é insatisfatível se e somente se para toda interpretaçãoI, existe no mínimo uma instância ground C' de alguma cláusula C em S tal que C' não é satisfeita por I.

Exemplo 4.9: 1. Considere a cláusula C = $\sim P(x) \vee Q(f(x))$. Sejam I_1, I_2, e I_3 definidas como segue:

$I_1 = \{ \sim P(a), \sim Q(a), \sim P(f(a)), \sim Q(f(a)), \sim P(f(f(a))), \sim Q(f(f(a))), ... \}$

$I_2 = \{ P(a), Q(a), P(f(a)), Q(f(a)), P(f(f(a))), Q(f(f(a))), ... \}$

$I_3 = \{ P(a), \sim Q(a), P(f(a)), \sim Q(f(a)), P(f(f(a))), \sim Q(f(f(a))), ... \}$

Nota-se que C é satisfeita por I_1 e I_2, mas invalidada por I_3.

2. Considere S = $\{ P(x), \sim P(a)\}$. Existe somente duas H-interpretações:

$I_1 = \{P(a)\}$ e $I_2 = \{ \sim P(a)\}$.

S é invalidada por ambas as H-interpretações, e portanto é insatisfatível.

4.4 ÁRVORES SEMÂNTICAS

Tendo-se introduzido o Universo Herbrand, será considerado agora o estudo de árvores semânticas. Será visto na seqüência que encontrar uma prova para um conjunto de cláusulas é o mesmo que gerar uma árvore semântica.

Definição: Se A é um átomo, então os dois literais A e ~A são ditos um complemento do outro, e o conjunto {A, ~A} é chamado de *par complementar*.

Uma cláusula é uma tautologia se ela contém um par complementar. Quando é usado o termo *tautologia*, faz-se referência específica a uma cláusula que é uma tautologia.

Definição: Dado um conjunto S de cláusulas, seja A o conjunto atômico de S. Uma *árvore semântica* para S é uma árvore (descendente) T, onde cada ligação é vinculada a um conjunto finito de átomos ou negações de átomos de A de forma que:

i. Para cada nó N, há somente um número finito de sucessores imediatos $L_1, ..., L_n$ de N. Seja Q_i a conjunção de todos os literais no conjunto vinculado a L_i, i=1,...,n. Então $Q_1 \vee Q_2 \vee ... \vee Q_n$ é uma fórmula proposicional válida.
ii. Para cada nó N, seja I(N) a união de todos os conjuntos vinculados a ligações deste ramo de T desde o nó raiz até N, inclusive. Então I(N) não contém nenhum par complementar.

Definição: Seja A = $\{A_1, A_2, ..., A_k, ...\}$ um conjunto de átomos de um conjunto S de cláusulas. Uma árvore semântica para S é dita ser *completa* se e somente se para cada nó terminal (ou folha) N de uma árvore semântica, isto é, um nó que não tem sucessores, I(N) contém ou A_i ou $\sim A_i$ para i = 1, 2, ...

Exemplo 4.10: Seja A = {P, Q, R} um conjunto de átomos de um conjunto S de cláusulas. Então cada uma das duas árvores na Fig. 4.1 é uma árvore semântica completa para S.

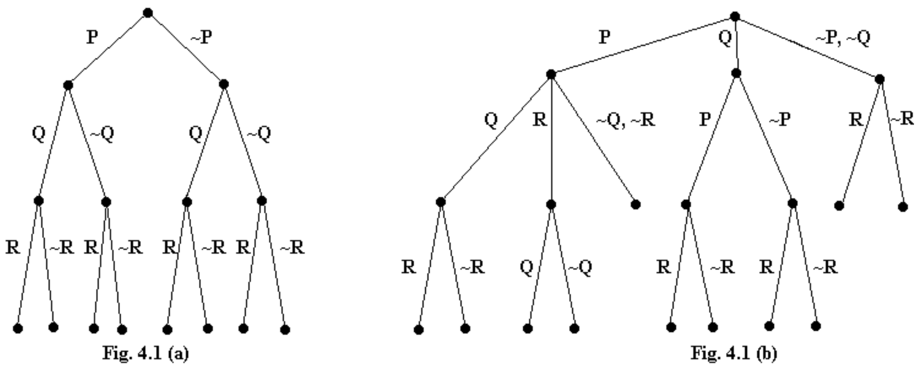

Figura 4.1 Árvores Semânticas completas para um conjunto de átomos A = {P, Q, R}

Exemplo 4.11: Considere S = {P(x), P(a)}. O conjunto de átomos de S é {P(a)}. Uma árvore semântica completa para S é mostrada na Fig. 4.2.

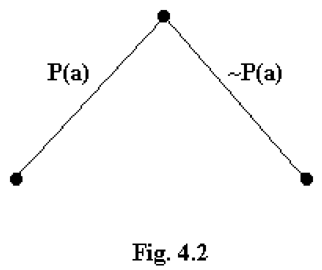

Figura 4.2 Árvore Semântica completa para um conjunto de átomos A = {P(a)}

Exemplo 4.12: Considere S = {P(x), Q(*f*(x))}. O conjunto de átomos de S é:

A = {P(a), Q(a), P(*f*(a)), Q(*f*(a)), P(*f*(*f*(a))), Q(*f*(*f*(a))), ...}.

A figura 4.3 mostra uma árvore semântica para S.

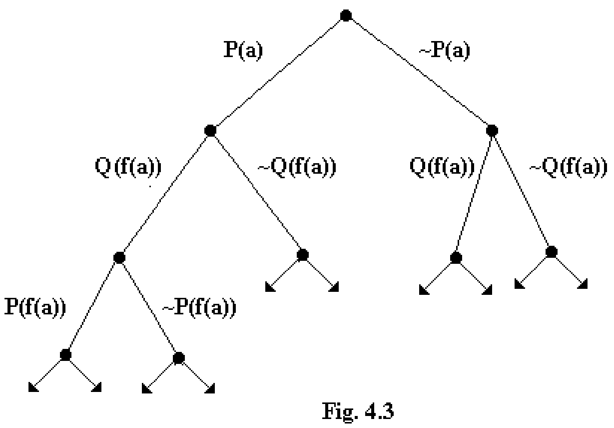

Figura 4.2 Árvore Semântica para um conjunto de átomos A = {P(a), Q(a), P(*f*(a)), Q(*f*(a)), P(*f*(*f*(a))), Q(*f*(*f*(a))), ...}

Nota-se que para cada nó N na árvore semântica para S, I(N) é um subconjunto de alguma interpretação para S. Por esta razão, I(N) será chamada *interpretação parcial* para S.

Quando um conjunto de átomos de um conjunto S de cláusulas é infinito, qualquer árvore semântica completa para S será infinita. Como é facilmente visto, uma árvore semântica completa para S corresponde a uma avaliação exaustiva de todas as possíveis interpretações para S. Se S é insatisfatível, então S não é verdadeiro em cada uma dessas interpretações. Dessa forma, pode-se terminar de expandir os nós a partir de um nó N se I(N) falsifica S. Isso motiva as seguintes definições:

Definição: Um nó N é um *nó falha* se I(N) falsifica alguma instâncias base de uma cláusula em S, mas I(N') não falsifica nenhuma instância base de uma cláusula em S para cada nó antecessor N' de N.

Definição: Uma árvore semântica T é dita ser *fechada* se e somente se cada subárvore de T termina em um nó falha.

Definição: Um nó N de uma árvore semântica fechada é chamada de *nó de inferência* se todos os sucessores imediatos de N são nós falha.

Exemplo 4.13: Seja S = {P, Q $\vee$ R, ~P $\vee$ ~Q, ~P $\vee$ ~R}. O conjunto de átomos de S é A = {P, Q, R}. Figura 4.4a é uma árvore semântica completa para S, enquanto fig. 4.4b é uma árvore semântica fechada para S.

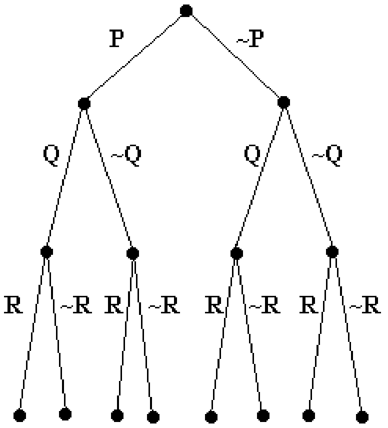

Fig. 4.4 (a)

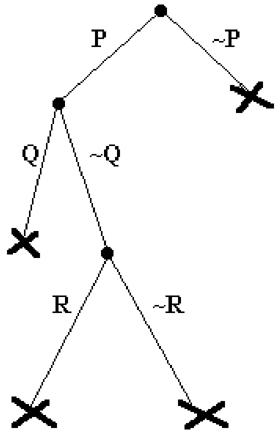

Fig. 4.4 (b)

Figura 4.4 Árvore semântica completa (a) e fechada (b) para S.

Exemplo 4.14: Considere S = {P(x), ~P(x) $\vee$ Q(*f*(x)), ~Q(*f*(a))}. O conjunto de átomos de S é igual a: A = {P(a), Q(a), P(*f*(a)), Q(*f*(a)), ...}. A figura 4.5 mostra uma árvore semântica fechada para S.

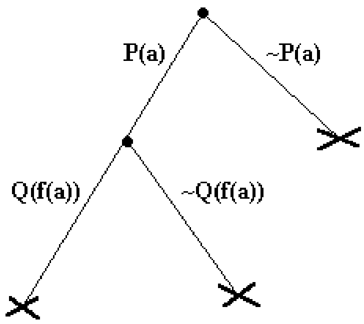

Fig. 4.5

Figura 4.5 Árvore semântica fechada para S = {P(x), ~P(x) $\vee$ Q(*f*(x)), ~Q(*f*(a))}.

4.5 TEOREMA DE HERBRAND

O teorema de Herbrand é muito importante na lógica simbólica; ele é a base para a maioria dos procedimentos atuais para prova automática de teoremas. O teorema Herbrand está bastante relacionado com o Teorema 4.2. Dessa forma, para testar se um conjunto S de cláusulas é insatisfatível, precisa-se considerar somente interpretações sobre o universo Herbrand de S. Se S é falso sob todas as interpretações do universo Herbrand de S, então pode-se concluir que S é insatisfatível. Uma vez que geralmente existem muitas, possivelmente um número infinito, dessas interpretações, há necessidade de organizá-las de alguma forma sistemática. Isto pode ser feito usando uma árvore semântica. Serão dadas duas versões do teorema Herbrand.

Teorema 4.3 (Teorema Herbrand, versão I): Um conjunto S de cláusulas é insatisfatível se e somente se correspondendo a toda árvore semântica completa de S, há uma árvore semântica fechada finita.

Prova: (IDA) Suponha S ser insatisfatível. Seja T uma árvore semântica completa para S. Para cada ramo B de T, seja I_B um conjunto de todos os literais ligados a todos os nós do ramo B. Então I_B é uma interpretação para S. Desde que S seja insatisfatória, I_B precisa falsificar uma instância base C' de uma cláusula C em S. Contudo, desde que C' é finito, deve haver um nó falha N_B (que está a um número finito de ligações do nó raiz) no ramo B. Uma vez que todo ramo de T tem um nó falha, há uma árvore semântica fechada T' para S. Além disso, uma vez que somente um número finito de ligações estão conectadas em cada nó de T', T' precisa ser finito (isto é, o número de nós em T' é finito), de outra maneira, pelo Lema de Konig, pode-se encontrar um ramo infinito contendo nenhum nó falha. Dessa forma, completa-se a primeira metade do teorema.

(VOLTA) De modo inverso se, correspondendo a toda árvore semântica completa T para S, há uma árvore finita semântica fechada, então cada ramo de T contém um nó falha. Isto significa que toda interpretação falsifica S. Conseqüentemente S é insatisfatível. Isto completa a prova da segunda parte do teorema.

Teorema 4.4 (Teorema de Herbrand, Versão II): Um conjunto de cláusulas é insatisfatível se e somente se há um conjunto finito insatisfatível S' de instâncias base de cláusulas de S.

Prova: (IDA) Suponha S ser insatisfatível. Seja T uma árvore semântica completa para S. Então, pelo teorema de Herbrand (Versão I), há uma árvore semântica fechada finita T' correspondendo a T. Seja S' um conjunto de todas as instâncias base das cláusulas que estão falsificadas em todos os nós falha de T'. S' é finito desde que haja um número finito de nós falhos em T'. Uma vez que S' é falso em toda interpretação de S', S' é insatisfatível.

(VOLTA) Suponha que há um conjunto finito insatisfatível S' de instâncias base das cláusulas em S. Desde que cada interpretação I de S contém uma interpretação I' de S', se I' falsifica S', então I deve também falsificar S'. Entretanto, S' é falsificado por toda a interpretação I'. Conseqüentemente, S' é falsificado por toda interpretação I de S. Portanto, S é falsificado por toda interpretação de S. Então, S é insatisfatível.

Exemplo 4.15: Seja S = {P(x), ~P(*f*(a))}. Este conjunto S é insatisfatível. Portanto, pelo teorema de Herbrand há um conjunto insatisfatível finito S' de instâncias base de cláusulas de S. Um desses conjuntos é: S' = {P(*f*(a)), ~P(*f*(a))}.

Exemplo 4.16: Seja S = {~P(x) $\vee$ Q(*f*(x),x), P(*g*(b)), ~Q(y,z)}. Esse conjunto é insatisfatível. Um de seus conjuntos insatisfatívels de instâncias baixas em S é:

S' = {~P(*g*(b) $\vee$ Q(*f*(g(b)), *g*(b)), P(*g*(b)), ~Q(*f*(*g*(b)), *g*(b))}.

Exemplo 4.17: Seja um conjunto S consistindo das seguintes cláusulas:

S = { ~P(x,y,u) $\vee$ ~P(y,z,v) $\vee$ ~P(x,v,w) $\vee$ P(u,z,w),
~P(x,y,u) $\vee$ ~P(y,z,v) $\vee$ ~P(u,z,w) $\vee$ P(x,v,w),
P(*g*(x,y), x, y), P(x, *h*(x, y), y), P(x, y, *f*(x,y)),
~P(*k*(x), x, *k*(x)) }.

Esse conjunto S também é insatisfatível. Entretanto, não é muito fácil encontrar manualmente um conjunto insatisfatível finito S' de instâncias base das cláusulas em S. Um meio de encontrar tal conjunto S' é gerar uma árvore semântica fechada T' para S. Então o conjunto S' de todas as instâncias base falsificadas em todos os nós falhos de T' é o conjunto desejado. O conjunto seguinte é o que se deseja como S'. Pode-se querer checar que cada cláusula base em S' é uma instância base de alguma cláusula em S, e que S' é insatisfatível.

S' = { P(a, *h*(a, a), a),
~P(*k*(*h*(a, a), *h*(a,a), *k*(*h*(a, a))),
P(*g*(a, *k*(*h*(a, a))), a, *k*(*h*(a, a))),
~P(*g*(a, *k*(*h*(a, a))), a, *k*(*h*(a, a))) ∨ ~P(a, *h*(a, a), a) ∨
~P(*g*(a, *k*(*h*(a, a))), a, *k*(*h*(a, a))) ∨ ~P(*k*(*h*(a, a), *h*(a,a), *k*(*h*(a, a))) }.

4.6 IMPLEMENTAÇÃO DO TEOREMA HERBRAND

A segunda versão do teorema de Herbrand sugere um procedimento de refutação. Isto é, dado um conjunto insatisfatível S de cláusulas para provar, se há um procedimento mecânico que pode sucessivamente gerar os conjuntos S_1', ..., S_n', ... de instâncias base de cláusulas em S e sucessivamente testar se eles são insatisfatíveis, então como garantido pelo Teorema de Herbrand, este procedimento pode detectar um número finito N tal que S_N' é insatisfatível.

Gilmore em 1960 desenvolveu um programa de computador que gerava, sucessivamente, os conjuntos S_0', S_1', ..., onde S_i' é o conjunto de todos as instâncias base obtidas pela troca das variáveis em S por constantes no conjunto de constantes i-nível H_i de S. Uma vez que cada S_i' é uma conjunção de cláusulas base, pode-se usar qualquer método disponível na lógica proposicional para checar sua insatisfabilidade. Gilmore usou o método multiplicativo. Isto é, quando cada S_i' é produzido, S_i' é transformado em uma forma normal disjuntiva. Qualquer conjunção na forma normal disjuntiva que contém um par complementar é removida. Se algum conjunto S_i' for vazio, então S_i' é insatisfatível e a prova é encontrada.

Examplo 4.18: Considere S = {P(x), ~P(a)}.
H_0 = {a}
S_0' = P(a) ∧ ~P(a) = €
Dessa forma S é provado ser insatisfatível.

Exemplo 4.19: Considere S = {P(a), ~P(x) ∧ Q(*f*(x)), ~Q(*f*(a))}.
H_0 = {a}.
S_0' = P(a) ∧ (~P(a) ∨ Q(*f*(a))) ∧ ~Q(*f*(a))
= (P(a) ∧ ~P(a) ∧ ~Q(*f*(a))) ∨ (P(a) ∧ Q(*f*(a)) ∧ ~Q(*f*(a)))
= € ∨ € = €
Dessa forma está provado que S é insatisfatível.

O método multiplicativo usado por Gilmore é ineficiente como é facilmente visto. Por exemplo, para um pequeno conjunto de 10 cláusulas base com dois literais, há 2^{10} conjunções. Para evitar essa ineficiência, Davis e Putnam introduziram um método mais eficiente para testar se um conjunto de cláusulas baixas é insatisfatível.

Método de Davis e Putnam: Seja S um conjunto de cláusulas base. Essencialmente, o método consiste das quatro regras a seguir:

I. *Regra Tautológica*: Retirar todas as cláusulas base de S que são tautológicas. O restante do conjunto S' é insatisfatível se e somente se S o for.

II. *Regra do Literal-Único*: Se há uma cláusula unitária base L em S, obter S' de S retirando todas aquelas cláusulas base em S contendo L. Se S' é vazia, então S é satisfatível. Caso contrário, obter um conjunto S" de S' pela retirada de ~L de S'. S" é insatisfatível se e somente se S o é. Note que se ~L é uma cláusula unitária base, então a cláusula torna-se € quando ~L é retirado da cláusula.

III. *Regra do Literal-Puro*: Um literal L em uma cláusula base S é dito ser *puro* em S se e somente se o literal ~L não aparecer em nenhuma cláusula base em S. Se um literal L é puro em S, retirar todas as cláusulas base contendo L. O restante do conjunto S' é insatisfatível se e somente se S o é.

IV. *Regra da Separação*: Se um conjunto S pode ser colocado na forma $(A_1 \vee L) \wedge ... \wedge (A_m \vee L) \wedge (B_1 \vee \sim L) \wedge ... \wedge (B_n \vee \sim L) \wedge R$, onde A_i, B_i, e R são livres de L e ~L, então obter os conjuntos $S_1 = A_1 \wedge ... \wedge A_m \wedge R$ e $S_2 = B_1 \wedge ... \wedge B_n \wedge R$. S é insatisfatível se e somente se $(S_1 \vee S_2)$ é insatisfatível, isto é, ambos S_1 e S_2 são insatisfatíveis.

Se um conjunto S original é insatisfatível, então o conjunto restante depois de aplicada uma das regras continua ainda insatisfatível, e vice-versa.

Regra I: Desde que uma tautologia é satisfatível para toda interpretação, S' é insatisfatória se e somente S o é.

Regra II: Se S' é vazio, então todas as instâncias base em S contém L. Conseqüentemente qualquer interpretação contendo L pode satisfazer S. Portanto S é satisfatível. Ainda tem que se mostrar que S" é insatisfatível se e somente se S é insatisfatível. Supondo que S" é insatisfatível. Se S é satisfatível, então há um modelo M de S contendo L. Para S", M precisa satisfazer todas as cláusulas que não contém L. Além disso, uma vez que M falsifica ~L, M precisa satisfazer todas as cláusulas que originalmente contém ~L. Portanto, M deve satisfazer S". Isto contradiz a hipótese de que S" é insatisfatível. Conseqüentemente S deve ser insatisfatível. De modo inverso, supondo S ser insatisfatível. Se S" é satisfatível, então há um modelo M" de S". Desse modo, qualquer interpretação de S contendo M" e L deve ser um modelo de S. Isso contradiz a hipótese que S não tem modelo. Conseqüentemente S" deve ser insatisfatível. Portanto, S" é insatisfatível se e somente se S o é.

Regra III: Supondo S' ser insatisfatível. Então S deve ser insatisfatível uma vez que S' é um subconjunto de S. De modo inverso, supondo que S é insatisfatível. Se S' é satisfatível, então há um modelo M de S'. Uma vez que nem L, nem ~L está em S', nem L, nem ~L está em M. Assim qualquer interpretação de S que contém M e L é um modelo de S. Isto contradiz a hipótese de que S não tem modelo. Conseqüentemente S' deve ser insatisfatível. Portanto, S' é insatisfatível se e somente se S é insatisfatível.

Regra IV: Supondo que S é insatisfatível. Se $(S_1 \vee S_2)$ é satisfatível, então S_1 ou S_2 tem um modelo. Se $S_1(S_2)$ tem um modelo M, então qualquer interpretação de S contendo ~L(L) é um modelo de S. Isto contradiz a hipótese de que S não tem modelo. Conseqüentemente $(S_1 \vee S_2)$ é insatisfatível. De modo inverso, supondo que $(S_1 \vee S_2)$ é insatisfatível. Se S é satisfatível, S precisa ter um modelo M. Se M contém ~L(L), M pode satisfazer $S_1(S_2)$. Isto contradiz a hipótese de que $(S_1 \vee S_2)$ é insatisfatível. Conseqüentemente S precisa ser insatisfatível. Portanto, S é insatisfatível se e somente se $(S_1 \vee S_2)$ o é.

As regras acima são muito importantes. Nos capítulos seguintes será visto que as essas regras tem muitas extensões.

Exemplo 4.20: Mostre que $S = (P \vee Q \vee {\sim}R) \wedge (P \vee {\sim}Q) \wedge {\sim}P \wedge R \wedge U$ é insatisfatível.
(1) $(P \vee Q \vee {\sim}R) \wedge (P \vee {\sim}Q) \wedge {\sim}P \wedge R \wedge U$
(2) $(Q \vee {\sim}R) \wedge ({\sim}Q) \wedge R \wedge U$ Regra II em $\sim P$
(3) ${\sim}R \wedge R \wedge U$ Regra II em $\sim Q$
(4) € $\wedge\, U$ Regra II em $\sim R$.
Uma vez que a última fórmula contém a cláusula vazia, S é insatisfatível.

Exemplo 4.21: Mostre que $S = (P \vee Q) \wedge {\sim}Q \wedge ({\sim}P \vee Q \vee {\sim}R)$ é satisfatível.
(1) $(P \vee Q) \wedge {\sim}Q \wedge ({\sim}P \vee Q \vee {\sim}R)$
(2) $P \wedge ({\sim}P \vee {\sim}R)$ Regra II em $\sim Q$
(3) $\sim R$ Regra II em P
(4) $\varnothing$ Regra II em $\sim R$.
Sendo último conjunto vazio, então S é satisfatível.

Exemplo 4.22: Mostre que $S = (P \vee {\sim}Q) \wedge ({\sim}P \vee Q) \wedge (Q \vee {\sim}R) \wedge ({\sim}Q \vee {\sim}R)$ é satisfatível.
(1) $(P \vee {\sim}Q) \wedge ({\sim}P \vee Q) \wedge (Q \vee {\sim}R) \wedge ({\sim}Q \vee {\sim}R)$
(2) $({\sim}Q \wedge (Q \vee {\sim}R) \wedge ({\sim}Q \vee {\sim}R))$
$\vee\, (Q \wedge (Q \vee {\sim}R) \wedge ({\sim}Q \vee {\sim}R))$ Regra IV em P
(3) ${\sim}R \vee {\sim}R$ Regra II em $\sim Q$ e Q
(4) $\varnothing \vee \varnothing$ Regra II em $\sim R$.
Uma vez que ambos os conjuntos que foram separados são satisfatíveis, então S é satisfatível.

Exemplo 4.23: Mostre que $S = (P \vee Q) \wedge (P \vee {\sim}Q) \wedge (R \vee Q) \wedge (R \vee {\sim}Q)$ é satisfatível.
(1) $(P \vee Q) \wedge (P \vee {\sim}Q) \wedge (R \vee Q) \wedge (R \vee {\sim}Q)$
(2) $(R \vee Q) \wedge (R \vee {\sim}Q)$ Regra III em P
(3) $\varnothing$ Regra III em R.
Logo, S é satisfatível.

O método acima para testar a insatisfabilidade (inconsistência) é mais eficiente que o método multiplicativo. Esse método pode ser aplicado para qualquer fórmula na lógica proposicional. Para tanto é necessário primeiro transformar a fórmula proposicional dada na forma normal conjuntiva, e então aplicar as quatro regras acima.

Para praticar a aplicação deste método utilize-o para resolver o Exemplo 2.13 do Capítulo 2.

4.7 EXERCÍCIOS

Seção 4.2

1. Encontre a forma padrão para cada uma das seguintes fórmulas:
   (a) ${\sim}((\forall x)\, P(x) \rightarrow (\exists y)(\forall z)\, Q(y,z))$
   (b) $(\forall x)(({\sim}E(x,0) \rightarrow ((\exists y)(E(y, g(x)) \wedge (\forall z)(E(z, g(x)) \rightarrow (E(y,z)))))$
   (c) ${\sim}((\forall x)P(x) \rightarrow (\exists y)P(y))$.
2. Suponha que $(\exists x)(\forall y)\, M[x,y]$ é uma forma normal prenex de uma fórmula F, onde $M[x,y]$ é a matriz que contém somente variáveis x e y. Seja f uma função que não ocorre em $M[x,y]$. Prove que F é válida se e somente se $(\exists x)M[x,f(x)]$ é válida.
3. Sendo S1 e S2 formas padrão das fórmulas F1 e F2, respectivamente. Se S1 = S2, é verdadeiro que F1 = F2? Explique.

4. Considere as seguintes proposições:

F_1: Todo aquele que guarda dinheiro, ganha juros.

F_2: Se não há juros, então ninguém guarda dinheiro.

Seja S(x,y), M(x), I(x) e E(x,y) representando, respectivamente, "x guarda y", "x é dinheiro", "x é juros" e "x ganha y".

(1) Simbolize F_1 e F_2.

(2) Encontre as formas normais de F_1 e $\sim F_2$.

Seção 4.3

5. Seja S = {P(*f*(x), a, *g*(*f*(x), b))}.

(1) Encontre H_0 e H_1.

(2) Encontre todas as instâncias base de S sobre H_0.

(3) Encontre todas as instâncias base de S sobre H_1.

6. Prove o Lema 4.1 desse capítulo.

7. Seja F uma fórmula. Seja S uma forma padrão de ~F. Encontre a condição necessária e suficiente para F de modo que o universo Herbrand de S seja finito.

8. Considere a seguinte cláusula C e a interpretação I:

C: $P(x) \vee Q(x,f(x))$

I: {~P(a), ~P(*f*(a)), ~P(*f*(*f*(a))), ...,
~Q(a, a), Q(a, *f*(a)), ~Q(a, *f*(*f*(a))), ...,
~Q(*f*(a), a), Q(*f*(a), *f*(a)), ~Q(*f*(a), *f*(*f*(a))), ...}.

A interpretação I satisfaz C?

9. Considere o seguinte conjunto S de cláusulas: S = {P(x), Q(f(y))}.
Seja uma interpretação I definida como abaixo:

I = {P(a), P(*f*(a)), P(*f*(*f*(a))), ...,
Q(a), ~Q(*f*(a)), Q(*f*(*f*(a))), ...}.

A interpretação I satisfaz S?

10. Considere S = {P(x), ~P(*f*(y))}.

1. Encontre H_0, H_1, H_2 e H_3 de S.
2. É possível encontrar uma interpretação que satisfaça S? Se possível, dê uma. Se não, por quê?

Seção 4.4

11. Seja S = {P, $\sim P \vee Q$, ~Q}. Dê uma árvore semântica fechada de S.

12. Considere S = {P(x), $\sim P(x) \vee Q(x,a)$, ~Q(y,a)}.

(a) Encontre o conjunto de átomos de S.

(b) Encontre uma árvore semântica completa de S.

(c) Encontre uma árvore semântica fechada de S.

Seção 4.5

13. Considere S = {P(x, a, *g*(x, b)), ~P(*f*(y), z, *g*(*f*(a), b))}. Encontre um conjunto insatisfatível S' de instâncias base de cláusulas em S.

14. Seja S = {P(x), $Q(x, f(x)) \vee \sim P(x)$, ~Q(*g*(y), z)}. Encontre um conjunto insatisfatível S' de instâncias base de cláusulas em S.

Seção 4.6

15. Use as regras sugeridas por Davis e Putnam para provar que as seguintes fórmulas são insatisfatíveis.

(1) $(\sim P \vee Q) \wedge \sim Q \wedge P$
(2) $(P \vee Q) \wedge (R \vee Q) \wedge \sim R \wedge \sim Q$
(3) $(P \vee Q) \wedge (\sim P \vee Q) \wedge (\sim R \vee \sim Q) \wedge (R \vee \sim Q)$.

16. Use as regras sugeridas por Davis e Putnam para provar que as seguintes fórmulas são satisfatórias.

(1) $P \wedge Q \wedge R$
(2) $(P \vee Q) \wedge (\sim P \vee Q) \wedge R$
(3) $(P \vee Q) \wedge \sim Q$.

CAPÍTULO 5 - PRINCÍPIO DA RESOLUÇÃO

5.1. INTRODUÇÃO

Nos capítulos anteriores foi considerado o teorema de Herbrand. Baseado nesse teorema foi apresentado um procedimento de refutação. Entretanto o procedimento de Herbrand tem uma grande desvantagem: ele precisa gerar os conjuntos S_1', S_2', ... de instâncias base das cláusulas. Para muitos casos, esta seqüência cresce exponencialmente. Para ver isso será dado um pequeno exemplo:

Considere: S = {P(*x*, *g*(*x*), *y*, *h*(*x*, *y*), *z*, *k*(*x*, *y*, *z*)), ~P(*u*, *v*, *e*(*v*), *w*, *f*(*v*, *w*), *x*)}.

Sendo: $H_0 = \{a\}$,

$H_1 = \{a, g(a), h(a, a), k(a, a, a), e(a), f(a, a)\}$

.

.

.

o número de elementos em S_0', S_1', ... é 2, 1512, ..., respectivamente. É interessante notar que o primeiro conjunto que é insatisfatível é S_5'. Entratanto, H_5' tem por volta de 10^{64} elementos. Conseqüentemente, S_5' tem 10^{256} elementos. Com a tecnologia computacional atual já é impraticável armazenar S_5' no computador, que dirá testar se é insatisfatível.

Para evitar a geração de conjuntos de instâncias base como requerido no procedimento de Herbrand, há necessidade de introduzir o princípio de resolução proposto em 1965 por Robinson. Ele pode ser aplicado diretamente a qualquer conjunto S de cláusulas (não necessariamente cláusulas base) para testar se S é insatisfatível.

A idéia principal do princípio de resolução é checar se S contém uma cláusula vazia €. Se S contém €, então S é insatisfatível. Se S não contém €, a próxima coisa a checar é se € pode ser derivado de S. Mais adiante ficará claro que, pelo teorema de Herbrand (versão I), checar pela presença de €, é equivalente a contar o número de nós de uma árvore semântica fechada para S. Pelo Teorema 4.3, S é insatisfatível se e somente se há uma árvore semântica fechada finita T para S. De maneira objetiva, S contém €, se e somente se T consiste de somente um nó - o nó raiz. Se S não contém €, T deve conter mais que um nó. Entretanto, pode ser que se possa reduzir o número de nós em T para um, eventualmente € pode ser forçado a aparecer. Isto é o que o princípio da resolução procura fazer. De fato, pode-se ver o princípio da resolução como uma regra de inferência que pode ser usada para gerar novas cláusulas a partir de S. Ao se colocar estas novas cláusulas em S, alguns nós da árvore original T podem ser forçados a se tornarem nós falha. Dessa forma, o número de nós em T pode ser reduzido e uma cláusula vazia € pode ser eventualmente derivada.

Neste capítulo será considerado primeiramente o princípio de resolução para a lógica proposicional. Em seguida será estendido para a lógica de predicados.

5.2 O PRINCÍPIO DE RESOLUÇÃO PARA A LÓGICA PROPOSICIONAL

O princípio de resolução é essencialmente uma extensão da *regra do um-literal* de Davis e Putnam, dada na seção 4.6 do capítulo 4. Considerando as seguintes cláusulas:

C_1: P

C_2: ~P $\vee$ Q.

Usando a regra um-literal, de C1 e C2 pode-se obter a cláusula

C_3:   Q.

O que a regra do um-literal requer é que primeiro se examine se há um par complementar de um literal (por exemplo, P) em C_1 e um literal (por exemplo, ~P) em C_2, e então se retire esse par de C_1 e C_2 para obter a cláusula C_3, a qual é Q.

Estendendo a regra acima e aplicando-a a qualquer par de cláusulas (não necessariamente cláusulas unitárias), tem-se a seguinte regra, a qual é chamada de *princípio de resolução: Para qualquer duas cláusulas C_1 e C_2, se há um literal L_1 em C_1 que é complementar a um literal L_2 em C_2, então retire L_1 e L_2 de C_1 e C_2 respectivamente, e construa a disjunção das cláusulas restantes. A cláusula construída dessa forma é um resolvente de C_1 e C_2.*

Exemplo 5.1: Considerar as seguintes cláusulas:
C_1: $P \vee R$
C_2: $\sim P \vee Q$.
A cláusula C_1 tem um literal P, o qual é complementar a ~P em C_2. Portanto, retirando P e ~P de C_1 e C_2, respectivamente, e construindo a disjunção das cláusulas restantes R e Q, obtém-se o resolvente $R \vee Q$.

Exemplo 5.2: Considerar as cláusulas:
C_1: $\sim P \vee Q \vee R$
C_2: $\sim Q \vee S$.
O resolvente de C_1 e C_2 é $\sim P \vee R \vee S$.

Exemplo 5.3: Considerar as cláusulas:
C_1: $\sim P \vee Q$
C_2: $\sim P \vee R$.
Desde que não há nenhum literal em C_1 que é complementar a nenhum literal em C_2, então não há resolvente para C_1 e C_2.

Uma importante propriedade de um resolvente é que qualquer resolvente de duas cláusulas C_1 e C_2 é uma implicação lógica de C_1 e C_2. Isso é mostrado no seguinte teorema.

Teorema 5.1   Dadas duas cláusulas C_1 e C_2, um resolvente C de C_1 e C_2 é uma implicação lógica de C_1 e C_2.

<u>**Prova:**</u> Sejam C_1, C_2 e C representados como segue: C_1 = L 2 C_1', C_2 = L 2 C_2' e C = C_1' 2 C_2', onde C_1' e C_2' são disjunções de literais. Suponha que C_1 e C_2 são verdadeiros em uma interpretação I. Quer-se provar que o resolvente C de C_1 e C_2 também é verdadeiro em I. Para provar isto, note que ou L ou ~L é falso em I. Assuma que l é falso em I. Então C_1 não deve ser uma cláusula unitária, senão C_1 seria falso em I. Portanto C_1' deve ser verdadeiro em I. Deste modo, o resolvente C, que é C_1' 2 C_2', deve ser verdadeiro em I. Da mesma maneira, pode-se mostrar que se ~L é falso em I, então C_2', deve ser verdadeiro em I. Portanto, C_1' 2 C_2', deve ser verdadeiro em I.

Tendo duas cláusulas unitárias, então o resolvente delas, se existir um, é uma cláusula vazia €. Mais importante ainda, se o conjunto S de cláusulas é insatisfatível, pode-se usar o princípio de resolução para gerar € a partir de S. Esse resultado será apresentado como teorema na Seção 5.6 deste capítulo. Enquanto isso será definida a noção de dedução.

Definição: Dado um conjunto S de cláusulas, a (resolução) *dedução* de C a partir de S é uma seqüência finita C_1, C_2, ..., C_k de cláusulas tal que cada C_i ou é uma cláusula em S ou é um resolvente de cláusulas precedente C_i e C_k = C. Uma dedução de € de S é chamada de *refutação* ou *prova* de S.

Diz-se que uma cláusula C pode ser *deduzida* ou *derivada* de S se há um dedução de C a partir de S. Serão apresentados vários exemplos que mostrarão como o princípio de resolução pode ser usado para provar se um conjunto de cláusulas é insatisfatível.

Exemplo 5.4: Considerar o conjunto:
(1) $\sim P \vee Q$
(2) $\sim Q$
(3) P
Ou seja, S = $\{\sim P \vee Q, \sim Q, P\}$.
A partir de (1) e (2), obtém-se um resolvente
(4) $\sim P$.
A partir de (4) e (3), obtém-se o resolvente
(5) € .
Como € é derivado de S por resolução então, de acordo com o Teorema 5.1, € é uma implicação lógica de S. Entretanto, € somente pode ser somente uma implicação lógica de um conjunto de cláusulas insatisfatível. Conseqüentemente, S é insatisfatível.

Exemplo 5.5: Para o conjunto S = $\{P \vee Q, \sim P \vee Q, P \vee \sim Q, \sim P \vee \sim Q\}$ tem-se:
(1) $P \vee Q$
(2) $\sim P \vee Q$
(3) $P \vee \sim Q$
(4) $\sim P \vee \sim Q$
A partir do conjunto acima geram-se os seguintes resolventes:
(5) Q a partir de (1) e (2)
(6) $\sim Q$ a partir de (3) e (4)
(7) € a partir de (5) e (6).
Como € é derivado, S é insatisfatível. A dedução acima pode ser representada pela árvore da figura 5.1, a qual é chamada de *árvore de dedução*.

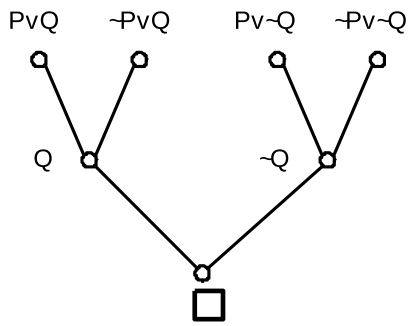

Figura 5.1 Árvore de Dedução para S = {P $\vee$ Q, ~P $\vee$ Q, P $\vee$ ~Q, ~P $\vee$ ~Q}

O princípio de resolução é uma regra de inferência bastante poderosa. Na seqüência será definido este princípio no contexto da lógica de predicados. Será mostrado também que o princípio de resolução é completo para provar se um conjunto de cláusulas é insatisfatível. Isto é, dado um conjunto S de cláusulas, S é insatisfatível se e somente se há uma resolução que produz uma cláusula vazia € a partir de S. Na seção 5.7, será aplicado o princípio de resolução para vários exemplos no qual pode se ver a utilidade desta regra de inferência.

5.3 SUBSTITUIÇÃO E UNIFICAÇÃO

Na seção anterior foi considerado o princípio de resolução para a lógica proposicional. A partir de agora será estendido para a lógica de predicados. Na seção 5.2 foi mostrado que a parte mais importante da aplicação do princípio de resolução é encontrar um literal em uma cláusula que é complementar a um literal em outra cláusula. Para cláusulas que não contém variáveis, isso é simples. Entretanto, para cláusulas contendo variáveis, isso é mais complicado. Por exemplo, considerando as seguintes cláusulas;

C_1: $P(x) \vee Q(x)$

C_2: $\sim P(f(x)) \vee R(x)$.

Não há literais em C_1 que sejam complementares a qualquer literal em C_2. Entretanto, se substituir-se *x* por *f*(*a*) em C_1 e *x* por *a* em C_2, obtém-se:

C_1': $P(f(a)) \vee Q(f(a))$

C_2': $\sim P(f(a)) \vee R(a)$.

Como se sabe que C_1' e C_2' são instâncias base de C_1 e C_2 respectivamente, e $P(f(a))$ e $\sim P(f(a))$ são complementares um com o outro. Conseqüentemente a partir de C_1' e C_2' obtém-se um resolvente

C_3': $Q(f(a)) \vee R(a)$.

Generalizando, se substituir *x* por *f*(*x*) em C_1, obtém-se:

C_1^*: $P(f(x)) \vee Q(f(x))$.

Novamente, C_1^* é uma *instância* de C_1. Agora o literal $P(f(x))$ em C_1^* é complementar ao literal $\sim P(f(x))$ em C_2. Portanto, pode-se obter um resolvente de C_1^* e C_2,

C_3: $Q(f(x)) \vee R(x)$.

C_3' é uma instância da cláusula C_3. Substituindo-se apropriadamente as variáveis em C_1 e C_2 como acima, pode-se gerar novas cláusulas a partir de C_1 e C_2. Além disso, a cláusula C_3 é a *cláusula mais geral* no sentido de que todas as outras cláusulas que podem ser geradas pelos processos acima são instâncias de C_3. C_3 será chamado de resolvente de C_1 e C_2. Na seqüência será considerado como gerar resolventes a partir de cláusulas (possivelmente contendo variáveis). Como a obtenção de resolventes a partir de cláusulas freqüentemente requer substituições para as variáveis, serão dadas as seguintes definições.

Definição: Uma *substituição* é um conjunto finito da forma de $\{t_1/v_1, ..., t_n/v_n\}$, onde cada v_i é uma variável, todo t_i é um termo diferente de v_i e quaisquer dois elementos no conjunto não tem as mesmas variáveis depois do símbolo a substituir (/). Quando $t_1, ..., t_n$ são termos base, a substituição é chamada de *substituição base*. A substituição sem elementos é chamada de *substituição vazia* e é denotada por ε. Serão usadas letras gregas para representar substituições.

Exemplo 5.6: Os dois conjuntos a seguir são substituições:{$f(z)/x$, y/z}, {a/x, $g(y)/y$, $f(g(b))/z$}.

Definição: Seja θ = {t_1/v_1, ..., t_n/v_n} uma substituição e E uma expressão. Então Eθ é uma expressão obtida a partir de E pela substituição simultânea de cada ocorrência da variável v_i, 1 1 i 1 n, em E pelo termo t_i. Eθ é chamado de uma *instância* de E. (Nota-se que a definição de uma instância é compatível com aquela da instância base de uma cláusula, definida no Capítulo 4.)

Exemplo 5.7: Seja θ = {a/x, $f(b)/y$, c/z} e E = P(x,y,z). Então Eθ = P(a, $f(b)$, c).

Definição: Sejam θ = {t_1/x_1,..., t_n/x_n} e λ = {u_1/y_1,..., u_m/y_m} duas substituições. Então a *composição* de θ e λ é a substituição, denotada por θ º λ, que é obtida a partir do conjunto:

$$\{t_1\lambda/x_1, ..., t_n\lambda/x_n, u_1/y_1, ..., u_m/y_m\}$$

pela retirada de todo elemento $t_j\lambda/x_j$ para o qual $t_j\lambda = x_j$ e todo elemento u_i/y_i tal que y_i está entre {x_1, x_2, ..., x_n}.

Exemplo 5.8: Sejam:

$$\theta = \{t_1/x_1, t_2/x_2\} = \{f(y)/x, z/y\}$$
$$\lambda = \{u_1/y_1, u_2/y_2, u_3/y_3\} = \{a/x, b/y, y/z\}, \text{ então:}$$
$$\{t_1\lambda/x_1, t_2\lambda/x_2, u_1/y_1, u_2/y_2, u_3/y_3\} = \{f(\mathrm{b})/x, y/y, a/x, b/y, y/z\}.$$

Visto que $t_2\lambda = x_2$,$t_2\lambda/x_2$, isto é, y/y deve ser retirado do conjunto. Ainda, como y_1 e y_2 estão entre {x_1, x_2, x_3}, u_1/y_1 e u_2/y_2, isto é, a/x e b/y devem ser retirados. Dessa forma obtém-se:

$$\theta \circ \lambda = \{f(b)/x, y/z\}.$$

Deve notar-se que a composição de substituições é associativa, e que a substituição vazia (ε) é uma identidade tanto a esquerda como a direita. Isto é, (θ ° λ)º μ = θ º (λ º μ) e ε º θ = θ º ε para todo θ, λ e μ.

No procedimento de prova por resolução, para identificar um par complementar de literais, necessita-se muito freqüentemente unificar (casar) duas ou mais expressões. Isto é, tem-se que encontrar uma substituição que possa fazer muitas expressões idênticas. Portanto, será considerada a unificação de expressões.

Definição: Uma substituição θ é chamada de *unificador* para um conjunto de expressões {E_1, ..., E_k} se e somente se $E_1\theta = E_2\theta = ... = E_k\theta$. O conjunto {$E_1$, ..., E_k} é dito ser *unificável* se há um unificador para o mesmo.

Definição: Um unificador σ para um conjunto de expressões {E_1, ..., E_k} é o *unificador mais geral* se e somente se para cada unificador θ para o conjunto há uma substituição λ tal que θ = σ º λ.

Exemplo 5.9: O conjunto {P(a,y), P(x, $f(b)$)} é unificável desde que a substituição θ = {a/x, $f(b)/y$} seja um *unificador* para o conjunto.

5.4 ALGORITMO DE UNIFICAÇÃO

Nesta seção será visto o algoritmo de unificação que será usado para encontrar o unificador mais geral para um conjunto unificável finito de expressões não vazias. Quando o conjunto não é unificável, o algoritmo também detectará esse fato.

Considere P(*a*) e P(*x*). Essas duas expressões não são idênticas: a diferença é que *a* ocorre em P(*a*) porém *x* ocorre em P(*x*). Para unificar P(*a*) e P(*x*) , primeiro tem-se que encontrar a diferença, e então tentar eliminá-la. Para P(*a*) e P(*x*), a diferença é {*a*, *x*}. Como *x* é uma variável, *x* pode ser substituído por *a*, e então a diferença será eliminada. Esta é a idéia na qual o algoritmo de unificação se baseia.

Definição: O conjunto diferença de um conjunto de expressões não vazio W é obtido pela localização do primeiro símbolo (contando a partir da esquerda) para o qual nem todas as expressões em W tem exatamente o mesmo símbolo e então extrair para cada expressão em W a subexpressão que começa com o símbolo ocupando aquela posição. O conjunto dessas respectivas subexpressões é o *conjunto diferença* de W.

Exemplo 5.10: Se W é {P(*x*, $\underline{f(y,z)}$), P(*x*, $\underline{a}$), P(*x*, $\underline{g(h(k(x)))}$)}, então a primeira posição do símbolo no qual nem todos os átomos em W são exatamente iguais é o quinto, visto que todos tem os primeiros quatro símbolos P(*x*, em comum. Dessa forma, o conjunto diferença consiste das respectivas subexpressões (termos sublinhados) que começam na posição do símbolo de número quinto. O conjunto é então {*f*(*y*, *z*), *a*, *g*(*h*(*k*(*x*)))}.

Algoritmo de Unificação

Passo 1: Faça $k = 0$, $W_k = W$ e $\sigma_k = \varepsilon$.

Passo 2: Se W_k é composto pelo mesmo elemento (singleton), então pare; σ_k é o unificador mais geral para W. Caso contrário, encontre o conjunto diferença D_k de W_k.

Passo 3: Se há elementos v_k e t_k em D_k tais que v_k é uma variável que não ocorre em t_k, vá para o passo 4. Caso contrário, pare; W não é unificável.

Passo 4: Faça $\sigma_{k+1} = \sigma_k\{t_k/v_k\}$ e $W_{k+1} = W_k\{t_k/v_k\}$. (Note que $W_{k+1} = W\sigma_{k+1}$.)

Passo 5: Faça $k = k + 1$ e vá para o passo 2.

Exemplo 5.11: Encontrar o unificador mais geral para W = {P(*a*, *x*, *f*(*g*(*y*))), P(*z*, *f*(*z*), *f*(*u*))}.

1. $\sigma_0 = \varepsilon$ e $W_0 = W$. Como W_0 não é composto por elementos idênticos, σ_0 não o unificador mais geral para W.

2. O conjunto diferença $D_0 = \{a, z\}$. Em D_0, há uma variável $v_0 = z$ que não ocorre em $t_0 = a$.

3. Sejam:
$$\sigma_1 = \sigma_0 \circ \{t_0/v_0\} = \varepsilon \circ \{a/z\} = \{a/z\}.$$
$$\begin{aligned} W_1 &= W_0\,\{t_0/v_0\} \\ &= \{P(a, x, f(g(y))), P(z, f(z), f(u))\}\,\{a/z\} \\ &= \{P(a, x, f(g(y))), P(a, f(a), f(u))\}. \end{aligned}$$

4. W_1 não é composto por elementos idênticos, conseqüentemente acha-se o conjunto diferença D_1 de W_1: $D_1 = \{x, f(a)\}$.

5. A partir de D_1 encontra-se que $v_1 = x$ e $t_1 = f(a)$.

6. Sejam:
$$\sigma_2 = \sigma_1 \circ \{t_1/v_1\} = (a/z\} \circ \{f(a)/x\} = \{a/z, f(a)/x\}$$
$$\begin{aligned} W_2 &= W_1\,\{t_1/v_1\} \\ &= \{P(a, x, f(g(y))), P(a, f(a), f(u))\}\,\{f(a)/x\} \\ &= \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}. \end{aligned}$$

7. W_2 não é composto por elementos idênticos, conseqüentemente encontra-se o conjunto diferença D_2 de W_2: $D_2 = \{g(y), u\}$.

8. A partir de D_2 acha-se que $v_2 = u$ e $t_2 = g(y)$.

9. Sejam:

$$\sigma_3 = \sigma_2 \circ \{t_2/v_2\} = \{a/z, f(a)/x\} \circ \{g(y)/u\} = \{a/z, f(a)/x, g(y)/u\}$$

$$\begin{aligned} W_3 &= W_2\ \{t_2/v_2\} \\ &= \{P(a, f(a), f(g(y))), P(a, f(a), f(u))\}\ \{g(y)/u\} \\ &= \{P(a, f(a), f(g(y))), P(a, f(a), f(g(y)))\} \\ &= \{P(a, f(a), f(g(y)))\}. \end{aligned}$$

10. Como W_3 é composto por elementos idênticos, $\sigma_3 = \{a/z, f(a)/x, g(y)/u\}$ é o unificador mais geral para W.

Exemplo 5.12: Determinar se o conjunto W = {Q($f(a)$, $g(x)$), Q(y, y)) é ou não unificável.

1. Sejam $\sigma_0 = \varepsilon$ e $W_0 = W$.

2. W_0 não é composto por elementos idênticos, conseqüentemente encontra-se o conjunto diferença D_0 de W_0: $D_0 = \{f(a), y\}$.

3. A partir de D_0 sabe-se que $v_0 = y$ e $t_0 = f(a)$.

4. Sejam:

$$\sigma_1 = \sigma_0 \circ \{t_0/v_0\} = \varepsilon \circ \{f(a)/y\} = \{f(a)/y\}.$$

$$\begin{aligned} W_1 &= W_0\ \{t_0/v_0\} = \{Q(f(a), g(x)), Q(y, y)\}\ \{f(a)/y\} \\ &= \{Q(f(a), g(x)), Q(f(a), f(a))\}. \end{aligned}$$

5. W_1 não é um composto por elementos idênticos, conseqüentemente encontra-se o conjunto diferença D_1 de W_1: $D_1 = \{g(x), f(a)\}$.

6. Entretanto, nenhum elemento de D_1 é uma variável. Conseqüentemente o algoritmo de unificação é terminado e conclui-se que W não é unificável.

Deve ser notado que o algoritmo de unificação acima sempre terminará para qualquer conjunto finito não vazio de expressões, caso contrário deveria ser gerada uma seqüência infinita $W\sigma_0$, $W\sigma_1$, $W\sigma_2$, ... de conjuntos finitos não vazios de expressões com a propriedade de que cada conjunto sucessivo contém uma variável a menos que o seu predecessor (a saber, $W\sigma_k$ contém v_k mas $W\sigma_{k+1}$ já não o contém). Isso se torna impossível, pois W contém um número finito de variáveis distintas.

Foi indicado anteriormente que se W é unificável, o algoritmo de unificação sempre encontrará o unificador mais geral para W. Isso é provado no teorema seguinte.

Teorema 5.2 (Teorema de Unificação): Se W é um conjunto unificável finito não vazio de expressões, então o algoritmo de unificação sempre terminará no passo 2 e o último σ_k é o unificador mais geral para W.

<u>Prova</u>: Uma vez que W é unificável, sendo 1 um unificador para W. Para k = 0, 1, ..., mostra-se que existe uma substituição 1_k tal que: $1 = 1_k\ 1\ 1_k$, por indução em k. Para k = 0, faz-se $1_0 = 1$. Então, $1 = 1_0\ 1\ 1_0$, uma vez que $1_0 = 1$. Suponha que $1 = 1_k\ 1\ 1_k$ ainda é válido para 0 1 k 1 n. Se $W1_n$ é composto somente por elementos idênticos, então o algoritmo de unificação termina no Passo 2. Uma vez que $1 = 1_n\ 1\ 1_n$, 1_n é o unificador mais geral para W. Se $W1_n$ não é composto somente por elementos idênticos, o algoritmo de unificação irá encontrar um conjunto diferença D_n de $W1_n$. Sendo $1 = 1_n\ 1\ 1_n$ um unificador de W, 1_n deve unificar D_n . Entretanto, uma vez que D_n é um conjunto diferença, deve existir uma variável em D_n . Sendo t_n qualquer outro elemento diferente de v_n . Então, uma vez que 1_n unifica D_n , $v_n 1_n = t_n 1_n$. Agora, se v_n ocorre em t_n , então $v_n 1_n$ ocorre em $t_n 1_n$. Entreatnto, isto é impossível, uma vez que v_n e t_n são distintos e $v_n 1_n = t_n 1_n$. Consequentemente, v_n não ocorre em t_n . Portanto, o algoritmo de unificação não terminará no Passo 3, mas irá para o Passo 4 para o conjunto $1_{n+1} = 1_n\ 1\ \{ t_n/v_n\}$. Sendo $1_{n+1} = 1_n - \{t_n 1_n/v_n\}$. Então, uma vez que v_n não ocorre em t_n , $t_n 1_{n+1} = t_n (1_n - \{t_n 1_n/v_n\}) = t_n 1_n$. Assim, tem-se:

$$
\begin{aligned}
\{ t_n/v_n\}\ 1\ 1_{n+1} &= \{t_n 1_{n+1}/v_n\}\ 2\ 1_{n+1} \\
&= \{t_n 1_n/v_n\}\ 2\ 1_{n+1} \\
&= \{t_n 1_n/v_n\}\ 2\ (1_n - \{t_n 1_n/v_n\}) \\
&= 1_n
\end{aligned}
$$

Isto é, $1_n = \{ t_n/v_n\}\ 1\ 1_{n+1}$. Portanto, $1 = 1_n\ 1\ 1_n = 1_n\ 1\ \{ t_n/v_n\}\ 1\ 1_{n+1} = 1_{n+1}\ 1\ 1_{n+1}$. Então, para todo k 1 0, existe uma substituição 1_k tal que $1 = 1_k\ 1\ 1_k$. Uma vez que o algoritmo de unificação deve terminar e não termina no Passo 3, ele deve terminar no Passo 2. Além disso, uma vez que $1 = 1_k\ 1\ 1_k$, para todo k, o último 1_k é o unificador mais geral para W.

5.5 O PRINCÍPIO DE RESOLUÇÃO DA LÓGICA DE PREDICADOS

Tendo introduzido o algoritmo de unificação na seção anterior, pode-se agora estudar o princípio de resolução para a lógica de predicados.

Definição: Se dois ou mais literais (com o mesmo sinal) de uma cláusula *C* têm o mesmo unificador geral 1, então *C1* é chamado *fator* de *C*. Se *C1* é uma cláusula unitária, então ela é denominada um fator unitário de *C*.

Exemplo 5.13: Seja C = <u>P(x)</u> 2 <u>P(f(y))</u> 2 ~Q(x). Então o primeiro e o segundo literais (sublinhados) tem o mesmo unificador geral 1 = { f(y) / x }. Portanto, *C1* = P(f(y)) 2 ~Q(F(y)) é um fator de C.

Definição: Sejam C_1 e C_2 duas cláusulas (chamadas cláusulas pai) com nenhuma variável em comum. Seja L_1 e L_2, dois literais em C_1 e C_2, respectivamente. Se L_1 e ~ L_2 têm o mesmo unificador geral 1 então a cláusula ($C_1 1$ - $L_1 1$) 2 ($C_2 1$ - $L_2 1$) é chamada *resolvente binário* de C_1 e C_2. Os literais L_1 e L_2 são chamados de *literais resolvidos.*

Exemplo 5.14: Seja C_1 = P(x) 2 Q(x) e C_2 = ~P(a) 2 R(x). Desde que x aparece tanto em C_1 quanto em C_2, pode-se renomear as variáveis em C_2 e fazer C_2 = ~P(a) 2 R(y). Escolhe-se L_1 = P(x) e L_2 = ~P(a). Desde que ~L_2 = P(a), L_1 e ~L_2 têm o unificador mais geral 1 = {a/x}. Portanto,

($C_1 1$ - $L_1 1$) 2 ($C_2 1$ - $L_2 1$) = ({P(a), Q(a)} - {P(a)}) 2 ({~P(a), R(y)} - {~P(a)})
= { Q(a) } 2 { R(y) } = { Q(a), R(y) } = Q(a) 2 R(y).

Então Q(a) 2 R(y) é um resolvente binário de C_1 e C_2. P(x) e ~P(x) são os literais resolvidos.

Definição: Um *resolvente* de cláusulas (pais) C_1 e C_2 é um dos seguintes resolventes binários:

1. um resolvente binário de C_1 e C_2,

2. um resolvente binário de C_1 e um fator de C_2,
3. um resolvente binário de um fator de C_1 e de C_2,
4. um resolvente binário de um fator de C_1 e um fator de C_2 .

Exemplo 5.15: Seja C_1 = P(x) 2 P(f(y)) 2 R(g(y)) e C_2 = ~P(f(g(a))) 2 Q(b). Um fator de C_1 e C_1’ = P(f(y)) 2 R(g(y)). Um resolvente binário de C_1’ e C_2 é R(g(g(a))) 2 Q(b). Portanto, R(g(g(a))) 2 Q(b) é um resolvente binário de C_1 e C_2.

O *princípio de resolução*, ou simplesmente *resolução*, é uma regra de inferência que gera *resolventes* a partir de um conjunto de cláusulas. Esta regra foi introduzida por Robinson em 1965. Ela é mais eficiente que os métodos anteriores de prova, que eram implementação direta do Teorema de Herbrand usados por Gilmore, Davis e Putnam. Além disso, a resolução é *completa*. Isto é, ela sempre gera uma cláusula vazia □ de um conjunto insatisfatível de cláusulas. Antes de provar esta declaração, será mostrado um exemplo na geometria plana.

Exemplo 5.16: Mostrar que os ângulos internos opostos formados pela diagonal de um trapezóide são iguais.

Para provar este teorema, primeiro se formula os seus axiomas. Seja T(x,y,u,v) a representação de que xyuv formam um trapezóide com o vértice superior esquerdo x, o vértice superior direito y, o vértice inferior direito u, e o vértice inferior esquerdo v; faça P(x,y,u,v) representar que o segmento de reta xy é paralelo ao segmento uv; e seja E(x,y,z,u,v,w) a representação de que o ângulo xyz é igual ao ângulo uvw. Então tem-se os seguintes axiomas:

A_1 : (3x) (3y) (3u) (3v) [T(x,y,u,v) 1 P(x,y,u,v)]   definição de um trapezóide

A_2 : (3x) (3y) (3u) (3v) [P(x,y,u,v) 1 E(x,y,v,u,v,y)]   ângulos internos opostos de retas paralelas são iguais , ver figura 5.2.

A_3 : T(a,b,c,d)

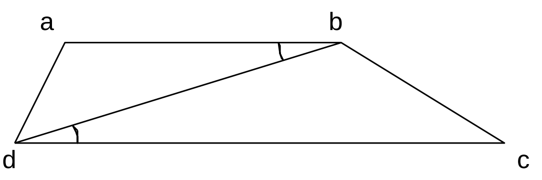

Figura 5.2 Trapezóide

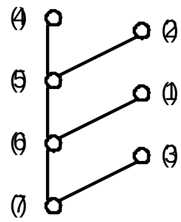

Figura 5.3 Árvore de Dedução

A partir desses aximas , deve-se estar apto a concluir que E(a,b,d,c,d,b) é verdadeiro, isto é: A_1 2 A_2 2 A_3 1 E(a,b,d,c,d,b) é uma fórmula válida. Desde que se quer provar isso por refutação, nega-se a conclusão e prova-se que: A_1 2 A_2 2 A_3 2 ~E(a,b,d,c,d,b) é insatisfatível. Para fazer isso, transforma-se a fórmula em uma forma padrão como segue:

S = { ~T(x,y,u,v) 2 P(x,y,u,v), ~P(x,y,u,v) 2 E(x,y,v,u,v,y), T(a,b,c,d), ~E(a,b,d,c,d,b) }.

A forma padrão S acima é um conjunto de quatro cláusulas. Agora será mostrado que o conjunto S é insatisfatível pela resolução:

(1)	~T(x,y,u,v) 2 P(x,y,u,v)	uma cláusula em S
(2)	~P(x,u,u,v) 2 E(x,y,v,u,v,y)	uma cláusula em S
(3)	T(a,b,c,d)	uma cláusula em S
(4)	~E(a,b,d,c,d,b)	uma cláusula em S
(5)	~P(a,b,c,d)	um resolvente de (2) e (4)

(6) ~T(a,b,c,d) um resolvente de (1) e (5)
(7) ☐ um resolvente de (3) e (6).

uma vez que que a última cláusula é a cláusula vazia que é derivada de S, pode-se concluir que S é insatisfatível. A prova acima pode ser facilmente representada por uma árvore, mostrada na figura 5.3.

A árvore da figura 5.3 é chamada de árvore de dedução. Isto é, uma árvore de dedução de um conjunto de cláusulas S é uma árvore (upward), na qual cada nó inicial contém uma cláusula em S e cada nó não inicial contém um resolvente das cláusulas contidas nos seus nós predecessores imediatos. Chama-se de árvore de dedução uma *árvore de dedução de R* se *R* é uma cláusula ligada ao nó raiz da árvore de dedução. Uma vez que a árvore de dedução é meramente uma árvore de representação de uma dedução, daqui para frente serão usadas tanto dedução como árvore de dedução.

5.6 COMPLETUDE DO PRINCÍPIO DE RESOLUÇÃO

No Capítulo 4, foi apresentado o conceito de árvores semânticas para provar o teorema de Herbrand. Nesta seção, serão usadas árvores semânticas para provar a completude do princípio de resolução. De fato, existe uma forte relação entre a árvore semântica e resolução, como é mostrado com o exemplo seguinte.

Exemplo 5.17: Considere o seguinte conjunto S de cláusulas:

(1) P
(2) ~P 2 Q
(3) ~P 2 ~Q

O conjunto de átomos de S é {P,Q}. Seja T uma árvore semântica completa como mostrado na figura 5.4a. T tem uma árvore semântica fechada T' mostrada na fig. 5.4b. O nó (2) na fig. 5.4b é um nó de inferência. Seus dois sucessores, nó (4) e (5), são nós falha. As cláusulas falsificadas nos nós (4) e (5) são ~P 2 ~Q e ~P 2 Q, respectivamente.Como foi visto anteriormente, essas duas cláusulas devem ter um par complementar de literais, e portanto podem ser resolvidas. Resolvendo ~P 2 ~Q e ~P 2 Q, obtém-se ~P. Nota-se que ~P é falsificado pela interpretação parcial correspondente ao nó (2). Se colocar ~P em S, então pode-se ter a árvore semântica fechada T'' para S 2 {~P}, como mostrado na Fig. 5.4c. Nesta figura, o nó (1) é um nó de inferência. Neste ponto, pode-se obter 1 resolvendo P e ~P. Colocando 1 em S 2 {~P}, chega-se à árvore semântica fechada T''' para S 2 {~P} 2 {1} como mostrado na figura 5.4d. A eliminação da árvore semântica acima corresponde a uma prova por resolução como segue:

(4) ~P um resolvente de (2) e (3)
(5) 1 um resolvente de (4) e (1).

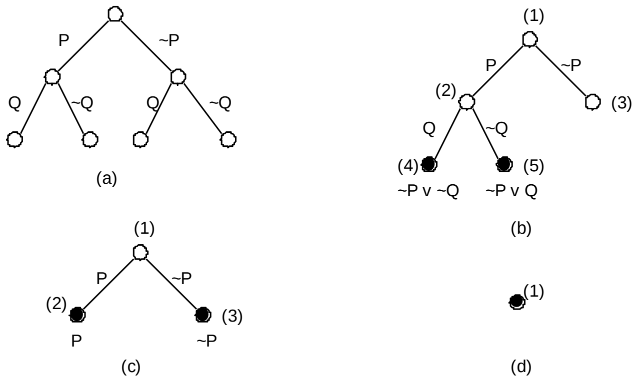

Figura 5.4 (a) T, (b) T'.(~P 2 Q) e (~P 2 ~Q) pode ser resolvido para gerar ~P, (c) T''.P e ~P pode ser resolvido para gerar □, (d) T'''.

Na seqüência, usa-se os conceitos desenvolvidos anteriormente para provar a completude do princípio da resolução. Isto é, após construir uma árvore semântica fechada para um conjunto insatisfatível de cláusulas, pode-se gradualmente forçar o desaparecimento da árvore e, ao mesmo tempo, obter uma prova por resolução. Antes de se provar o teorema da completude, prova-se um lema chamado de *lifting lemma*.

Lema 5.1: (*Lifiting Lemma*) Se C_1' e C_2' são instâncias de C_1 e C_2 , respectivamente, e se C' é um resolvente de C_1' e C_2' , então existe um resolvente C de C_1 e C_2 tal que C' é uma instância de C.

Prova: Renomeia-se, se necessário, as variáveis em C_1 e C_2 ,tal que as variáveis em C_1 sejam todas diferentes das em C_2 . Sendo L_1' e L_2' , os literais a serem resolvidos, e sendo:

$$C'' = (C_1'1 - L_1'1)\ 2\ (C_2'1 - L_2'1),$$

onde 1 é o unificador mais geral de L_1' e ~L_2' . Uma vez que C_1' e C_2' são instâncias de C_1 e C_2 , respectivamente, existe uma substituição 1 tal que $C_1' = C_1 1$ e $C_2' = C_2 1$. Sendo L_i^1, ..., L_i^{ri} os literais em C_i correspondentes a L_i' , (isto é, $L_i^1 1 = ... = L_i^{ri} 1 = L_i'$), i = 1,2. Se $r_i > 1$, obtém-se um unificador mais geral 1_i para $\{L_i^1, ..., L_i^{ri}\}$ e faça $L_i = L_i^1 1_i$, i = 1,2. (Observe que $L_i^1 1_i$, ... , $L_i^{ri} 1_i$ são os mesmos, uma vez que 1_i é um unificador mais geral). Então L_i é um literal no fator $C_i 1_i$ de C_i . Se $r_i = 1$, sendo $1_i = 1$ e $L_i = L_i^1 1_i$. Faça $1 = 1_1\ 2\ 1_2$. Desse modo, L_i' é uma instância de L_i. Uma vez que L_i' e L_2' são unificáveis, L_1 e ~L_2 também o são. Sendo 1 o unificador mais geral de L_1 e ~L_2 . Considerando:

$$\begin{aligned} C &= ((C_1 1)1 - L_1 1)\ 2\ ((C_2 1)1 - L_2 1) \\ &= ((C_1 1)1 - (\{L_1^1, ..., L_1^{r1}\}1)1)\ 2\ ((C_2 1)1 - (\{L_2^1, ..., L_2^{r2}\}1)1) \\ &= ((C_1(111) - \{L_1^1, ..., L_1^{r1}\}(111))\ 2\ ((C_2(111) - \{L_2^1, ..., L_2^{r2}\}(111)) \end{aligned}$$

C é um resolvente de C_1 e C_2 . De modo claro, C' é uma instância de C, uma vez que

$$\begin{aligned} C' &= (C_1'1 - L_1'1)\ 2\ (C_2'1 - L_2'1) \\ &= ((C_1 1)1 - (\{L_1^1, ..., L_1^{r1}\}1)1)\ 2\ ((C_2 1)1 - (\{L_2^1, ..., L_2^{r2}\}1)1) \\ &= ((C_1(1\ 11) - \{L_1^1, ..., L_1^{r1}\}(1\ 11))\ 2\ ((C_2(1\ 11) - \{L_2^1, ..., L_2^{r2}\}(1\ 11)) \end{aligned}$$

e 111 é mais geral que 1 11. Assim completa-se a prova deste lema.

Teorema 5.3: (O Princípio de Resolução) Um conjunto de cláusulas é não satisfatório se e somente se existir uma dedução de uma cláusula vazia □ de S.

Prova: (IDA) Supondo que S é insatisfatível. Seja A = $\{A_1, A_2, A_3, ... \}$ o conjunto de átomos de S e T a árvore semântica completa como mostrado na figura 5.5. Pelo Teorema de Herbrand (versão I), T tem uma árvore semântica fechada finita T'. Se T' consiste somente de um nó (raiz), então □ deve estar em S, pois nenhuma outra cláusula pode ser falsificada na raiz de uma árvore semântica. Neste caso, o teorema é obviamente verdadeiro. Assumindo que T' consiste de mais de um nó, então T' tem pelo menos um nó de inferência, pois, se não existir, então todo nó poderia ter pelo menos um sucessor não falha. Poder-se-ia, então, encontrar um ramo infinito em T', contradizendo o fato de que T' é uma árvore semântica fechada finita. Sendo N um nó de inferência em T' e N_1 e N_2 os nós falha imediatamente abaixo de N. Seja,

$$I(N) = \{m_1, m_2, ... , m_n\}$$
$$I(N_1) = \{m_1, m_2, ... , m_n, m_{n+1}\}$$
$$I(N_2) = \{m_1, m_2, ... , m_n, \sim m_{n+1}\}$$

Uma vez que N_1 e N_2 são nós falha, mas N não é um nó falha, devem existir duas instâncias base C_1’ e C_2’ de cláusulas C_1 e C_2 , tais que C_1’ e C_2’ são falsas em $I(N_1)$ e $I(N_2)$, respectivamente, mas tanto C_1’ quanto C_2’ não são falsificadas por I(N). Portanto C_1’ deve conter $\sim m_{n+1}$ e C_2’ deve conter m_{n+1} . Sendo L_1’ = $\sim m_{n+1}$ e L_2’ = m_{n+1} . Resolvendo com os literais L_1’ e L_2’ , pode-se obter um resolvente C' de C_1’ e C_2’ , dito,

$$C' = (C_1' - L_1') \; 2 \; (C_2' - L_2').$$

C' deve ser falsa em I(N) uma vez que (C_1’ - L_1’) e (C_2’ - L_2’) são falsos em I(N). Pelo lema 5.1, existe um resolvente C de C_1 e C_2 tal que C' é uma instância base de C. Sendo T'' a árvore semântica fechada de (S 2 {C}) obtida, a partir de T', pela eliminação de todo nó ou ramo que estiver abaixo do primeiro nó onde o resolvente C' é falsificado. De maneira clara, o número de nós em T'' é menor que em T'. Aplicando o processo acima, novamente, em T'', pode-se obter outro resolvente das cláusulas em (S 2 {C}). Colocando este resolvente em (S 2 {C}), pode-se obter outra árvore semântica ainda menor. Este processo é aplicado repetidamente até que a árvore semântica seja composta de um único nó, ou seja, o nó raiz é gerado. Isto é possível somente quando □ é derivado. Portanto, é uma dedução de □ a partir de S.

(VOLTA) De modo inverso, suponha que existe uma dedução de □ a partir de S. Seja R_1, R_2, ... , R_k os resolventes na dedução. Assuma que S é satisfatível. Então existe um modelo M de S. Entretanto, se um modelo satisfaz as cláusulas C_1 e C_2 , ele deve satisfazer, também, qualquer resolvente de C_1 e C_2 (Teorema 5.1). Portanto, M satisfaz R_1, R_2, ... , R_k . Entretanto, isto é impossível, pois um destes resolventes é □. Portanto, S deve ser insatisfatível.

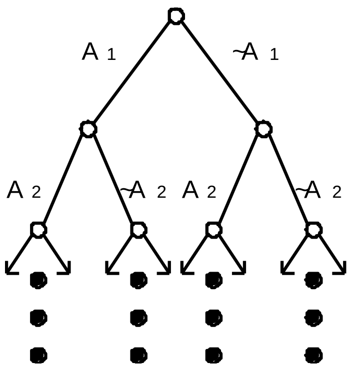

Figura 5.5 Árvore Semântica Completa

5.7 EXEMPLOS USANDO O PRINCÍPIO DE RESOLUÇÃO

Nos capítulos anteriores, foram usados muitos métodos para provar a inconsistência de fórmulas. Nesta seção serão mostrados exemplos da eficiência do princípio de resolução na prova de teoremas.

Exemplo 5.18: Usando-se novamente o Exemplo 2.10. Tem-se 4 declarações:

(1’) P 1 S
(2’) S 1 U
(3’) P
(4’) U.

Para mostrar que (4’) vem de (1’), (2’) e (3’), primeiro transforma-se todas essa declarações em formas padrões. Então, tem-se:

(1) ~P 2 S
(2) ~S 2 U
(3) P
(4) U.

Prova-se que U é conseqüência lógica de (1), (2) e (3) por refutação. Nega-se (4) e obtém-se a seguinte prova:

(1) ~P 2 S
(2) ~S 2 U
(3) P
(4) ~U negação da conclusão
(5) S um resolvente de (3) e (1)
(6) U um resolvente de (5) e (2)
(7) ☐ um resolvente de (6) e (4).

Exemplo 5.19: Reconsiderando o exemplo 2.12 no capítulo 2. Tem-se três fórmulas F_1 , F_2 e F_3. Deseja-se provar que ~Q é uma consequência lógica de F_1 2 F_2 2 F_3. Nega-se ~Q e transforma-se F_1 2 F_2 2 F_3 em uma forma padrão. Isso gera o seguinte conjunto de cláusulas:

(1) ~P 2 ~Q 2 R de F_1
(2) ~P 2 ~Q 2 S de F_1
(3) P de F_2
(4) ~S de F_3
(5) Q negação da conclusão.

Usando resolução, obtém-se a seguinte prova:

(6) ~Q 2 S um resolvente de (3) e (2)
(7) S um resolvente de (6) e (5)
(8) ☐ um resolvente de (7) e (4).

Exemplo 5.20: Considerar o seguinte conjunto de fórmulas:

F_1: (3x) (C(x) 1 (W(x) 2 R(x)))
F_2: (3x) (C(x) 2 O(x))
G: (3x) (O(x) 2 R(x)).

O problema é mostrar que G é uma consequência lógica de F_1 e F_2. Transforma-se F_1 e F_2 e ~G em formas padrões e obtém-se as cinco cláusulas seguintes:

(1) ~C(x) 2 W(x) de F_1
(2) ~C(x) 2 R(x) de F_1
(3) C(a) de F_2
(4) O(a) de F_2
(5) ~O(x) 2 ~R(x) de ~G.

O conjunto de cláusulas acima é insatisfatível. Isto pode ser provado por resolução como segue:

(6)	R(a)	um resolvente de (3) e (2)
(7)	~R(a)	um resolvente de (5) e (4)
(8)	☐	um resolvente de (7) e (6).

Portanto, G é uma consequência lógica de F_1 e F_2 .

Exemplo 5.21: Reconsiderando o Exemplo 3.15. Tem-se:

F_1: (3x) (P(x) 2 (3y) (D(y) 1 L(x,y)))

F_2: (3x) (P(x) 1 (3y) (Q(y) 1 ~L(x,y)))

G: (3x) (D(x) 1 ~Q(x)).

De modo usual, F_1 , F_2 e ~G são transformados nas seguintes cláusulas:

(1)	P(a)	de F_1
(2)	~D(y) 2 L(a, y)	de F_1
(3)	~P(x) 2 ~Q(y) 2 ~L(x,y)	de F_2
(4)	D(b)	de ~G
(5)	Q(b)	de ~G.

Usando resolução, obtém-se a seguinte prova:

(6)	L(a,b)	um resolvente de (4) e (2)
(7)	~Q(y) 2 ~L(a,y)	um resolvente de (3) e (1)
(8)	~L(a,b)	um resolvente de (5) e (7)
(9)	☐	um resolvente de (6) e (8).

O princípio de resolução é bastante natural. Vai-se tentar de traduzir a prova acima para o português, como segue:

(a) De F_1, pode-se assumir que exista um paciente *a* que gosta de todos os médicos (cláusulas (1) e (2)).

(b) Assume-se que a conclusão está errada. Isto é , assume-se que *b* é tanto um médico quanto um curandeiro (cláusulas (4) e (5)).

(c) Desde que o paciente **a** gosta de todos os médicos, *a* gosta de *b* (cláusula (6)).

(d) Desde que *a* é um paciente, *a* não gosta de nenhum curandeiro (cláusula (7)).

(e) Porém, *b* é um curandeiro. Portanto, *a* não gosta de *b* ((cláusula (8)).

(f) Isso é impossível a partir de (c). Então, a prova fica completa.

Exemplo 5.22: Premissas: Os policiais da fronteira revistavam todos que entravam em seu país e que não eram pessoas VIP. Alguns traficantes de drogas entravam neste país e eram revistados somente por traficantes de drogas. Nenhum traficante de drogas era VIP.

Conclusão: Algum dos policiais da fronteira eram traficante de drogas.

Fazendo E(x) representar "x entrou no país", V(x) representar "x era um VIP", S(x,y) "y revistou x", C(x) "x era um policial da fronteira", e P(x) representar "x era um traficante de drogas".

As premissas são representadas pelas seguintes fórmulas:

(3x) (E(x) 2 ~V(x) 1 (3y) (S(x,y) 2 C(y)))

(3x) (P(x) 2 E(x) 2 (3y) (S(x,y) 1 P(y)))

(3x) (P(x) 1 ~V(x))

e a conclusão é: (3x) (P(x) 2 C(x)).

Transformando as premissas em cláusulas, obtém-se:

(1)	~E(x) 2 V(x) 2 S(x, f(x))
(2)	~E(x) 2 V(x) 2 C(f(x))
(3)	P(a)
(4)	E(a)

(5) ~S(a, y) 2 P(y)
(6) ~P(x) 2 ~V(x).
A negação da conclusão é: (7) ~P(x) 2 ~C(x).
A prova por resolução é a seguinte:

(8)	~V(a)	de (3) e (6)
(9)	V(a) 2 C(f(a))	de (2) e (4)
(10)	C(f(a))	de (8) e (9)
(11)	V(a) 2 S(a, f(a))	de (1) e (4)
(12)	S(a, f(a))	de (8) e (11)
(13)	P(f(a))	de (12) e (5)
(14)	~C(f(a))	de (13) e (7)
(15)	☐	de (10) e (14).

Então, consegue-se chegar a conclusão.

Exemplo 5.23: A premissa é que todo aquele que economiza dinheiro recebe juros. A conclusão é que se não existir juros, ninguém economiza dinheiro. Fazendo S(x,y), M(x), I(x), e E(x,y) representar "x economiza y", "x é dinheiro", "x é juros" e "x ganha y", respectivamente. Então a premissa é simbolizada como:

(3x) ((3y) (S(x,y) 2 M(y)) 1 (3y) (I(y) 2 E(x,y)))

e a conclusão é: ~(3x) I(x) 1 (3x) (3y) (S(x,y) 1 ~M(y)).

Transformando a premissa em cláusulas, tem-se:

(1) ~S(x,y) 2 ~M(y) 2 I(f(x))
(2) ~S(x,y) 2 ~M(y) 2 E(x, f(x)).
A negação da conclusão é: (3) ~I(z)
(4) S(a,b)
(5) M(b).
A prova mostrada a seguir é simples:

(6)	~S(x,y) 2 ~M(y)	de (3) e (1)
(7)	~M(b)	de (6) e (4)
(8)	☐	de (7) e (5).

Assim, chega-se à conclusão.

Exemplo 5.24: Premissa: Estudantes são cidadãos.
Conclusão: Votos de estudantes são votos de cidadãos.

Fazendo S(x), C(x), e V(x,y) representar "x é um estudante", "x é um cidadão", e "x é um voto de y", respectivamente. Então a premissa e a conclusão são simbolizadas como o seguinte:

(3y) (S(y) 1 C(y)) (premissa)
(3x) ((3y) (S(y) 2 V(x,y) 1 (3z) (C(z) 2 V(x,z))) (conclusão)

Uma forma padrão para a premissa é:

(1) ~S(y) 2 C(y).

Uma vez que: ~((3x) ((3y) (S(y) 2 V(x,y)) 1 (3z) (C(z) 2 V(x,z))))
= ~((3x) ((3y) (~S(y) 2 ~V(x,y)) 2 (3z) (C(z) 2 V(x,z))))
= ~((3x) (3y)(3z) (~S(y) 2 ~V(x,y)) 2 (C(z) 2 V(x,z))))
= (3x) (3y)(3z) (S(y) 2 V(x,y)) 2 (~C(z) 2 ~V(x,z)))

tem-se três cláusulas derivadas da negação da conclusão,

(2) S(b)
(3) V(a,b)
(4) ~C(z) 2 ~V(a,z).

O procedimento de prova é o seguinte:

(5)	C(b)	de (1) e (2)
(6)	~V(a, b)	de (5) e (4)
(7)	□	de (6) e (3).

Pode-se traduzir a prova acima para o português, como segue: Assume-se que *b* é um estudante, *a* é o seu voto, e *a* não é um voto de qualquer cidadão. Desde que *b* é um estudante, *b* deve ser um cidadão. Portanto, *a* não deve ser um voto de *b* por que *b* é um cidadão. Isto é impossível. Então a prova está concluída.

5.8 ESTRATÉGIA DE ELIMINAÇÃO

Nas seções anteriores, mostrou-se que a resolução é completa. Resolução também é mais eficiente que os métodos anteriores tal como o procedimento de Herbrand usado por Gilmore. No entanto, aplicações repetitivas da resolução podem gerar muitas cláusulas irrelevantes e redundantes. Para observar isso, considera-se um exemplo simples: Suponha que se quer mostrar, por resolução, que o conjunto de cláusulas S = {P 2 Q, ~P 2 Q, P 2 ~Q, ~P 2 ~Q} é contraditório. Um caminho simples para a resolução do conjunto S é calcular todos os resolventes dos pares de cláusulas de S, adicionar esses resolventes ao conjunto S, calcular todos os resolventes , e repetir este processo até que se chegue à cláusula □. Isto é , gera-se as sequências , S^0, S^1, S^2, ..., onde: $S^0 = S$,

$$S^n = \{\text{soluções de } C_1 \text{ e } C_2 \mid C_1\ 2\ (S^0\ 2 \ldots 2\ S^{n-1}) \text{ e } C_2\ 2\ S^{n-1}\},\ n = 1, 2, 3, \ldots$$

Este procedimento é chamado *método (resolução) de saturação*. Para programar este método em computadores, pode-se primeiro listar cláusulas de (S^0 2 ... 2 S^{n-1}) em ordem, e então calcular os resolventes comparando todas as cláusulas C_1 2 (S^0 2 ... 2 S^{n-1}) com uma cláusula C_2 2 S^{n-1} a qual é listada depois de C_1. Quando um resolvente é calculado, ele é anexado ao fim da lista que foi gerada. Isso é o que foi feito no conjunto de cláusulas geradas mostrado abaixo:

S^0:	(1)	P 2 Q	
	(2)	~P 2 Q	
	(3)	P 2 ~Q	
	(4)	~P 2 ~Q	
S^1:	(5)	Q	de (1) e (2)
	(6)	P	de (1) e (3)
	(7)	Q 2 ~Q	de (1) e (4)
	(8)	P 2 ~P	de (1) e (4)
	(9)	Q 2 ~Q	de (2) e (3)
	(10)	P 2 ~P	de (2) e (3)
	(11)	~P	de (2) e (4)
	(12)	~Q	de (3) e (4)
S^2:	(13)	P 2 Q	de (1) e (7)
	(14)	P 2 Q	de (1) e (8)
	(15)	P 2 Q	de (1) e (9)
	(16)	P 2 Q	de (1) e (10)
	(17)	Q	de (1) e (11)
	(18)	P	de (1) e (12)
	(19)	Q	de (2) e (6)
	(20)	~P 2 Q	de (2) e (7)

(21) ~P 2 Q de (2) e (8)
(22) ~P 2 Q de (2) e (9)
(23) ~P 2 Q de (2) e (10)
(24) ~P de (2) e (12)
(25) P de (3) e (5)
(26) P 2 ~Q de (3) e (7)
(27) P 2 ~Q de (3) e (8)
(28) P 2 ~Q de (3) e (9)
(29) P 2 ~Q de (3) e (10)
(30) ~Q de (3) e (11)
(31) ~P de (4) e (5)
(32) ~Q de (4) e (6)
(33) ~P 2 ~Q de (4) e (7)
(34) ~P 2 ~Q de (4) e (8)
(35) ~P 2 ~Q de (4) e (9)
(36) ~P 2 ~Q de (4) e (10)
(37) Q de (5) e (7)
(38) Q de (5) e (9)
(39) ☐ de (5) e (12).

Muitas cláusulas irrelevantes e reduntantes foram geradas. Por exemplo, (7), (8), (9) e (10) são tautologias. Desde que uma tautologia é verdadeira em qualquer interpretação, retirando uma tautologia de um conjunto de cláusulas insatisfatível, o conjunto resultante continuará insatisfatível. Portanto, uma tautologia é uma cláusula irrelevante e não deve ser gerada. Se for gerada, com exceção de poucos casos, deve ser eliminada. Do contrário, ela pode interagir com outras cláusulas e produzir outro grupo de cláusulas reduntantes. Por exemplo, (13) - (16), (20) - (23), (26) - (29), (33) - (36), e (37) - (38) são cláusulas dessa forma. Além disso, as cláusulas P, Q, ~P, e ~Q são repetidamente geradas.

Na verdade, para se conseguir uma prova para S, necessita-se somente gerar as cláusulas (5), (12) e (39). Para resolver esse problema da redundância, será estudada nesta seção a *estratégia de eliminação*. Antes de descrevê-la, considera-se a seguinte definição.

Definição: Uma cláusula C engloba uma cláusula D se e somente se existe uma substituição 1 tal que C1 2 D . D é chamada cláusula englobada.

Exemplo 5.25: Fazendo C = P(x) e D = P(a) 2 Q(a). Se 1 = { a/x}, então C1 = P(a). Desde que C1 2 D, C engloba D.

Percebe-se que se D é idêntico a C, ou D é uma instância de C, então D é englobado por C. *A estratégia de eliminação é a eliminação de qualquer tautologia e qualquer cláusula englobada sempre que possível.* A estratégia de eliminação é uma generalização da regra de tautologia de Davis e Putnam mostrada na seção 4.6. A completude da estratégia de eliminação depende de como as tautologias e as cláusulas englobadas são eliminadas. A estratégia de deleção será completa se for usada com o método de saturação na seguinte ordem: Primeiro, listar as cláusulas (S^0 2 ... 2 S^{n-1}) em ordem; então calcular resolventes comparando todas as cláusulas C_1 2 (S^0 2 ... 2 S^{n-1}) com uma cláusula C_2 2 S^{n-1} a qual é listada após C_1 . Quando um resolvente é calculado, é anexado ao final da lista se não for uma tautologia nem uma cláusula englobada. Do contrário é eliminado.

Aplicando-se a estratégia de eliminação ao exemplo acima chega-se a seguinte lista:

S^0: (1) P 2 Q

(2) ~P 2 Q
(3) P 2 ~Q
(4) ~P 2 ~Q

S^2:
(5) Q de (1) e (2)
(6) P de (1) e (3)
(7) ~P de (2) e (4)
(8) ~Q de (3) e (4)
(9) □ de (5) e (8).

Nota-se que essa lista é menor que a lista gerada anteriormente. Portanto, a estratégia de eliminação pode aumentar a eficiência do princípio de resolução.

É importante , para se usar a estratégia de eliminação, saber decidir se uma cláusula é uma tautologia e se uma cláusula engloba outra cláusula. Fica fácil testar se uma cláusula é uma tautologia se nela estiver presente um par complementar. Entretanto, o teste de englobamento não é simples. Agora, será descrito o algoritmo para realizar esse teste.

Sejam C e D cláusulas. Seja 1 = {a_1/x_1, ..., a_n/x_n}, onde x_1, ..., x_n são variáveis que aparecem em D, e a_1, ..., a_n são novas constantes distintas que não aparecem nem em C nem em D. Supondo que D = L_1 2 L_2 2 ... 2 L_m. Então, D1 = $L_1$1 2 $L_2$1 2 ... 2 L_m1. Nota-se que D1 é uma cláusula base. ~ D1 = ~$L_1$1 2 ... 2 ~L_m1. O algoritmo seguinte testa se C engloba ou não D.

Algoritmo de Englobamento

Passo 1: Sendo W = { ~$L_1$1 , ... , ~L_m1}.

Passo 2: Faça k = 0 e U^0 = {C}.

Passo 3: Se U^k contém □, terminar; C engloba D. Caso contrário, faz-se U^{k+1} = {resolventes de C_1 e C_2 | C_1 2 U^k e C_2 2 W}.

Passo 4: Se U^{k+1} é vazia, terminar; C não engloba D. Caso contrário, faz-se k = k+1 e vai-se para o Passo 3.

No algoritmo acima, nota-se que cada cláusula em U^{k+1} é menor, em um literal, que a cláusula em U^k da qual foi obtida. Portanto, a sequência U^0, U^1, ..., deve, em algum momento, conter um conjunto que contenha □ ou ser vazia. O algoritmo de englobamento é correto. Isto é, C engloba D se e somente se terminar no Passo 3. Isto é provado como segue: se C engloba D existe uma substituição 1, tal que C1 2 D. Portanto, C(1 1 1) 2 D1. Deste modo, os literais em C(1 1 1) podem ser resolvidos usando-se cláusulas unitárias base em W. Entretanto, C(1 11) é uma instância de C. Portanto, literais em C podem ser resolvidos mais adiante usando cláusulas unitárias em W. Isto significa que pode-se, eventualmente, encontrar um U^k que contenha □. Portanto, o algoritmo irá terminar no Passo 3. De modo inverso, se o algoritmo terminar no Passo 3, então obtém-se a refutação mostrada na figura 5.6, onde B_0, ... , B_r são cláusulas unitárias de W. R_1 é um resolvente de C e B_0 e R_i é um resolvente de R_{i-1} e B_{i-1} para i = 2, ... , r.

Sendo 1_0 e 1_i os unificadores mais gerais obtidos quando se reslve C e B_0 e R_i e B_i , i = 1, ... , r, respectivamente. Então, C(1_0 1 1_1 1 ... 1 1_r) = {~B_0, ~B_1, ..., ~B_r} 2 D1. Sendo 1 = 1_0 1 1_1 1 ... 1 1_r , então C1 2 D1. Sendo 1 a substituição obtida de 1 pela substituição, em cada componente de 1_i, de x_i por a_i , para i = 1, ..., n. Então C1 2 D, portanto C engloba D.

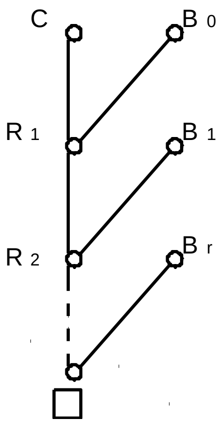

Figura 5.6 Refutação obtida quando o Algoritmo de Englobamento pára no Passo 3

Exemplo 5.26: Sendo C = ~P(x) 2 Q(f(x), a) e D = ~P(h(y)) 2 Q(f(h(y)), a) 2 ~P(z). Verificar se C engloba D.

1. y e z são variáveis em D. Seja 1 = {b/y, c/z }. Nota-se que b e c não se encontram em C ou D. Então D1 = ~P(h(b)) 2 Q(f(h(b)), a) 2 ~P(c). Portanto, D1 = ~P(h(b)) 2 ~Q(f(h(b)), a) 2 P(c). Deste modo: W = { P(h(b)) , ~Q(f(h(b)), a), P(c) }
U^0 = { ~P(x) 2 Q(f(x), a)}.
2. Uma vez que U^0 não contém □, obtém-se:
U^1 = { Q(f(h(b)), a) , ~P(h(b)), Q(f(c), a) }.
3. Já que U^1 não é vazio e não contém □, obtém-se U^2 = {□}.
4. Uma vez que U^2 contém □, o algoritmo termina e conclui-se que C engloba D.

Exemplo 5.27: Seja C = P(x,x) e D = P(f(x), y) 2 P(y, f(x)) . Determine se C engloba ou não D.

1. x e y são variáveis em D. Escolher novas constantes a e b diferentes de qualquer constante em C ou D. Seja 1 = { a/x, b/y }. Então,
D1 = P(f(a), b) 2 P(b, f(a)).
~D1 = ~P(f(a), b) 2 ~P(b, f(a)).
Assim, W = { ~P(f(a), b), ~P(b, f(a)) }
U^0 = {P(x,x)}.
2. Uma vez que U^0 não contém □, obtém-se: U^1 = 1 .
3. Uma vez que U^1 é vazio, o algoritmo termina e conclui-se que C não engloba D.

5.9 EXERCÍCIOS

Seção 5.2

1. Provar que o seguinte conjunto de cláusulas é insatisfatível usando resolução:
P 2 Q 2 R , ~P 2 R, ~Q 2 ~R.
2. A partir do conjunto S = { P 2 Q, ~Q 2 R , ~P 2 Q, ~R} derivar em uma cláusula vazia de por resolução.

Seção 5.3

3. Sendo 1 = {a/b, b/y, g(x, y)/z} uma substituição e E = P(h(x), z). Encontrar E1.

4. Sendo 1_1 = {a/x, f(z)/y, y/z} e 1_2 = {b/x, z/y, g(x)/z}. Achar a composição de 1_1 e 1_2 .

5. Provar o seguinte teorema: Sejam 1, 1, e 1 substituições. Então 1 1 1 1 1 = 1 1 (1 1 1) (associatividade).

Seção 5.4

6. Determinar se cada um dos seguintes conjuntos de cláusulas é unificável. Se for, obter um unificador mais geral.

(1) W = {Q(a), Q(b) }
(2) W = {Q(a, x), Q(a, a) }
(3) W = {Q(a, x, f(x)), Q(a, y, y) }
(4) W = {Q(x, y, z), Q(u,h(v,v), u)}
(5) W = {P(x_1, g(x_1), x_2, h(x_1, x_2), x_3 , k(x_1, x_2, x_3)), P(y_1, y_2, e(y_2), y_3, f(y_2, y_3), y_4)}.

Seção 5.5

7. Determinar se as seguintes cláusulas têm fatores e quando houver, determiná-los.

(1) P(x) 2 Q(y) 2P(f(x))
(2) P(x) 2P(a) 2 Q(f(x)) 2 Q(f(a))
(3) P(x,y) 2P(a, f(a))
(4) P(a) 2 P(b) 2P(x)
(5) P(x) 2 P(f(y)) 2Q(x,y).

8. Encontrar todos os resolventes possíveis (se houver) do seguinte conjunto de cláusulas:

(1) C = ~P(x) 2 Q(x,b), D = P(a) 2 Q(a,b)
(2) C = ~P(x) 2 Q(x,x), D = ~Q(a, f(a))
(3) C = ~P(x, y, u) 2 ~P(y, z, v) 2 ~P(x, v, w) 2 P(u, z, w), D = P(g(x,y), x, y)
(4) C = ~P(v, z, v) 2 P(w, z, w), D = P(w, h(x,x), w).

Seção 5.7

9. Reconsiderar o exemplo 2.13 na Seção 2.6. Mostrar que H_2CO_3 pode ser provado através de resolução.

10. No exercício 14 do Capítulo 2, pediu-se para mostrar que Q é uma consequência lógica de P e (P 1 Q). Pode-se provar isso usando o princípio de resolução?

11. Reconsiderar o exercício 12 do Capítulo 3. Resolva-o usando o princípio de resolução.

12. Considerar o exemplo 4.16. Provar que o conjunto de cláusulas nesse exemplo é insatisfatível através do princípio de resolução.

13. Usar o princípio de resolução para provar que o conjunto de cláusulas no exercício 14 do Capítulo 4 é insatisfatível.

14. Provar que (~Q 1 ~P) é uma consequência lógica de (P 1 Q) através da resolução.

Seção 5.8

15. Seja C = P(x, y) 2 Q(z) e D = Q(a) 2 P(b,b) 2 R(u). Determinar se C engloba D.

16. Seja C = P(x,y) 2 R(y,x) e D = P(a, y) 2 R(z, b). Provar que C não engloba D.

17. Seja C = ~P(x) 2 P(f(x)) e D = ~P(x) 2 P(f(f(x))). Mostrar que C implica em D, mas C não engloba D.

CAPÍTULO 6 - RESOLUÇÃO SEMÂNTICA E RESOLUÇÃO INDEXADA

6.1 INTRODUÇÃO

No Capítulo anterior, considerou-se o princípio da resolução como uma regra de inferência que pode ser usada para gerar novas cláusulas a partir de outras. Também foi visto que aplicações repetidas e ilimitadas da resolução podem gerar várias cláusulas irrelevantes e redundantes, além das realmente necessárias ao processo. Apesar da estratégia de eliminação poder ser utilizada para eliminar algumas destas cláusulas irrelevantes ou redundantes logo após elas serem geradas, o tempo para gerá-las já foi gasto. Além disso, se cláusulas inúteis são geradas, inúmeros recursos, tais como, tempo computacional e memória, são necessários para determinar se elas são realmente irrelevantes ou redundantes. Portanto, de modo a se ter procedimentos eficientes de prova de teoremas, deve-se evitar a geração de um grande número de cláusulas inúteis. Isto leva a alguns questionamentos sobre como refinar o processo de resolução. Existem muitas maneiras de se melhorar o processo de resolução, considerar-se-á aqui três destes refinamentos, ou seja, resolução semântica , resolução indexada (discutidos neste Capítulo) e resolução linear (discutida no Capítulo seguinte).

6.2 INTRODUÇÃO INFORMAL À RESOLUÇÃO SEMÂNTICA

Considere o seguinte exemplo:

Exemplo 6.1: Seja S = {~P ∨ ~Q ∨ R, P ∨ R, Q ∨ R, ~R}. Para provar a insatisfiabilidade de S, usando o princípio ordinário de resolução, seria gerado o seguinte conjunto de cláusulas:

(1)	~P ∨ ~Q ∨ R	conjunto original S
(2)	P ∨ R	conjunto original S
(3)	Q ∨ R	conjunto original S
(4)	~R	conjunto original S
(5)	~Q ∨ R	a partir de (1) e (2)
(6)	~P ∨ R	a partir de (1) e (3)
(7)	~P ∨ ~Q	a partir de (1) e (4)
(8)	P	a partir de (2) e (4)
(9)	Q	a partir de (3) e (4)
(10)	~Q ∨ R	a partir de (1) e (8)
(11)	~P ∨ R	a partir de (1) e (9)
(12)	R	a partir de (2) e (6)
(13)	~Q	a partir de (4) e (5)
(14)	~P	a partir de (4) e (6)
(15)	□	a partir de (4) e (12)

Entre todas estas cláusulas geradas, somente (6) e (12) são realmente usadas na prova. Todas as outras são irrelevantes e/ou redundantes. Vai-se discutir agora, mecanismos que reduzem o número destas cláusulas inúteis.

Imagine que se pode, de algum modo, dividir o conjunto de cláusulas S em dois grupos: S_1 e S_2. Indo mais adiante, imagine que não se permita que cláusulas dentro do mesmo grupo sejam resolvidas entre si. Isto, obviamente, reduzirá o número de cláusulas geradas.

Por exemplo, dividindo-se o conjunto de cláusulas do Exemplo 6.1 em S_1 (cláusulas (1) e (3)) e S_2 (cláusulas (1) e (4)). Uma vez que a cláusula (1) e a cláusula (4) estão em S_2, (1) e (4) não poderão ser resolvidas. Como pode ser visto o bloqueio da resolução de (1) e (4) nãp afetará a habilidade do método em deduzir a cláusula vazia □.

A questão é: como dividir as cláusulas em dois grupos? Na resolução semântica, usa-se uma interpretação para dividir as cláusulas. Por isto este tipo de resolução é chamada de resolução semântica. Observe que no Exemplo 6.1, (2) e (3) são falsificadas pela interpretação I = {~P, ~Q, ~R}, enquanto que (1) e (4) são satisfeitas por I. Portanto, esta interpretação pode ser usada para dividir S em S_1 e S_2, onde S_1 contém todas as cláusulas falsificadas por I e S_2 contém as satisfeitas por I. Deve-se lembrar que trata-se de um conjunto insatisfatível de cláusulas, portanto, nenhuma interpretação poderá satisfazer ou falsificar todas as cláusulas. Desse modo, toda interpretação I divide S em dois conjuntos não vazios de cláusulas e, como vai-se ver na seqüência, qualquer interpretação poderá ser utilizada.

Até este momento, obteve-se sucesso em bloquear a resolução de (1) e (4). Poder-se-á impor algumas restrições adicionais sobre a resolução de modo que mais resoluções sejam bloqueadas? Observe que no Exemplo 6.1, (2) e (3) ainda podem ser resolvidas com (4). Deve-se mostrar que estas resoluções também podem ser bloqueadas.

Para bloquear estas resoluções, pode-se usar um esquema ingênuo, ou seja, uma ordenação dos predicados. Suponha que se ordene todos os predicados em S como segue:

$$P > Q > R.$$

Suponha que se exija que, quando se resolve duas cláusulas (uma de S_1 e outra de S_2), o literal resolvido na cláusula de S_1contenha o maior predicado na cláusula. Com esta resrtrição, não se pode resolver (2) com (4), nem (3) com (4), pois R não é o maior literal em (2) e (3).

As idéias seguintes - o uso de uma interpretação para dividir as cláusulas em dois grupos e o uso de uma ordenação de predicados para reduzir o número de possíveis resoluções - são conceitos importantes na resolução semântica. Estas idéias serão elaboradas no decorrer deste capítulo, enquanto isto, será introduzido outro conceito importante, o de conflito.

Exemplo 6.2: Considere as cláusulas do Exemplo 6.1:

(1) $\sim P \vee \sim Q \vee R$
(2) $P \vee R$
(3) $Q \vee R$
(4) $\sim R$

Sendo I = {~P, ~Q, ~R} e uma ordenação de predicados de modo que: P > Q > R. Usando-se as duas restrições discutidas anteriormente, pode-se gerar apenas duas cláusulas:

(5) $\sim Q \vee R$ a partir de (1) e (2)
(6) $\sim P \vee R$ a partir de (1) e (3)

Ambas as cláusulas são satisfeitas por I, portanto são colocadas em S_2 e resolve-se elas com as cláusulas em S_1. Vê-se que (5) pode ser resolvida com (3) e (6) com (2), mas ambas resultam na mesma cláusula R:

(7) R a partir de (3) e (5)
(8) R a partir de (2) e (6)

A cláusula R é falsa em I, portanto é colocada em S_1 e resolve-se ela com as cláusulas em S_2. Encontra-se ~R em S_2. Portanto, resolvendo R com ~R, obtém-se a cláusula vazia □.

A figura 6.1 mostra as duas maneiras pelas quais R foi gerada. As duas deduções de R usam as cláusulas (1), (2) e (3) e diferem somente na ordem em que estas cláusulas são utilizadas. Uma vez que somente uma delas é necessária à prova, é bastante inútil gerar as duas. De modo a evitar esta redundância, introduz-se o conceito de *conflito*.

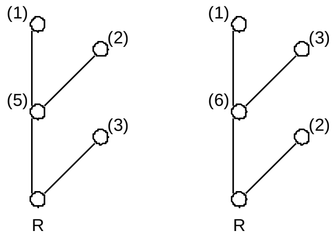

Figura 6.1 Conflito encontrado para a geração da cláusula R no Exemplo 6.2

A idéia para solucionar um conflito é gerar R diretamente de (1), (2) e (3) sem passar pela cláusula *intermediária* (5) ou (6). O conjunto {(1), (2), (3)} será chamado de conflito. Uma descrição mais elaborada de conflito será vista na próxima seção, onde será visto, também, que se pode combinar os conceitos de interpretação e ordenação de predicados com o de conflito para restringir ainda mais soluções.

6.3 DEFINIÇÕES FORMAIS E EXEMPLOS DE RESOLUÇÃO SEMÂNTICA

Definição: Sendo I uma interpretação e P uma ordenação de predicados. Um conjunto finito de cláusulas $\{E_1, \ldots, E_q, N\}$, q 1 1 é chamado *conflito semântico* com respeito a P e I (ou de maneira reduzida, PI-*conflito*), se e somente se $E_1, \ldots, E_q$ (chamados elétrons) e N (chamado núcleo) satisfizerem as seguintes condições:

1) $E_1, \ldots, E_q$ são falsos em I.
2) Sendo $R_1 = N$. Para cada $i = 1, \ldots, q$, existe um resolvente R_{i+1} de R_i e E_i.
3) O literal E_i, que será resolvido, contém o maior predicado em E_i, $i = 1, \ldots, q$.
4) R_{q+1} é falso em I.

R_{q+1} é chamado de PI-*resolvente* (do PI-conflito $\{E_1, \ldots, E_q, N\}$).

Exemplo 6.3: Sendo $E_1 = A_1 \vee A_3$, $E_2 = A_2 \vee A_3$, $N = \sim A_1 \vee \sim A_2 \vee A_3$, $I = \{\sim A_1, \sim A_2, \sim A_3\}$ e P uma ordenação na qual $A_1 > A_2 > A_3$. Então, $\{E_1, E_2, N\}$é um PI-conflito. o PI-resolvente deste conflito é A_3. Observe que A_3 é falso em I.

$R_1 = \sim A_1 \vee \sim A_2 \vee A_3$ $\qquad E_1 = A_1 \vee A_3$

$R_2 = \sim A_2 \vee A_3$ $\qquad E_2 = A_2 \vee A_3$

$R_3 = A_3$

Exemplo 6.4: No Exemplo 6.3, nem $\{E_1, N\}$ nem $\{E_2, N\}$ são PI-conflitos. Por exemplo, pode-se ver que o resolvente de E_1 é $\sim A_2 \vee A_3$ que é verdadeiro em I, assim $\{E_1, N\}$ não é um PI-conflito. Pode-se ver também que, se for mudada a ordenação tal que $A_3 > A_2 > A_1$, então $\{E_1, E_2, N\}$ não é um PI-conflito, uma vez que a terceira condição para que exista um PI-conlfito não é satisfeita.

Exemplo 6.5: Sendo $E_1 = \sim Q(z) \vee \sim Q(a)$,

$E_2 = \sim R(b) \vee S(c)$,

$N = Q(x) \vee Q(a) \vee \sim R(y) \vee \sim R(b) \vee S(c)$,

$I = \{Q(a), Q(b), Q(c), \sim R(a), \sim R(b), \sim R(c), \sim S(a), \sim S(b), \sim S(c)\}$ e P uma ordenação de predicados na qual $Q > R > S$. Sendo $R_1 = N$, existe um resolvente de R_1 e E_1, dito, $\sim R(y) \vee \sim R(b) \vee S(c)$, chamado de R_2. Então, existe outro resolvente de R_2 e E_2, dito S(c), chamado de R_3 que é falso em I. Deste modo $\{E_1, E_2, N\}$ satisfaz todas as quatro condições para ser um PI-conflito e tem como PI-resolvente S(c).

Em um PI-conflito $\{E_1, \ldots, E_q, N\}$ E_1 é considerado o primeiro elétron, E_2 o segundo e E_q o último. De fato, a ordem destes elétrons é irrelevante. Pode-se reordenar os elétrons obedecendo qualquer ordem e obter o mesmo PI-resolvente deste conflito. Portanto, pode-se evitar a geração de muitas deduções do mesmo resolvente, como mostrou o Exemplo 6.2. Esta é a razão para o uso do conceito de conflito.

Definição: Sendo I uma interpretação de um conjunto de cláusulas S e P uma ordenação de predicados em S. Uma dedução de S é chamada de PI-*dedução* se e somente se cada cláusula da dedução é uma cláusula em S ou um PI-resolvente.

Exemplo 6.6: Sendo $S = \{A_1 \vee A_2, A_1 \vee \sim A_2, \sim A_1 \vee A_2, \sim A_1 \vee \sim A_2\}$, $I = \{A_1, \sim A_2\}$ e P uma ordenação na qual $A_1 < A_2$. A dedução mostrada na figura 6.2 é uma PI-dedução de $\square$ a partir de S. Na figura 6.2, existem três PI-resolventes, ditos, $\sim A_1$, A_2, $\square$.

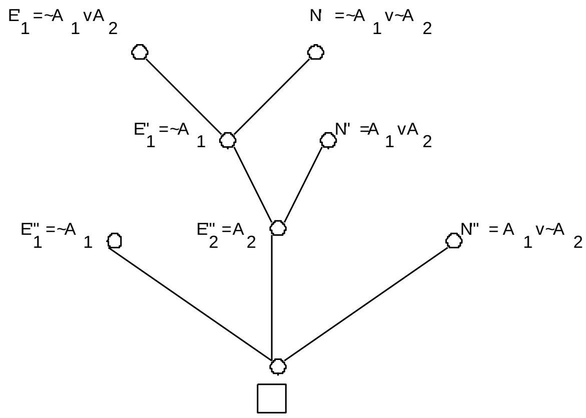

Figura 6.2 PI-dedução de $\square$ a partir de S, para o Exemplo 6.6

Um aspecto interessante sobre resolução semântica é que pode-se usar qualquer ordenação e qualquer interpretação. Observe, no Exemplo 6.6, tomando $I = \{\sim A_1, \sim A_2\}$ e P uma ordenação na qual $A_1 > A_2$, obtém-se uma PI-dedução de $\square$ a partir de S, como pode-se ver na figura 6.3. De fato, a PI-resolução é completa, ou seja, a partir de qualquer conjunto de cláusulas insatisfatível, pode-se sempre derivar a cláusula vazia $\square$, usando alguma PI-resolução. Isto será provado na próxima seção.

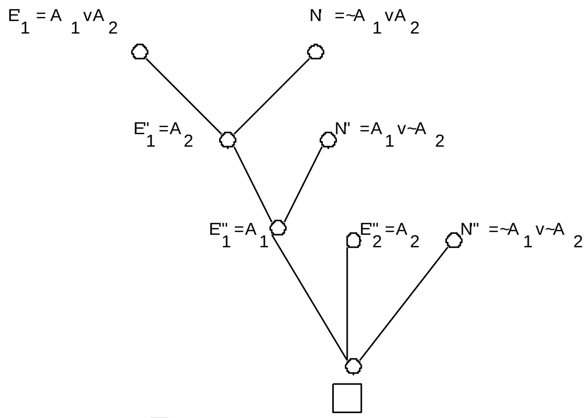

Figura 6.3 PI-dedução de □ a partir de S, para I = {~A_1, ~A_2} e P tal que: $A_1 > A_2$

6.4 COMPLETUDE DA RESOLUÇÃO SEMÂNTICA

Lema 6.1: Se P é uma ordenação de predicados em um conjunto finito insatisfatível de cláusulas base e, se I é uma interpretação de S, então existe uma PI-dedução de □ a partir de S.

Prova: Este lema pode ser provado por indução. Seja A o conjunto de átomos de S. Se A consiste de um único elemento, dito Q, então, entre os elementos de S, existem cláusulas Q e ~Q. Claramente se vê que o resolvente de Q e ~Q é □. Uma vez que Q ou ~Q deve ser falso em I, e □ é falso em I, □ é um PI-resolvente. Portanto, o Lema 6.1 é váilido para este caso.

Assumindo que o Lema 6.1 é válido quando A consiste de i elementos, 1 l i 1 n. Para completar a indução, considera-se A tal que A consiste de exatamente n+1 elementos. Com isto, existem dois casos possíveis:

CASO 1: S contém uma cláusula unitária L que é falsa em I (L é um literal). Sendo S' o conjunto obtido de S eliminando-se as cláusulas contendo o literal L e eliminando-se ~L das cláusulas restantes. Certamente, S' é insatisfatível (relembre a regra do literal-único da seção 4.6). Uma vez que S' contém n ou menos átomos, pela hipótese da indução, existe uma PI-resolução D' de □ a partir de S'. Da dedução D', pode-se obter uma PI-resolução de □ a partir de S. Isto é feito do seguinte modo: Primeiro, para cada PI-conflito {E'_1 , ... , E'_q , N'}, onde E'_1 , ... , E'_q , N', são cláusulas associadas a nós iniciais de D', se N' é obtido a partir da cláusula N em S pela eliminação de ~L de N, substitua o conflito {E'_1 , ... , E'_q , N'} pelo conflito {E'_1 , ... , E'_q , L, N'}, onde E'_1 , ... , E'_q , L são os elétrons e N o núcleo. Segundo, se E'_i é obtido da cláusula E_i em S pela eliminação de ~L de E_i , conecte o PI-conflito {L,E_i} acima do nó E'_i . Depois do processo acima descrito ser executado, fica claro que se pode obter uma PI-dedução de □ a partir de S.

CASO 2: S não contém uma cláusula unitária L que é falsa em I. Neste caso, escolha um elemento B no conjunto de átomos A de S tal que B contenha o menor predicado. B ou ~B deve ser falso em I. Sendo L o elemento em {B, ~B} que é falso em I e S' o conjunto obtido a partir de S eliminando aquelas cláusulas contendo o literal ~L e eliminando L das cláusulas restantes, S' é insatisfatível. Uma vez que S' contém n ou menos átomos, pela hipótese da indução, existe uma PI-resolução D' de □ a partir de S'. Sendo D_1 a dedução obtida de D', colocando-se o literal L de volta àquelas cláusulas das quais ele foi eliminado, D_1 ainda é uma PI-dedução, uma vez que L contém o menor predicado e L é falso em I. D_1 é uma PI-dedução de □ ou é L. Se D_1 for uma PI-dedução de □, chegou-se ao objetivo. Se D_1 for L, considere o conjunto (S 2 {L}) que contém a cláusula unitária L que é falsa em I, pela prova do Caso 1, vista anteriormente, existe uma PI-dedução D_2 de □ a partir de (S 2 {L}). Combinando D_1 e D_2 , pode-se obter PI-dedução de □ a partir de S. Isto completa a prova do Lema 6.1.

Teorema 6.1 (Completude da PI-resolução): Se P é uma ordenação dos predicados num conjunto finito e insatisfatível de cláusulas S, e se I é uma interpretação de S, então existe uma PI-dedução de □ a partir de S.
Prova: Uma vez que S é insatisfatível, pelo Teorema de Herbrand (versão II) existe um conjunto finito insatisfatível S' de instâncias base de cláusulas em S. Pelo Lema 6.1, existe uma PI-dedução D' de □ a partir de S'. Partindo PI-resolução D', pode-se mostrar agora que é possível produzir uma PI-dedução de □ a partir de S. Para se ver isto, apenas associa-se a cada nó de D' uma cláusula, acima ou abaixo da cláusula base já existente no local, da seguinte maneira: Para cada nó inicial, associe uma cláusula em S para a qual a cláusula base no local é sua instância. Então, para cada nó não inicial, se as cláusulas forma associadas deste modo para cada um de seus nós predecessores imediatos e constituem um PI-conflito, associe o PI-resolvente para o qual a cláusula base no local é sua instância (Isto é possível devido ao Lema 5.1). Deste modo, uma cláusula é associada a cada nó onde uma cláusula base é sua instância. A cláusula associada ao nó terminal deve ser □, uma vez que a cláusula já existente no local é a própria □. Fica fácil de se ver que a árvore de dedução, junto com as cláusulas associadas, nada mais é do que uma PI-dedução de □ a partir de S. Isto completa a prova.

Nota-se que, sem perda da propriedade de completude, a estratégia de eliminação vista na seção 5.8, pode ser usada em conjunto com a PI-resolução.

6.5 HIPERRESOLUÇÃO E ESTRATÉGIA DO CONJUNTO DE APOIO: CASOS ESPECIAIS DA RESOLUÇÃO SEMÂNTICA

Nesta seção vai-se introduzir dois casos especiais de interpretações que serão usados na resolução semântica. Uma destas interpretações dá origem a hiperresolução e outra a estratégia do conjunto de apoio.

6.5.1 Hiperresolução

Vai-se considerar uma interpretação I na qual todo literal é a negação de um átomo. Se esta interpretação é usada, todo elétron e todo PI-resolvente devem conter somente átomos. Do mesmo modo, se todo literal em I é um átomo, então todo elétron e todo PI-resolvente devem conter somente negações de átomos. Hiperresolução está baseada nestas considerações.

Definição: Uma cláusula é chamada de *positiva* se todos os seus literais não contém nenhum sinal de negação. Uma cláusula é chamada de *negativa* se todos os seus literais contém o sinal de negação. Uma cláusula é chamada de *mista* se não é positiva nem negativa.

Definição: Uma *hiperresolução positiva* é um caso especial de PI-resolução na qual todo literal da interpretação I contém o sinal de negação. Ela é chamada de hiperresolução positiva, pois todos os elétrons e PI-resolventes para este caso são positivos.

Definição: Uma *hiperresolução negativa* é um caso especial de PI-resolução na qual a interpretação I não contém nenhum sinal de negação. Ela é chamada de hiperresolução negativa, pois todos os elétrons e PI-resolventes para este caso são negativos.

Do Teorema 6.1, pode-se ver que as hiperresoluções, tanto negativa quanto positiva, são completas.

Exemplo 6.7: Para o seguinte conjunto de cláusulas:

$$S = \{Q(a) \vee R(x), Q(x) \vee R(x), \sim R(a) \vee \sim S(a), S(x)\}$$

seja P uma ordenação definida como R < Q < S. Então obtém-se as hiperresoluções positiva e negativa mostradas nas figuras 6.4(a) e 6.4(b), respectivamente.

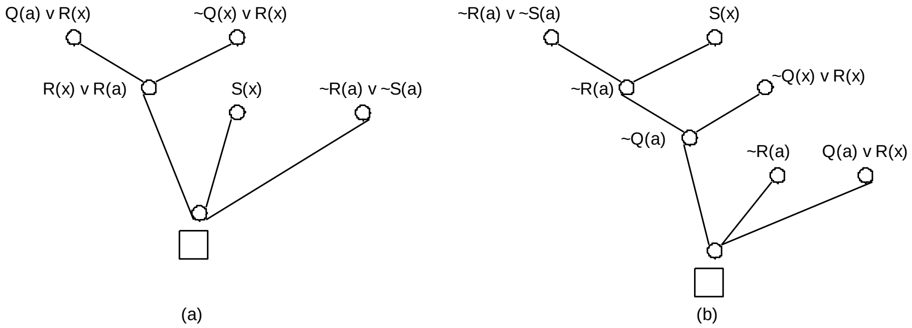

Figura 6.4 Hiperresoluções positiva (a) e negativa (b) para as cláusulas do Exemplo 6.7

Exemplo 6.8: Sendo os axiomas, Q, R e $\sim Q \vee \sim R \vee S$ e a negação da conclusão ~S. Tem-se quatro cláusulas:

(1) Q
(2) R
(3) $\sim Q \vee \sim R \vee S$
(4) ~S

Sendo a ordem dos predicados Q > R > S. Então a figura 6.5 (a) é uma hiperdedução positiva de □. Pode-se notar que a dedução parte dos axiomas e deriva S que é contraditório a negação da conclusão ~S. Na figura 6.5 (b) a dedução de □ é uma hiperdedução negativa. Neste caso a dedução inicia da negação da conclusão. Assumindo que a conclusão é falsa, deriva-se a contradição dessa premissa.

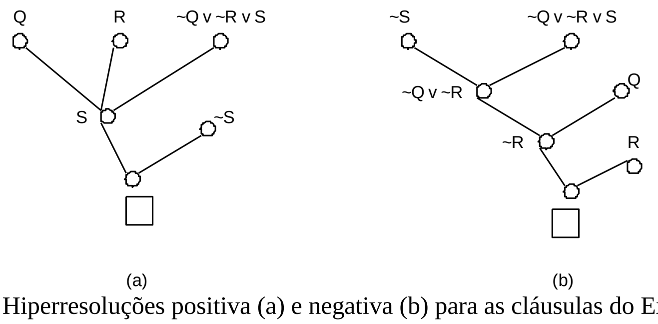

Figura 6.4 Hiperresoluções positiva (a) e negativa (b) para as cláusulas do Exemplo 6.8

6.5.2 Estratégia do Conjunto de Apoio

Como visto nos capítulos anteriores, um teorema consiste em um conjunto de axiomas A_1, A_2 , ... , A_n e uma conclusão B. Para provar teorema, tem-se que, essencialmente, provar que A_1 2 A_2 2 ... 2 A_n 2 ~B é insatisfatível. Uma vez que A_1 2 A_2 2 ... 2 A_n é, geralmente, satsifatível, pode ser considerado inteligente evitar resolver as cláusulas em A_1 2 A_2 2 ... 2 A_n entre si. Isto é o que o conjunto de apoio leva em conta.

Definição: Um subconjunto T de um conjunto S de cláusulas é chamado de *conjunto de apoio* de S se S - T é satisfatível. Uma *resolução por conjunto de apoio* é a resolução de duas cláusulas onde ambas não estão em S - T. Uma *dedução por conjunto de apoio* é uma dedução na qual toda resolução é uma resolução por conjunto de apoio.

Vai-se provar que a resolução por conjunto de apoio é completa. Isto é facilmente feito usando-se o Teorema 6.1.

Teorema 6.2 (Completude da Estratégia do Conjunto de Apoio): Se S é um conjunto finito insatisfatível de cláusulas e T é um subconjunto de S tal que S - T é satisfatível, então existe uma dedução por conjunto de apoio de $\square$ a partir de S tendo T como conjunto de apoio.
Prova: Uma vez que o conjunto S - T é satisfatível, existe uma interpretação I que satisfaz S - T. Escolhendo qualquer ordenação P dos predicados em S, pelo Teorema 6.1 existe uma PI-dedução D de $\square$ a partir de S. A partir de D, considere cada PI-conflito $\{E_1, \ldots, E_q, N\}$, onde todo resolvente envolve um elétron E_i, entretanto, todo elétron é falso em I. Portanto, para toda resolução correspondente, as duas cláusulas não podem estar ambas em S - T. Deste modo, a dedução D pode ser transformada em uma dedução por conjunto de apoio de $\square$ a partir de S.

Exemplo 6.9: Sendo S o conjunto consistindo das seguintes cláusulas:

(1) $P(g(x_1,y_1),x_1,y_1)$
(2) $\sim P(x_2,h(x_2,y_2),y_2)$
(3) $\sim P(x_3,y_3,u_3) \vee P(y_3,z_3,v_3) \vee \sim P(x_3,v_3,w_3) \vee P(u_3,z_3,w_3)$
(4) $\sim P(k(x_4),x_4,k(x_4))$

Sendo T um conjunto consistindo da cláusula (4). Então a seguinte dedução é uma dedução por conjunto de apoio com T sendo o conjunto de apoio. Vê-se que nenhuma resolução é efetuada entre (1), (2) e (3).
O primeiro resolvente gerado (5) é obtido pela resolução entre as cláusulas (4) e (3) e para obter o unificador mais geral entre as duas cláusulas, aplica-se o algoritmo de unificação entre os seguintes literais:

$\sim P(k(x_4),x_4,k(x_4))$	$\sim P(k(x_4),x_4,k(x_4))$	$\sim P(k(x_4),z_3,k(z_3))$	$\sim P(k(x_4),z_3,k(z_3))$
$P(u_3,z_3,w_3)$	$P(k(x_4),z_3,w_3)$	$P(k(x_4),z_3,w_3)$	$P(k(x_4),z_3,k(z_3))$
$1_0 = \{k(x_4)/u_3\}$	$1_1 = 1_0\ 1\{z_3/x_4\}$	$1_2 = 1_1\ 1\{k(z_3)/w_3\}$	
	$1_1 = \{k(z_3)/u_3,z_3/x_4\}$	$1_2 = \{k(z_3)/u_3,z_3/x_4,k(z_3)/w_3\} = 1$	

(5) $\sim P(x_3,y_3,k(z_3)) \vee P(y_3,z_3,v_3) \vee \sim P(x_3,v_3,k(z_3))$

O segundo resolvente gerado (6) é obtido pela resolução entre (5) e (2), aplicando-se o algoritmo de unificação entre os seguintes literais:

$P(y_3,z_3,v_3)$	$P(y_3,z_3,v_3)$	$P(y_3,h(y_3,y_2),v_3)$	$P(y_3,h(y_3,y_2),v_3)$
$\sim P(x_2,h(x_2,y_2),y_2)$	$\sim P(y_3,h(y_3,y_2),y_2)$	$\sim P(y_3,h(y_3,y_2),y_2)$	$\sim P(y_3,h(y_3,y_2),v_3)$
$1_0 = \{y_3/x_2\}$	$1_1 = 1_0\ 1\{h(y_3,y_2)/z_3\}$	$1_2 = 1_1\ 1\{v_3/y_2\}$	
	$1_1 = \{y_3/x_2,h(y_3,y_2)/z_3\}$	$1_2 = \{y_3/x_2,h(y_3,v_3)/z_3,v_3/y_2\} = 1$	

(6) $\sim P(x_3,y_3,k(h(y_3,v_3))) \vee \sim P(x_3,v_3,k(h(y_3,v_3)))$

O terceiro resolvente gerado (7) é obtido pela resolução entre (6) e (1), aplicando-se o algoritmo de unificação entre os seguintes literais:

$\sim P(x_3,y_3,k(h(y_3,v_3)))$	$\sim P(g(x_1,y_1),y_3,k(h(y_3,v_3)))$
$P(g(x_1,y_1),x_1,y_1)$	$P(g(x_1,y_1),x_1,y_1)$

$1_0 = \{g(x_1,y_1)/x_3\}$	$1_1 = 1_0\ 1\{y_3/x_1\}$
	$1_1 = \{g(y_3,y_1)/x_3,y_3/x_1\}$
$\sim P(g(x_1,y_1),y_3,k(h(y_3,v_3)))$	$\sim P(g(x_1,y_1),y_3,k(h(y_3,v_3)))$
$P(g(x_1,y_1),y_3,y_1)$	$P(g(x_1,y_1),y_3,k(h(y_3,v_3)))$

$1_2 = 1_1\ 1\{k(h(y_3,v_3))/y_1\}$

$1_2=\{g(y_3,k(h(y_3,v_3)))/x_3,y_3/x_1,k(h(y_3,v_3))/y_1\} = 1$

(7) $\sim P(g(y_3,k(h(y_3,v_3))),v_3,k(h(y_3,v_3)))$

O quarto resolvente gerado (8) é obtido pela resolução entre (7) e (1), aplicando-se o algoritmo de unificação entre os seguintes literais:

$\sim P(g(y_3,k(h(y_3,v_3))),v_3,k(h(y_3,v_3)))$	$\sim P(g(x_1,k(h(y_3,v_3))),v_3,k(h(y_3,v_3)))$
$P(g(x_1,y_1),x_1,y_1)$	$P(g(x_1,y_1),x_1,y_1)$
$1_0 = \{x_1/y_3\}$	$1_1 = 1_0\ 1\{k(h(y_3,v_3))/y_1\}$
	$1_1 = \{x_1/y_3,k(h(y_3,v_3))/y_1\}$
$\sim P(g(x_1,k(h(y_3,v_3))),v_3,k(h(y_3,v_3)))$	$\sim P(g(x_1,k(h(y_3,v_3))),v_3,k(h(y_3,v_3)))$
$P(g(x_1,k(h(y_3,v_3))),x_1,k(h(y_3,v_3)))$	$P(g(x_1,k(h(y_3,v_3))),v_3,k(h(y_3,v_3)))$

$1_2 = 1_1\ 1\{v_3/x_1\}$

$1_2 = \{v_3/y_3,k(h(y_3,v_3))/y_1,v_3/x_1\}$

(8) □

6.6 RESOLUÇÃO SEMÂNTICA USANDO CLÁUSULAS ORDENADAS

Aqui vai-se considerar algumas ordenações diferentes de ordenações de predicados. Para a lógica proposicional, ordenação de predicados usado em conjunto com resolução semântica se mostrou bastante eficiente. Para qualquer elétron E, não há dificuldade em escolher um literal em E para ser resolvido. Por exemplo, considere $E_1 = A_1 \vee A_3$, $E_2 = A_2 \vee A_3$, $N = \sim A_1 \vee \sim A_2 \vee A_3\}$, como no Exemplo 6.3. Usando $I = \{\sim A_1, \sim A_2, \sim A_3\}$ e a ordenação P onde $A_1 > A_2 > A_3$, pode-se facilmente decidir que os literais A_1 e A_2 em E_1 e E_2, respectivamente, devem ser resolvidos. Entretanto, para lógica de predicados, onde variáveis estão envolvidas, em um elétron E poderá existir mais de um literal que contenha o mesmo predicado em E. Qualquer um destes literais é candidato a ser resolvido. Isto fica melhor ilustrado através de um exemplo.

Exemplo 6.10: Considere: $E = Q(a) \vee Q(b) \vee Q(c) \vee Q(d)$

$N = \sim Q(x)$

Sendo I uma interpretação na qual todo literal é negativo e P uma ordenação qualquer dos predicados, fica claro que {E,N} é um PI-conflito. Uma vez que todos os literais em E contém o mesmo predicado Q, qualquer um deles pode ser resolvido com N. Portanto do PI-conflito {E,N}, pode-se obter quatro PI-resolventes, que são, $Q(b) \vee Q(c) \vee Q(d)$, $Q(a) \vee Q(c) \vee Q(d)$, $Q(a) \vee Q(b) \vee Q(d)$, $Q(a) \vee Q(b) \vee Q(c)$.

Do exemplo acima, pode-se ver que, usando uma ordenação dos predicados, há a possibilidade de não se conseguir escolher um único literal para ser resolvido em um elétron. Assim, pode-se ser forçado a gerar mais que um resolvente semântico de um conflito semântico. Para resolver esta situação, considera-se *cláusulas ordenadas*.

A idéia de cláusulas ordenadas é considerar uma cláusula como uma seqüência de literais ao invés de um conjunto de literais. Para se fazer isto, atribui-se uma ordem a todos os literais na cláusula. Convenciona-se que um literal L_2 é maior que um literal L_1 em uma cláusula no caso de L_2 seguir L_1 na seqüência especificada pela cláusula. Deste modo, o último literal em uma cláusula ordenada, será sempre considerado o maior literal na cláusula. Por exemplo, considerando $A_3 \vee A_1 \vee A_2$ como uma cláusula ordenada, então A_2 é o maior literal.

Dar-se-á agora as definições formais.

Definição: Uma *cláusula ordenada* é uma seqüência de literais distintos.

Da mesma maneira que uma cláusula, uma cláusula ordenada é também interpretada como uma disjunção de todos os literais que compõem a cláusula ordenada. A única diferença é que a ordem dos literais em uma cláusula é irrelevante, enquanto que a ordem dos literais na cláusula ordenada é especificada. Uma cláusula ordenada também será escrita como uma disjunção de literais. Isto não causará confusão, pois sempre se usará a palavra *ordenada* para fazer referência a cláusulas ordenadas.

Definição: Um literal L_2 é dito ser maior que um literal L_1 em uma cláusula ordenada (ou L_1 é menor que L_2) se e somente se L_2 segue L_1, na seqüência especificada pela cláusula ordenada.

Pode-se considerar resolução semântica usando cláusulas ordenadas. Entretanto, define-se primeiro a resolução ordenada para cláusulas ordenadas.

Definição: Se dois ou mais literais (com o mesmo sinal) de uma cláusula C, tem um unificador mais geral 1, então a cláusula ordenada é obtida da seqüência C1, pela eliminação de todo literal que é idêntico a um literal menor na seqüência, é chamada de *fator ordenado* de C.

Exemplo 6.11: Considere a cláusula ordenada C = $\underline{P(x)} \vee Q(x) \vee \underline{P(a)}$, o primeiro e o último literal, sublinhados, de C tem um unificador mais geral 1 = {a/x}. Portanto, C1 = $P(a) \vee Q(a) \vee P(a)$. Na seqüência C1, existem dois literais idênticos. Uma vez que o segundo P(a) é idêntico a um literal menor, ou seja, ao primeiro P(a), elimina-se o segundo P(a) de C1 e obtém-se a seqüência $P(a) \vee Q(x)$, que é um fator ordenado de C. Observa-se que $Q(a) \vee P(a)$ não é um fator ordenado de C, pois foi obtido de C1 pela eliminação do primeiro P(a).

Definição: Sendo C_1 e C_2 cláusulas ordenadas, sem variáveis em comum e L_1 e L_2 dois literais em C_1 e C_2 , respectivamente. Se L_1 e $\sim L_2$ tem um unificador mais geral 1 e se C é a cláusula ordenada obtida concatenando-se as seqüências $C_1 1$ e $C_2 1$, removendo $L_1 1$ e $L_2 1$ e eliminando todo literal que é idêntico a um literal menor na seqüência restante, então C é chamado de *resolvente binário ordenado* de C_1 contra C_2. Os literais L_1 e L_2 são chamados de literais resolvidos.

A definição acima, é idêntica a definição do resolvente binário dada na Seção 5.5 do Capítulo 5, com exceção de que a ordem dos literais em um resolvente binário ordenado está, deliberadamente, especificada. Note que usou-se a palavra *contra* para enfatizar que um resolvente binário ordenado de C_1 contra C_2 não é o mesmo que C_2 contra C_1.

Exemplo 6.12: Considere as seguintes cláusulas ordenadas:

$$C_1 = P(x) \vee Q(x) \vee R(x) \text{ e } C_2 = \sim P(a) \vee Q(a).$$

Escolha $L_1 = P(x)$ e $L_2 = \sim P(a)$. Uma vez que $\sim L_2 = P(a)$, L_1 e $\sim L_2$ tem o mesmo unificador mais geral σ = {a/x}. Inicialmente, concatenando as seqüências $C_1\sigma$ e $C_2\sigma$, obtém-se a seqüência $P(a) \vee Q(a) \vee R(a) \vee \sim P(a) \vee Q(a)$. Num segundo momento, remove-se $L_1\sigma$ e $L_2\sigma$, da seqüência anterior, obtendo $Q(a) \vee R(a) \vee Q(a)$. Finalmente, uma vez que em $Q(a) \vee R(a) \vee Q(a)$, o segundo Q(a) é idêntico ao menor literal, ou seja, o primeiro Q(a), elimina-se o segundo Q(a) e obtém-se $Q(a) \vee R(a)$. A seqüência $Q(a) \vee R(a)$ é um resolvente binário ordenado de C_1 contra C_2 e P(x) e ~P(a) são os literais resolvidos.

Definição: Um *resolvente ordenado* de uma cláusula ordenada C_1 contra uma cláusula ordenada C_2 é um dos seguintes resolventes binários ordenados:

1. um resolvente binário ordenado de C_1 contra C_2,
2. um resolvente binário ordenado de C_1 contra um fator ordenado de C_2,
3. um resolvente binário ordenado de um fator ordenado de C_1 contra de C_2,
4. um resolvente binário ordenado de um fator ordenado de C_1 contra um fator ordenado de C_2 .

Exemplo 6.13: Considere o seguinte conjunto de cláusulas ordenado: $C_1 = P(x) \vee Q(x) \vee R(x) \vee P(a)$ e $C_2 = {\sim}P(a) \vee Q(a)$. Um fator ordenado de C_1 e $C_1' = P(a) \vee Q(a) \vee R(a)$. Um resolvente binário ordenado de C_1' contra C_2 é $Q(a) \vee R(a)$. Portanto, $Q(a) \vee R(a)$ é um resolvente binário ordenado de C_1 contra C_2.

Resolução ordenada é uma regra de inferência que gera resolventes ordenados a partir de um conjunto ordenado de cláusulas. Não é difícil de se ver que a resolução ordenada é completa. Isto é, resolução ordenada sempre gerará a cláusula vazia □ de um conjunto insatisfatível de cláusulas ordenadas. A prova ficará como exercício.

Tendo definido resolução ordenada, vai-se considerar agora, resolução semântica para cláusulas ordenadas. Deve-se, ainda, utilizar conceitos de interpretação e conflito. Entretanto, deve-se usar o conceito de cláusulas ordenadas ao invés de ordenação de predicados. Como discutido no início desta seção, isto representa uma vantagem, pois agora pode-se escolher um único literal em um elétron para ser resolvido.

Definição: Sendo I uma interpretação. Uma seqüência de cláusulas ordenadas $\{E_1, E_2, \ldots, E_q, N\}$, $q \geq 1$, é chamado de *conflito ordenado semântico* com respeito a I (ou OI-conflito, resumidamente) se e somente se $E_1, E_2, \ldots, E_q$, (chamados *elétrons ordenados*) e N (chamado *núcleo ordenado*) satisfazendo as seguintes condições:

1. $E_1, E_2, \ldots, E_q$, são falsos em I.
2. Sendo $R_q = N$, para cada i = q, q-1, ... , 1, existe um resolvente ordenado R_{i-1} de E_i contra R_i.
3. O literal E_i que é resolvido é o "último literal" em E_i , i = 1, ..., q; o literal R_i que é resolvido é o "maior literal" que tem uma instância verdadeira em I.
4. R_0 é falso em I.

R_0 é chamado de OI-*resolvente* do OI-conflito $\{E_1, E_2, \ldots, E_q, N\}$.

Com exceção da condição (3), a definição acima é semelhante a de PI-resolvente dada na Seção 6.3. Ver-se-á agora um exemplo para ilustrar como um OI-conflito poderá ser encontrado.

Exemplo 6.14: Considere as seguintes cláusulas ordenadas:

(1) $A_1 \vee A_2$

(2) $A_1 \vee A_3$

(3) $\sim A_3 \vee \sim A_2 \vee A_1$
Sendo I = $\{\sim A_1, \sim A_2, \sim A_3\}$. Uma vez que (1) e (2) são falsos em I, eles podem ser usados como elétrons ordenados. Uma vez que (3) tem um literal que é verdadeiro em I, (3) pode ser usado como um núcleo ordenado. Portanto, N = $\sim A_3 \vee \sim A_2 \vee A_1$. Em N existem dois literais, $\sim A_3$ e $\sim A_2$ que são verdadeiros em I. Então, necessita-se dois elétrons ordenados. Sendo R_2 = N, isto é, $R_2 = \sim A_3 \vee \sim A_2 \vee A_1$. Em R_2, o maior literal que tem uma instância (neste caso, ela mesma) verdadeira em I é $\sim A_2$. Uma vez que $\sim A_2$ e o último literal de (1) podem ser resolvidos, tem-se $E_2 = A_1 \vee A_2$. O resolvente ordenado R_1 de E_2 contra R_2 é dado como $R_1 = \sim A_3 \vee A_1$. Agora, em R_1, o maior literal que tem uma instância (neste caso, ela mesma) verdadeira em I é $\sim A_3$. Uma vez que $\sim A_3$ e o último literal de (2) podem ser resolvidos, tem-se $E_1 = A_1 \vee A_3$. O resolvente ordenado R_0 de E_1 contra R_1 é dado como $R_0 = A_1$. Uma vez que R_0 é falso em I, conclui-se que $\{E_1, E_2, N\}$, ou {(1), (2), (3)} é um OI-conflito e A_1 é o OI-resolvente deste conflito.

Do exemplo acima, pode-se ver que a ordem do conjunto ordenado de elétrons é determinado pela ordem dos literais no núcleo ordenado. Portanto, para o exemplo acima {(2), (1), (3)} poderia não ser um OI-conflito.

Exemplo 6.15: Considere o conjunto de cláusulas dadas no Exemplo 6.10:

(1) $Q(a) \vee Q(b) \vee Q(c) \vee Q(d)$
(2) $\sim Q(x)$

Supondo que se trata (1) e (2) como cláusulas ordenadas. Sendo I uma interpretação na qual todo literal é negativo. Uma vez que (1) é falso em I, (1) pode ser usado como elétron. Resolvendo o último literal de (1) contra $\sim Q(x)$, obtém-se o resolvente $Q(a) \vee Q(b) \vee Q(c)$, que é falso em I. Portanto, {(1), (2)} é um OI-conflito com (1) e (2) sendo o elétron ordenado e o núcleo ordenado, respectivamente. Neste caso existe apenas um OI-resolvente deste conflito ao invés dos quatro PI-resolventes que se teria usando ordenação de predicados.

Definição: Sendo I uma interpretação para um conjunto ordenado de cláusulas S. Uma dedução de S é chamada OI-dedução se e somente se cada cláusula ordenada em na dedução é uma cláusula ordenada em S ou um OI-resolvente.

Exemplo 6.16: Considere o seguinte conjunto ordenado de cláusulas:

(1) $Q(a) \vee R(x)$
(2) $\sim Q(x) \vee R(x)$
(3) $\sim R(x) \vee \sim S(a)$
(4) $S(x)$

Sendo I uma interpretação na qual todo literal é negativo, então a dedução mostrada na figura 6.6, é uma OI-dedução de $\square$ a partir de S.

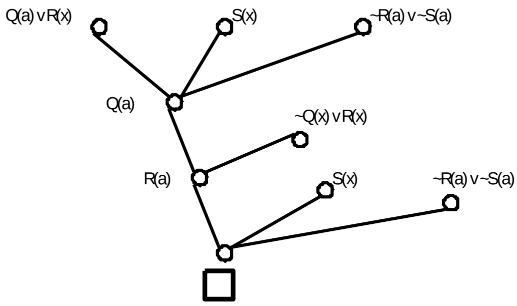

Figura 6.6 OI-Dedução de □ a partir de S do Exemplo 6.16.

Exemplo 6.17: Considere o conjunto S consintindo das cláusulas ordenadas Q(a) ∨ Q(b) ∨ Q(c) ∨ Q(d) e ~Q(x), como dado no Exemplo 6.10. S é insatisfatível e I pode ser uma interpretação na qual todo literal é negativo. Então a dedução mostrada na figura 6.7 é uma OI-Dedução de □ a partir de S. Somente quatro OI-resolventes ordenados foram gerados. Entretanto, se fosse utilizada a PI-resolução, 40 PI-resolventes poderiam ser gerados pelo método de saturação, antes da cláusula vazia □ ser encontrada.

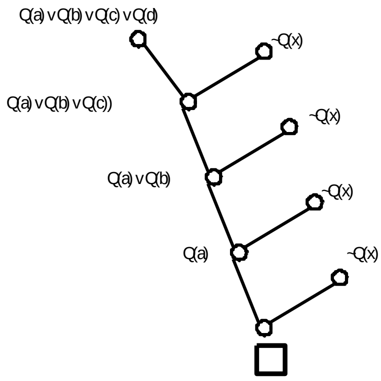

Figura 6.7 OI-Dedução de □ a partir de S do Exemplo 6.17.

Vários pesquisadores utilizaram OI-resolução e indicaram que seus resultados foram bastante eficientes. Muitos teoremas foram provados pela OI-resolução, porém, infelizmente, OI-resolução não é completa. Para que se possa verr isso, observe o seguinte contra-exemplo.

Exemplo 6.18: Considere o seguinte conjunto ordenado de cláusulas S:

(1) P ∨ Q
(2) Q ∨ R
(3) R ∨ W
(4) ~R ∨ ~P
(5) ~W ∨ ~Q
(6) ~Q ∨ ~R

Sendo I uma interpretação onde todo literal é negativo, as cláusulas (1)-(3) podem ser utilizadas como elétrons ordenados e as cláusulas (4)-(6) podem ser usadas como núcleos ordenados. Das cláusulas (1)-(6), pode-se obter os seguintes OI-resolventes:

(7) $R \vee P$ do OI-conflito {(3),(1),(5)}

(8) $P \vee Q$ do OI-conflito {(1),(2),(6)}

A partir das cláusulas (1)-(8) pode-se obter o seguinte OI-resolvente:

(9) $Q \vee R$ do OI-conflito {(2),(7),(4)}

Observa-se que as cláusulas (8) e (9) estão em S. Portanto, das cláusulas (1)-(9) não se pode produzir nenhum OI-resolvente novo. Isto é, a cláusula vazia $\square$, não poderá ser produzida por OI-resolução, apesar de S ser insatisfatível. Portanto OI-resolução não é completa.

6.7 IMPLEMENTAÇÃO DA RESOLUÇÃO SEMÂNTICA

Nesta seção, considera-se como implementar a PI-resolução. Apesar da OI-resolução não ser completa, pode-se utilizar o conceito de cláusulas ordenadas para implementar PI-resolução. Pode-se dizer que uma *cláusula positiva ordenada* é uma cláusula ordenada que não possui nenhum sinal de negação e uma *cláusula negativa ordenada* é uma cláusula ordenada na qual todos os sinais são de negação. Uma *cláusula não-positiva (não-negativa) ordenada* é uma cláusula ordenada que é não-positiva (negativa). De maneira a evitar que se estabeleça uma interpretação toda a vez que se usar uma PI-resolução, considerar-se-á somente hiperresolução positiva. (hiperresolução negativa poderá ser tratada de maneira semelhante). Deste modo, considera-se somente cláusulas positivas ordenadas como candidatas a elétrons e as não-positivas como candidatas a núcleo. *Convenciona-se que para quaisquer cláusulas ordenadas não-positivas, os literais negativos são colocados depois dos literais positivos. Seja S um conjunto de cláusulas ordenadas e P uma ordenação dos predicados em S.*O seguinte algoritmo pode calcular hiperresolventes positivos.

Geração de Hiperresolventes Positivos

Passo 0: Faça M e N o conjunto de todas as cláusulas positivas e não-positivas em S, respectivamente.

Passo 1: Faça j = 1.

Passo 2: Faça $A_0 = \emptyset$ e $B_0 = N$.

Passo 3: Faça i = 0.

Passo 4: Se A_i contém $\square$, pare; uma contradição foi encontrada. Caso contrário vá para o próximo passo.

Passo 5: Se B_i é vazio, vá para o Passo 8. Caso contrário, vá para o próximo passo.

Passo 6: Calcule o conjunto:

W_{i+1} = { resolventes ordenados de C_1 contra C_2, onde C_1 é uma cláusula ordenada ou um fator ordenado de uma cláusula ordenada em M, C_2 é uma cláusula ordenada em B_i , o literal resolvido de C_1 contém o "maior" predicado em C_1 e o literal resolvido de C_2 é o "último" literal de C_2 }.

Faça A_{i+1} e B_{i+1} iguais aos conjuntos de todas as cláusulas ordenadas positivas e não-positivas em W_{i+1} , respectivamente.

Passo 7: Faça i = i + 1 e vá para o Passo 4.

Passo 8: Faça $T = A_0 \cup ... \cup A_i$ e $M = T \cup M$.

Passo 9: Faça j = j + 1.

Passo 10: Calcule o conjunto:

R = {resolventes ordenados de C_1 contra C_2, onde C_1 é uma cláusula ordenada ou um fator ordenado de uma cláusula ordenada em T, C_2 é uma cláusula ordenada em N, o literal resolvido de C_1 contém o "maior" predicado em C_1}.

(Observe que na definição acima de R, o literal resolvido de C_2 pode ser qualquer literal, não necessariamente o último literal de C_2). Faça A_0 e B_0 iguais aos conjuntos de todas as cláusulas ordenadas positivas e não-positivas em R, respectivamente.

Passo 11: Vá para o Passo 3.

No algoritmo acima, para cada ciclo, isto é, para cada j, B_i poderá, eventualmente, ser vazio, uma vez que o número máximo de literais negativos em uma cláusula ordenada de B_i decresce de uma unidade cada vez que i aumenta de uma unidade. Todas as cláusulas em cada A_i são hiperresolventes positivos. Não é difícil de se ver que se S é insatisfatível, então $\square$ pode ser gerado a partir do algoritmo acima. A estratégia de eliminação também pode ser incorporada ao algoritmo acima sem perda da propriedade de completude. Isto é, para T e M obtidos no Passo 8 do algoritmo, qualquer cláusula em T ou M englobadas por outras cláusulas em T ou M podem ser eliminadas. (Observe que não há tautologias em T ou M, uma vez que todas as cláusulas em T ou M são positivas). Entretanto, no exemplo a seguir não se usa a estratégia de eliminação.

Exemplo 6.19: Seja S um conjunto de cláusulas ordenadas definido como S = $\{P_6 \vee P_4$, $P_5 \vee P_4$, $P_4 \vee P_1$, $\sim P_1 \vee \sim P_2$, $P_3 \vee \sim P_6$, $\sim P_4$, $P_2 \vee \sim P_5 \vee \sim P_3\}$. Para simplificar a notação, representaremos P_1 , P_2 , ... , P_6 por 1, 2, ... , 6, respectivamente. Portanto, S é escrito como S = $\{6 \vee 4 , 5 \vee 4 , 4 \vee 1 , \sim 1 \vee \sim 2 , 3 \vee \sim 6 , \sim 4 , 2 \vee \sim 5 \vee \sim 3\}$. Seja P uma ordenação de predicados na qual $1 < 2 < 3 < 4 < 5 < 6$. A partir de S, obtém-se:

M = $\{6 \vee 4 , 5 \vee 4 , 4 \vee 1\}$

N = $\{\sim 1 \vee \sim 2 , 3 \vee \sim 6 , \sim 4 , 2 \vee \sim 5 \vee \sim 3\}$, e pode-se aplicar o algoritmo anterior:

1. j = 1.
   a. $A_0 = \emptyset$
      $B_0 = N = \{\sim 1 \vee \sim 2 , 3 \vee \sim 6 , \sim 4 , 2 \vee \sim 5 \vee \sim 3\}$.
   b. Uma vez que nem A_0 contém $\square$ nem B_0 é vazio aplica-se o Passo 6 e obtém-se: $W_1 = \{4 \vee 3 , 1\}$
      $A_1 = \{4 \vee 3 , 1\}$
      $B_1 = \emptyset$.
   c. Uma vez que B_1 é vazio, vai-se para o Passo 8, e faz-se:
      $T = A_0 \cup A_1 = \{4 \vee 3 , 1\}$
      $M = T \cup M = \{4 \vee 3 , 1 , 6 \vee 4 , 5 \vee 4 , 4 \vee 1\}$
2. j = 2.
   a. Calculando R, obtém-se R = $\{3 , \sim 2\}$. A partir de R, obtém-se: $A_0 = \{3\}$
      $B_0 = \{\sim 2\}$.
   b. Uma vez que, nem A_0 contém $\square$ nem B_0 é vazio aplica-se o Passo 6 e obtém-se: $W_1 = A_1 = B_1 = \emptyset$.
   c. Uma vez que B_1 é vazio, faz-se:
      $T = A_0 \cup A_1 = \{3\}$
      $M = T \cup M = \{3 , 4 \vee 3 , 1 , 6 \vee 4 , 5 \vee 4 , 4 \vee 1\}$
3. j = 3.
   a. Aplicando o Passo 10, obtém-se R = $\{2 \vee \sim 5\}$, $A_0 = \emptyset$, $B_0 = \{2 \vee \sim 5\}$.
   b. Uma vez que, nem A_0 contém $\square$ nem B_0 é vazio aplica-se o Passo 6 e obtém-se: $W_1 = \{4 \vee 2\}$, $A_1 = \{4 \vee 2\}$, $B_1 = \emptyset$.
   c. Uma vez que B_1 é vazio, faz-se:

$T = A_0 \cup A_1 = \{ 4 \vee 2 \}$

$M = T \cup M = \{4 \vee 2 , 3 , 4 \vee 3 , 1 , 6 \vee 4 , 5 \vee 4 , 4 \vee 1 \}$

4. j = 4.
   a. Aplicando o Passo 10, obtém-se $R = \{2\}$, $A_0 = \{2\}$, $B_0 = \emptyset$.
   b. Uma vez que B_0 faz-se:
      $T = A_0 = \{2\}$
      $M = T \cup M = \{2 , 4 \vee 2 , 3 , 4 \vee 3 , 1 , 6 \vee 4 , 5 \vee 4 , 4 \vee 1 \}$
5. j = 5.
   a. Aplicando o Passo 10, obtém-se $R = \{\sim 1\}$, $A_0 = \emptyset$, $B_0 = \{\sim 1\}$.
   b. Uma vez que, nem A_0 contém □ nem B_0 é vazio aplica-se o Passo 6 e obtém-se: $W_1 = \{ \square \}$, $A_1 = \{ \square \}$, $B_1 = \emptyset$.
   c. Uma vez que A_1 contém □ o algoritmo termina e uma contradição foi encontrada.

Foram gerados seis hiperresolventes positivos, que são, □ , 2 , $4 \vee 2$, 3 , $4 \vee 3$ e 1. Para conseguir a hiperdedução de □, inicialmente, se traça os ancestrais de □, como mostrado na figura 6.8, que pode, então, ser facilmente transformada na hiperdedução mostrada na figura 6.9. Nota-se que todos os hiperresolventes gerados pelo algoritmo foram usados na prova. Neste exemplo, nenhum resolvente redundante ou irrelevante foi gerado pelo algoritmo.

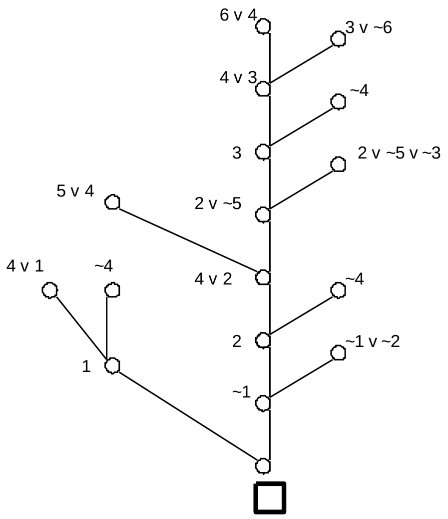

Figura 6.8 Ancestrais de □

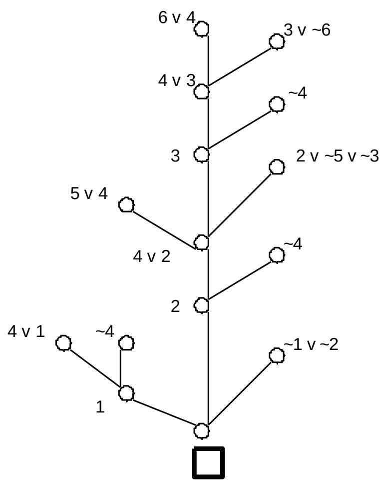

Figura 6.9 Hiperdedução de □

6.8 RESOLUÇÃO INDEXADA

Resolução Indexada é um refinamento da resolução que usa um conceito semelhante ao de cláusulas ordenadas. Dado um conjunto de cláusulas S, a idéia de resolução indexada é, essencialmente, usar índices para ordenar os literais das cláusulas em S. Isto é, ela faz uma indexação arbitrária de cada ocorrência de um literal em S através de um inteiro; diferentes ocorrências de um mesmo literal podem ser indexadas diferentemente. Resolução é permitida somente com literais de menor índice em cada cláusula. Os literais dos resolventes, herdam seus índices das cláusulas pai. Se um literal em um resolvente tem mais de um índice que pode ser herdado, o menor índice é atribuído a este literal. Antes de se dar uma definição formal de resolução indexada, consederar-se-á o seguinte exemplo:

Exemplo 6.20: Considere o seguinte conjunto de cláusulas:

(1) $_1P \vee {}_2Q$

(2) $_3{\sim}P \vee {}_4Q$.

O inteiro abaixo do literal é o índice associado a este literal. Os literais P e Q da cláusula (1) tem índices 1 e 2, respectivamente. Deve-se usar $_1P$ e $_2Q$ para se referir aos literais P e Q da cláusula (1), respectivamente. Da mesma maneira, deve usar $_3{\sim}P$ e $_4Q$ para se referir aos literais ~P e Q da cláusula (2), respectivamente. Uma vez que o índice de $_1P$ é menor que o de $_2Q$, $_1P$ é passível de ser resolvido. Do mesmo modo, uma vez que o índice de $_3{\sim}P$ é menor que o de $_4Q$, $_3{\sim}P$ pode ser resolvido. Assim, resolvendo as cláusulas (1) e (2) com os literais $_1P$ e $_3{\sim}P$, obtém-se a seguinte cláusula:

(3) $_2Q \vee {}_4Q$.

Agora $_2Q$ e $_4Q$ representam o mesmo literal Q. Uma vez que 2 é menor que 4, Q é indexado com 2. Portanto, obtém-se:

(4) $_2Q$.

A cláusula (4) é chamada de resolvente indexado das cláusulas (1) e (2). Observe que se os literais da cláusula (2) forem indexados de maneira diferente, tal como:

(2^*) $_4{\sim}P \vee {}_3Q$.

então o literal da cláusula (2^*) passível de ser resolvido é $_3Q$. Entretanto, $_1P$ e $_3Q$ não podem ser resolvidos, portanto, não existe resolvente indexado das cláusulas (1) e (2^*).

Definição: Sendo C uma cláusula onde cada um dos seus literais tem um índice inteiro. Se dois ou mais literais (com o mesmo sinal) de C tem um unificador mais geral σ, então as cláusulas obtidas a partir de Cσ pela eliminação de todo literal que é idêntico a um literal de índice mais baixo é chamado de *fator indexado* de C.

Exemplo 6.21: Seja C = $_2P(x) \vee {}_8Q(a) \vee {}_{11}P(a) \vee {}_5Q(x)$. $_2P(x)$ e $_{11}P(a)$ tem um unificador mais geral σ = {a/x}. Assim, Cσ = $_2P(a) \vee {}_8Q(a) \vee {}_{11}P(a) \vee {}_5Q(a)$. Em Cσ, exceto pelo índice, $_2P(a)$ é idêntico a $_{11}P(a)$. Uma vez que $_{11}P(a)$ tem um índice maior que $_2P(a)$, $_{11}P(a)$ será eliminado. Do mesmo modo, uma vez que $_8Q(a)$ tem um índice maior que $_5Q(a)$, $_8Q(a)$ será eliminado. Portanto eliminando $_{11}P(a)$ e $_8Q(a)$, obtém-se $_2P(a) \vee {}_5Q(a)$ que é um fator indexado de C.

Em uma cláusula, se existe mais de um literal idêntico, sempre se mantém aquele que tiver o menor índice e elimina-se todos os outros literais idênticos. Esta operação é denominada *agrupamento inferior* para literais idênticos. No exemplo anterior, foi feito o agrupamento inferior para $_2P(a)$ e $_{11}P(a)$ e, também, para $_5Q(a)$ e $_8Q(a)$.

Definição: Sendo C_1 e C_2 duas cláusulas sem nenhuma variável em comum e com todos seus literais indexados e L_1 e L_2 os dois literais de menor índice em C_1 e C_2 , respectivamente. Se L_1 e $\sim L_2$ tem o mesmo unificador mais geral σ e se C é a cláusula obtida a partir de ($C_1\sigma \vee C_2\sigma$) pela eliminação de $L_1\sigma$ e $\sim L_2\sigma$ e fazendo o agrupamento inferior de todos os literais idênticos na cláusula restante, então C é chamado de *resolvente binário indexado* de C_1 e C_2 . Os literais L_1 e L_2 são chamados literais resolvidos.

Exemplo 6.22: Considere as cláusulas: $C_1 = {}_1P(x) \vee {}_2Q(x) \vee {}_3R(x)$ e $C_2 = {}_4{\sim}P(a) \vee {}_5Q(a)$. Uma vez que ${}_1P(x)$ e ${}_4{\sim}P(a)$ tem os menores índices em C_1 e C_2 , respectivamente, escolhe-se $L_1 = P(x)$ e $L_2 = {\sim}P(a)$. L_1 e ${\sim}L_2$ tem o mesmo unificador mais geral $\sigma = \{a/x\}$. Portanto, $(C_1\sigma \vee C_2\sigma) = {}_1P(a) \vee {}_2Q(a) \vee {}_3R(a) \vee {}_4{\sim}P(a) \vee {}_5Q(a)$. Eliminando L_1 e ${\sim}L_2$, isto é, ${}_1P(a)$ e ${}_4{\sim}P(a)$, de $(C_1\sigma \vee C_2\sigma)$, obtém-se ${}_2Q(a) \vee {}_3R(a) \vee {}_5Q(a)$. Agora, ${}_2Q(a)$ e ${}_5Q(a)$ são literais idênticos com índices diferentes. Agrupando inferiormente ${}_2Q(a)$ e ${}_5Q(a)$, obtém-se ${}_2Q(a) \vee {}_3R(a)$, que é um resolvente binário indexado de C_1 e C_2 . ${}_1P(a)$ e ${}_4{\sim}P(a)$ são os literais resolvidos.

Definição: Sendo C_1 e C_2 duas cláusulas com todos seus literais indexados. Um *resolvente indexado* de C_1 e C_2 é um dos seguintes resolventes binários indexados:

1. um resolvente binário indexado de C_1 contra C_2,
2. um resolvente binário indexado de C_1 contra um fator indexado de C_2,
3. um resolvente binário indexado de um fator indexado de C_1 contra de C_2,
4. um resolvente binário indexado de um fator indexado de C_1 contra um fator indexado de C_2 .

Definição: Sendo S um conjunto de cláusulas onde todo literal em S é indexado com um inteiro. Uma dedução a partir de S é chamada de *dedução indexada* se e somente se toda cláusula na dedução é uma cláusula em S ou um resolvente indexado.

Exemplo 6.23: Considere o seguinte conjunto de cláusulas S:

(1) ${}_1P \vee {}_2Q$

(2) ${}_3P \vee {}_4{\sim}Q$

(3) ${}_6{\sim}P \vee {}_5Q$

(4) ${}_8{\sim}P \vee {}_7{\sim}Q$.

A partir das cláusulas (1)-(4), existe somente um resolvente indexado:

(5) ${}_6{\sim}P$ de (3) e (4)

A partir das cláusulas (1)-(5), existem somente dois resolventes indexados:

(6) ${}_2Q$ de (1) e (5)

(7) ${}_4{\sim}Q$ de (2) e (5)

Finalmente, resolvendo (6) e (7), obtém-se:

(8) $\square$.

Assim, obtém-se uma dedução indexada de $\square$ mostrada na figura 6.10. Note que, no todo, somente três resolventes indexados foram gerados. Entretanto, se a resolução ordinário (sem refinamentos) fosse utilizada, 37 resolventes poderiam ser gerados pelo método de saturação antes de $\square$ ser gerado. (Veja Seção 5.8)

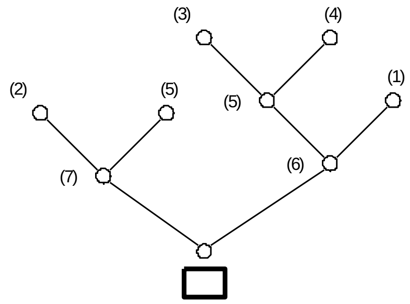

Figura 6.10 Dedução Indexada para o Exemplo 6.23

Exemplo 6.24: Considere o seguinte conjunto S de cláusulas:

(1) $_5P(y,a) \vee {}_1P(f(y),y)$
(2) $_6P(y,a) \vee {}_2P(y,f(y))$
(3) $_8{\sim}P(x,y) \vee {}_3P(f(y),y)$
(4) $_9{\sim}P(x,y) \vee {}_4P(y,f(y))$
(5) $_{10}{\sim}P(x,y) \vee {}_7{\sim}P(y,a)$

A partir de S, pode-se obter a seguinte resolução indexada de □:

(6)	$_5P(a,a) \vee {}_{10}{\sim}P(x,f(a))$	um resolvente indexado de (1) e (5)
(7)	$_8{\sim}P(x,a) \vee {}_{10}{\sim}P(y,f(a))$	um resolvente indexado de (3) e (5)
(8)	$_{10}{\sim}P(x,f(a)) \vee {}_{10}{\sim}P(y,f(a))$	um resolvente indexado de (6) e (7)
(9)	$_6P(a,a)$	um resolvente indexado de (8) e (2)
(10)	$_9{\sim}P(x,a)$	um resolvente indexado de (8) e (4)
(11)	□	um resolvente indexado de (9) e (10).

6.9 COMPLETUDE DA RESOLUÇÃO INDEXADA

Nesta seção, vai-se provar a completude do processo de resolução indexada. De modo usual, isto é feito, primeiro provando a completude do processo de resolução indexada para o caso base, que então é "elevado" ao caso mais geral. Para uma cláusula C que tem todos os literais indexados, toda instância de C é obtida, inicialmente, fazendo as substituições para as variáveis e fazendo os agrupamentos inferiores para literais idênticos. Isto é, se σ é uma substituição, primeiro obtém-se $C\sigma$ e então faz-se o agrupamento inferior para literais idênticos em $C\sigma$ para obter uma instância de C. Por exemplo, sendo $C = {}_1P(x) \vee {}_2P(y)$. Se $\sigma = \{a/x\ ,\ a/y\}$, então uma instância de C é $_1P(a)$. Por esta convenção, o seguinte lema de "elevação" (*lifting lemma*) é óbvio. Esta prova ficará como exercício. (Use o Lema 5.1)

Lema 6.2: Sendo C_1 e C_2 duas cláusulas com todos seus literais indexados. Se C_1' e C_2' são isntâncias de C_1 e C_2 , respectivamente, e se C' é um resolvente indexado de C_1' e C_2' , então existe um resolvente indexado C de C_1 e C_2 tal que C' é uma instância de C.

Agora vai-se provar a completude da resolução indexada para o caso base.

Lema 6.3: Sendo S um conjunto de cláusulas base, onde todos os literais em S estão indexados com um inteiro. Se S é insatisfatível, então existe uma dedução indexada da cláusula vazia □ a partir de S.

Prova: Seja k(S) definido como o número de ocorrências de literais em S menos o número de cláusulas em S. k(S) é chamado de *parâmetro de excesso de literal*. Prova-se o Lema 6.3 por indução sobre k(S). Se □ está em S, o Lema 6.3 é óbvio. Assumindo que □ não está em S. Se k(S) = 0, então S consiste somente de cláusulas unitárias. Uma vez que S é insatisfatível, existe um literal L tal que $_iL$ e $_j{\sim}L$ estão em S, onde i e j são os índices de $_iL$ e $_j{\sim}L$, respectivamente. Certamente, □ é um resolvente indexado de $_iL$ e $_j{\sim}L$. Assim, o Lema 6.3 é válido para k(S) = 0. Assumindo que o Lema 6.3 permanece válido quando k(S) < n. Para completar a indução, considera-se o caso onde k(S) = n e n > 0. Uma vez que k(S) > 0, existe pelo menos uma cláusula não unitária em S. Sendo r o índice mais alto entre todas as cláusulas não unitárias de S e $C = C' \vee {_rL}$ a cláusula não unitária que tem um literal $_rL$ indexada com r, onde C' é uma cláusula não vazia. Seja $S_1 = (S - \{C\}) \cup \{C'\}$ e $S_2 = (S - \{C\}) \cup \{_rL\}$. Certamente, S_1 e S_2 são insatisfatíveis. Entretanto, $k(S_1) < n$ e $k(S_2) < n$. Portanto, pela hipótese de indução, existem resoluções indexadas D_1' e D_2' de □ a partir de S_1 e S_2 , respectivamente. Sendo D_1 a dedução obtida a partir de D_1' colocando $_rL$ e C' juntos novamente. De modo claro, D_1 também é uma resolução indexada uma vez que r é o índice mais alto em uma cláusula relacionada a qualquer nó não inicial de D_1. D_1 é uma dedução indexada de □ ou $_rL$ a partir de S. Se for do primeiro (□) a prova está concluída. Se for do último, combinando D_1 e D_2', obtém-se uma dedução indexada de □ a partir de S. Isto completa a prova do Lema 6.3.

Teorema 6.3 (Completude da Resolução Indexada): Sendo S um conjunto de cláusulas, onde cada literal em S está indexado com um número inteiro. Se S é insatisfatível, então existe uma resolução indexada da cláusula vazia □ a partir de S.
Prova: Uma vez que S é insatisfatível, pelo Teorema de Herbrand (versão II) existe um conjunto finito insatisfatível S' de instâncias base das cláusulas em S. Pelo Lema 6.3, existe uma dedução indexada D' de □ a partir de S'. Usando o Lema 6.2 e um processo similar ao visto na prova do Teorema 6.1, pode-se transformar D' para produzir uma dedução indexada de □ a partir de S. Isto completa a prova.

Apesar da resolução indexada ser completa, ela não é compatível com a maioria das estratégias de resolução. Por exemplo, a combinação de resolução indexada com a estratégia de eliminação não é completa, bem como resolução indexada não é compatível com a estratégia de conjunto de apoio. (Observe os exercícios 20 e 21 deste capítulo). Resolução indexada é um refinamento muito restritivo de resolução, mas ainda assim uma regra de inferência muito eficiente.

6.10 EXERCÍCIOS

Seção 6.3

1. Determine se os seguintes conjuntos de cláusulas, interpretações e ordenações de átomos, constituem PI-conflitos. Se sim, encontre os PI-resolventes, caso contrário, dê as razões pelas quais não se configuram PI-coonflitos.

(a) $E_1 = P$, $E_2 = Q$, $E_3 = R$, $N = {\sim}P \vee {\sim}Q \vee {\sim}R \vee {\sim}W$
$I = \{{\sim}P, {\sim}Q, {\sim}R, W\}$
$P < Q < R < W$

(b) $E_1 = A_1 \vee A_3$, $E_2 = A_2 \vee A_3$, $N = {\sim}A_1 \vee {\sim}A_2 \vee A_3$
$I = \{{\sim}A_1, {\sim}A_2, {\sim}A_3\}$
$A_1 > A_2 > A_3$

(c) $E_1 = A_1 \vee A_3$, $E_2 = A_2 \vee A_3$, $N = \sim A_1 \vee \sim A_2 \vee A_3$

$I = \{\sim A_1, \sim A_2, \sim A_3\}$

$A_3 > A_1 > A_2$

(d) $E_1 = P(a)$, $E_2 = Q(a,b) \vee S(b,c)$, $N = \sim P(x) \vee \sim Q(x,y) \vee R(x)$

$I = \{\sim P(a), P(b), \sim Q(a,b), \sim S(b,c), \sim R(a)\}$

$P > Q > R > S$

2. Prove o seguinte teorema: Para um caso base, o núcleo N de um PI-conflito é verdadeiro em I, e para o caso geral (não base), existe pelo menos uma instância N' de N tal que N' é verdadeiro em I.

3. Sendo $S = \{P, Q \vee \sim P, R \vee \sim P, \sim P \vee \sim Q \vee \sim R\}$. Prove que S é insatisfatível por resolução semântica. Os seguintes casos devem ser considerados:

(a) $I = \{\sim P, \sim Q, \sim R\}$; $P > Q > R$

(b) $I = \{P, Q, R\}$; $Q > P > R$

(c) $I = \{\sim P, \sim Q, R\}$; $R > Q > P$

Seção 6.4

4. Prove o Teorema 6.1 usando a técnica de árvore semântica, isto é, a técnica dada na prova do Teorema 5.3. (Para maiores detalhes, e se for possível, consulte KOWALSWKI, R. e HAYES, P.J. *Semantic trees in automatic theorem-proving*, in Machine Intelligence, Vol. 4, (B. Meltzer e D. Michie, eds.), American Elsevier, New York, pp. 87-101, 1969).

5. Prove que a estratégia de eliminação é compatível com a PI-resolução, isto é, que a combinação da estratégia de eliminação com a PI-resolução é completa.

Seção 6.5

6. Sendo $S = \{P \vee Q, \sim P \vee Q, P \vee \sim Q, \sim P \vee \sim Q\}$ e $P > Q$. Prove que S é insatisfatível através:

(a) de hiperresolução positiva

(b) de hiperresolução negativa

7. Considere o exemplo 5.16. Nesse exemplo, tem-se o seguinte conjunto de cláusulas S:

$$S = \{\sim T(x,y,u,v) \vee P(x,y,u,v), \sim P(x,y,u,v) \vee E(x,y,u,v,y), T(a,b,c,d) \vee \sim T(a,b,c,d,b)\}$$

Prove que S é insatisfatível por: (a) de hiperresolução positiva

(b) de hiperresolução negativa

Traduza cada passo da prova em Português. Que dedução corresponde ao raciocínio *forward* (ou *backward*)?

8. Sendo S, o seguinte conjunto de cláusulas:

(1) $M(a,s(c),s(b))$

(2) $P(a)$

(3) $M(x,x,s(x))$

(4) $\sim M(x,y,z) \vee M(y,x,z)$

(5) $\sim M(x,y,z) \vee D(x,z)$

(6) $\sim P(x) \vee \sim M(y,z,u) \vee \sim D(x,u) \vee D(x,y) \vee D(x,z)$

(7) $\sim D(a,b)$

Considerando como conjunto de apoio a cláusula (7). Prove a insatisfiabilidade de S através da estratégia de conjunto de apoio.

9. Reconsidere o Exemplo 5.21, onde se tinha o seguinte conjunto de cláusulas:

(1) $P(a)$

(2) $\sim D(y) \vee L(a,y)$

(3) $\sim P(x) \vee \sim Q(y) \vee \sim L(x,y)$

(4) $D(b)$

(5) $Q(b)$

Considerando como conjunto de apoio as cláusulas (4) e (5). Prove a insatisfiabilidade do conjunto de cláusulas acima através da estratégia de conjunto de apoio. Traduza a prova completa para o Português.

Seção 6.6
10. Considere o seguinte conjunto de cláusulas ordenadas:

$$S = \{P \vee Q, \sim P \vee Q, P \vee \sim Q, \sim P \vee \sim Q\}$$

Sendo I = {P, Q}. Prove a insatisfiabilidade de S através de OI-resolução.
11. No Exemplo 6.18, mostrou-se um contra-exemplo para mostrar que OI-resolução não é completa. Você é capaz de mostrar outro contra-exemplo?

Seção 6.7
12. Sendo $S = \{\sim P \vee \sim Q \vee \sim R, P, Q, R\}$:

(a) use o algoritmo mencionado na seção 6.7 para provar que S é insatisfatível.
(b) use, novamente, o mesmo algoritmo sem o conceito de cláusulas ordenadas, isto é, no Passo 6, o literal resolvido de C_2 não está restrito a ser o último literal de C_2 .
(c) compare o número de cláusulas geradas em cada caso.

13. Use o algoritmo da Seção 6.7 para provar os teoremas dados em: (a) Exemplo 5.21
(b) Exemplo 5.22.
14. Mostre que se um conjunto ordenado de cláusulas S é insatisfatível, então $\square$ pode ser gerado pelo algoritmo da Seção 6.7.

Seção 6.8
15. Sendo $S = \{P, Q, R, W, \sim P \vee \sim Q \vee \sim R \vee \sim W\}$.

(a) se resolução ordinária é usada, quantos resolventes serão gerados a partir de S pelo método de saturação antes de $\square$ ser gerada?
(b) atribua índices aos literais de S. Se resolução indexada é usada, quantos resolventes indexados serão gerados a partir de S pelo método de saturação antes de $\square$ ser gerada?
(c) compare o número de cláusulas geradas em cada caso.

16. Considere os conjuntos de cláusulas S dados no Exemplo 6.18, faça a indexação dos literais em S como segue:

(1) $_7P \vee {}_1Q$
(2) $_8Q \vee {}_2R$
(3) $_9R \vee {}_3W$
(4) $\sim_{10}R \vee \sim_4P$
(5) $\sim_{11}W \vee \sim_5Q$
(6) $\sim_{12}Q \vee \sim_6R$

Deduza a cláusula vazia $\square$ a partir de S por resolução indexada.
17. Considere o seguinte conjunto de cláusulas (este conjunto foi tomado do Exemplo 5.22):

(1) $_1{\sim}E(x) \vee {}_2V(x) \vee {}_3S(x,f(x))$
(2) $_4{\sim}E(x) \vee {}_5V(x) \vee {}_6C(f(x))$
(3) $_7P(a)$
(4) $_8E(a)$
(5) $\sim_9S(a,y) \vee {}_{10}P(y)$
(6) $\sim_{11}P(x) \vee \sim_{12}V(x)$
(7) $\sim_{13}P(x) \vee \sim_{14}C(x)$

Deduza a cláusula vazia $\square$ a partir do conjunto de cláusulas acima por resolução indexada.

Seção 6.9

18. Prove o Lema 6.2.

19. Para qualquer conjunto arbitrário S de cláusulas:

    (a) encontre um método para indexar todo literal em S tal que o conjunto de todos os resolventes indexados de S seja igual ao conjunto de todos os resolventes ordinários (sem refinamentos) de S, isto é, simule resolução através de resolução indexada.

    (b) simule hiperresolução positiva através de resolução indexada.

20. Encontre um contra-exemplo para mostrar que a combinação de estratégia de eliminação e resolução indexada não é completa.

20. Encontre um contra-exemplo para mostrar que a combinação de estratégia de conjunto de apoio e resolução indexada não é completa.

CAPÍTULO 7 - RESOLUÇÃO LINEAR

7.1 INTRODUÇÃO

Quando se está provando uma indentidade, geralmente, se inicia pelo lado esquerdo da identidade, aplica-se uma regra de inferência para obter alguma outra expressão, então aplica-se alguma outra regra de inferência sobre a expressão recém obtida e repete-se esta operação até que o lado esquerdo da expressão é idêntico ao lado direito da expressão da identidade. A idéia de resolução linear é semelhante a este tipo de raciocínio em cadeia. Ele inicia com uma cláusula, resolve esta cláusula contra outra para obter um resolvente, então resolve este resolvente contra alguma cláusula até que a cláusula vazia □ seja encontrada.

A característica principal de dedução linear é sua estrutura simples. Além disso, resolução linear é completa e compatível com a estratégia de conjunto de apoio. Em adição, será mostrado que métodos heurísticos podem ser convenientemente aplicados em conjunto com ela. Porém, entre todas as suas vantagens, a que nos parece ser mais interessante é a sua facilidade de ser implementada em um computador.

7.2 RESOLUÇÃO LINEAR

Definição: Dado um conjunto de cláusulas S e uma cláusula C_0 em S, uma *dedução linear* de C_n a partir de S considerando como *cláusula de partida* C_0, é uma dedução da forma mostrada na figura 7.1, onde:

1. para i = 0, 1, ... , n-1, C_{i+1} é um resolvente de C_i (chamada *cláusula central*) e B_i (chamada de *cláusula lateral*), e
2. cada B_i está em S, ou é um C_j para algum j, j < i.

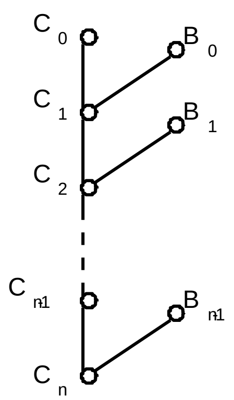

Figura 7.1 Dedução linear de C_n a partir de S considerando como cláusula de partida C_0

Exemplo 7.1: Sendo S = {P ∨ Q, ~P ∨ Q, P ∨ ~Q, ~P ∨ ~Q}. Então a figura 7.2 é a dedução linear de □ a partir de S, tendo como cláusula de partida P ∨ Q. Existem quatro cláusulas laterais. Três delas estão em S, enquanto que, uma delas, Q (sublinhada), é um resolvente gerado antes de □ ser deduzido.

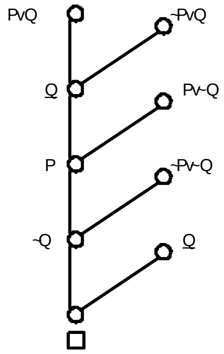

Figura 7.2 Dedução linear de □ a partir de S, tendo como cláusula de partida P ∨ Q.

Exemplo 7.2: Considere o conjunto de cláusulas S = {M(a,s(c),s(b)), P(a), M(x,x,s(x)), ~M(x,y,z) ∨ M(y,x,z), ~M(x,y,z) ∨ D(x,z), ~P(x) ∨ ~M(y,z,u) ∨ ~D(x,u) ∨ D(x,y) ∨ D(x,z), D(a,b)}. Então a figura 7.3 é a dedução linear de □ a partir de S.

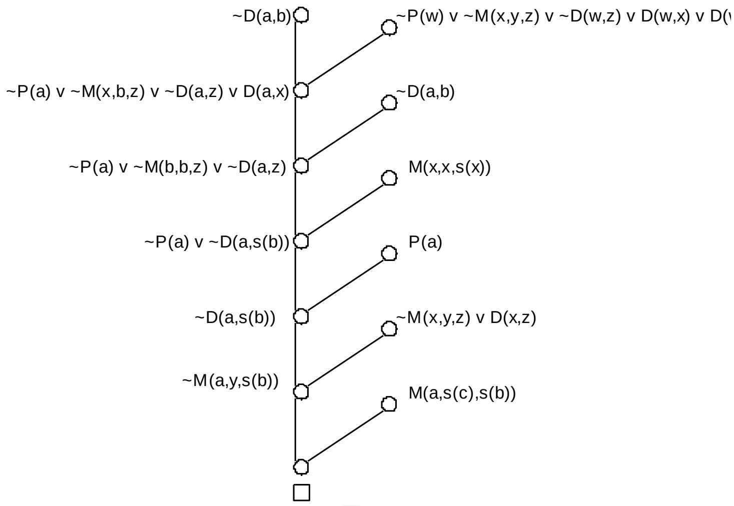

Figura 7.3 Dedução linear de □ a partir de S para o Exemplo 7.2.

A dedução linear introduzida nesta seção é uma forma primitiva e será modificada gerando a OL-*dedução* discutida na seção 7.4. A prova de completude da resolução linear é dada na seção 7.5.

7.3 RESOLUÇÃO POR ENTRADA E RESOLUÇÃO UNITÁRIA

Quando se considera um refinamento de resolução, se deseja que este seja completo, isto é, que garanta que a cláusula vazia será sempre encontrada a partir de um conjunto insatisfatível de cláusulas. Entretanto, a eficiência do processo também é importante na prova automática de teoremas. Algumas vezes, pode-se necessitar trocar a completude pela eficiência. Isto é, podem existir alguns refinamentos para resolução que são eficientes, mas incompletos. Se um refinamento da resolução é eficiente e poderoso o bastante para provar uma grande classe de teoremas, mesmo sendo incompleto, ele ainda poderá ser útil. Nesta seção, vai-se considerar aluns refinamentos para resolução que apesar de incompletos se mostraram bastante eficientes, são eles: resolução por entrada e resolução unitária. Elas são muito mais fáceis de implementar e, apesar de não completas, são muito mais eficientes que resolução linear. Vai-se mostrar que resolução por entrada é equivalente a resolução unitária, isto é, os teoremas que podem ser provados por resolução por entrada também poderão ser provados por resolução unitária e vice-versa.

Dado um conjunto de cláusulas S, uma vez que S é o conjunto de entrada original, pode-se chamar cada membro de S de *cláusula de entrada*.

Definição: Uma *resolução por entrada* é uma resolução na qual uma das duas cláusulas pai é uma cláusula de entrada. Uma *dedução por entrada* (para enfatizar o conjunto de entrada S, e que algumas vezes pode ser chamada de *dedução S-entrada*) é uma dedução na qual toda resolução é uma resolução por entrada. Uma *refutação por entrada* é uma resolução por entrada de □ a partir de S.

Uma dedução por entrada é, na verdade, uma dedução linear onde toda cláusula lateral é uma cláusula de entrada e, portanto, é uma subcaso da dedução linear.

Exemplo 7.3: Considere o seguinte conjunto de cláusulas:

~P(x,y,u) ∨ ~P(y,z,v) ~P(x) ∨ ~P(x,v,w) ∨ P(u,z,w)
P(g(x,y),x,y)
P(x,h(x,y),y)
~P(k(x),x,k(x))

A figura 7.4 mostra uma refutação por entrada deste conjunto.

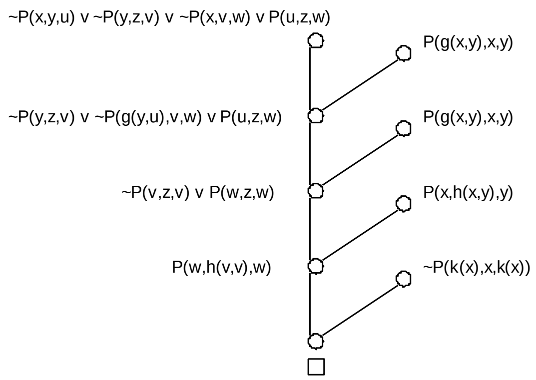

Figura 7.4 Refutação por entrada para o conjunto de cláusulas do Exemplo 7.3

Definição: Uma *resolução unitária* é uma resolução na qual um resolvente é obtido usando-se pelo menos uma cláusula pai unitária, ou um fator unitário de uma cláusula pai. Uma *dedução unitária* é uma dedução na qual toda resolução é uma resolução unitária. Uma *refutação unitária* é uma dedução unitária de □.

Resolução unitária é, essencialmente, uma extensão da regra do literal-único de Davis e Putnam (veja seção 4.6) para a lógica de predicados. Esta regra é especialmente importante, pois para deduzir □ a partir de um conjunto de cláusulas, deve-se obter, sucessivamente, cláusulas menores, e resolução unitária fornece um meio de progressão rápida em direção à cláusulas menores. Prova-se agora a equivalência entre resolução por entrada e resolução unitária.

Lema 7.1: Existe uma refutação unitária a partir de um conjunto de cláusulas base S se e somente se existe uma refutação por entrada a partir de S.
<u>Prova</u>: O Lema 7.1 é provado por indução. Sendo A o conjunto de átomos de S. Se A consiste de um único elemento, dito Q, então, entre os elementos de S, existem as cláusulas Q e ~Q. Claramente, o resolvente de Q e ~Q é a cláusula vazia □. Esta dedução é, ao mesmo tempo, uma refutação por entrada e unitária. Portanto, o Lema 7.1 é válido para este caso. Assuma que o Lema 7.1 é válido quando A consiste de i elementos, 1 1 i 1 n. Para completar a indução, considera-se A tal que A consiste de, exatamente, n + 1 elementos.
(IDA) Se existe uma refutação unitária a partir de S, então S deve conter no mínimo uma cláusula unitária, dita L, onde L é um literal. Sendo S' o conjunto obtido a partir de S eliminando-se as cláusulas que contém o literal L e eliminando ~L das cláusulas restantes. Certamente, uma vez que existe uma refutação unitária a partir de S, deve existir uma refutação unitária a partir de S' (a prova disto será deixada como exercício). Porém S' contém n ou menos átomos. Pela hipótese da indução, existe uma refutação S'-entrada D' de S'. Sendo D a dedução obtida a partir de D' acrescentando, novamente, o literal ~L de volta às cláusulas de onde ele foi eliminado. Sendo T a cláusula associada ao nó raiz de D, D é uma dedução S-por entrada de T a partir de S. T deve ser □ ou ~L. Se T é □, fica-se satisfeito. Se T é ~L, pode-se obter □ resolvendo T com L, que também é uma cláusula de entrada. Assim, a refutação obtida a partir de D e a resolução de T e L é uma refutação por entrada a partir de S. Portanto completou-se a prova da primeira parte do Lema 7.1.
(VOLTA) De modo inverso, se existe uma refutação por entrada a partir de S, então S deve conter, pelo menos, uma cláusula unitária, denominada L, onde L é um literal. Sendo:

S' = { S 2 [Res(C,L) | C 2 S]} - {todas as cláusulas que contenham L ou ~L}

onde Res(C,L) representa o resolvente de uma cláusula C e uma cláusula L. Uma vez que existe uma refutação S-entrada a partir de S, deve existir uma refutação S'-entrada a partir de S'. Porém S' contém n ou menos átomos. Pela hipótese da indução, existe uma refutação unitária a partir de S'. Entretanto, toda a cláusula em S' é membro de S ou é um resolvente obtido pela aplicação de resolução unitária na cláusula L e uma cláusula em S. Portanto, existe uma refutação unitária a partir de S. Isto completa a prova da segunda parte do Lema 7.1.

Teorema 7.1 (Equivalência das Resoluções por Entrada e Unitária): Existe uma refutação unitária a partir de um conjunto de cláusulas S se e somente se existe uma refutação por entrada a partir de S.
<u>Prova</u>: (IDA) Se existe uma refutação unitária D_1 a partir de S, então a partir de D_1 pode-se obter uma refutação unitária base D_1' substituindo-se a cláusula C associada a cada nó em D_1 pela instância base apropriada de C. Sendo S' o conjunto de cláusulas base associado aos nós iniciais de D_1', pelo Lema 7.1, existe uma refutação por entrada D_2' a partir de S'. A partir de D_2' usando o Lema 5.1 (*Lifting Lemma*) pode-se obter a refutação por entrada D_2 a partir de S.
(VOLTA) A prova da segunda parte do Teorema 7.1 é idêntica a mostrada acima exceto que as palavras "unitária" e "por entrada" são trocadas.

Dado um conjunto de cláusulas S, se S tem uma refutação por entrada, então, pelo Teorema 7.1, pode-se também derivar □ a partir de S aplicando resolução unitária. Uma vez que resolução unitária é mais fácil de implementar que resolução por entrada, normalmente, opta-se por implementar a resolução unitária que se mostra bastante eficiente.

7.4 RESOLUÇÃO LINEAR USANDO CLÁUSULAS ORDENADAS E INFORMAÇÃO DOS LITERAIS RESOLVIDOS

Pode-se limitar a resolução linear introduzindo dois conceitos a ela. Um deles é o conceito de cláusulas ordenadas. Como visto no Capítulo 6 a incorporação deste conceito a resolução semântica aumenta bastante a sua eficiência, observar-se-á que isto também é verdade para resolução linear. Além disso, de maneira contrária a resolução semântica, a incorporação do conceito de cláusulas ordenadas na resolução linear não destruirá sua completude. Outro conceito é aquele que usa a informação dos literais resolvidos. Na resolução, quando um resolvente é obtido, os literais resolvidos são eliminados. Na verdade, a informação fornecida por estes literais é bastante útil e poderá ser usada para melhorar a resolução linear.

Antes de introduzir o mecanismo de utilização de informação sobre os literais resolvidos, vai-se primeiro examinar um caso mais simples. Considere o conjunto de cláusulas definido como $S = \{P \vee Q, P \vee \sim Q, \sim P \vee Q, \sim P \vee \sim Q\}$, a figura 7.5 mostra uma refutação linear a partir de S.

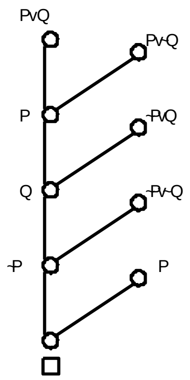

Figura 7.5 Refutação linear para $S = \{P \vee Q, P \vee \sim Q, \sim P \vee Q, \sim P \vee \sim Q\}$

Note que uma das cláusulas laterais (a cláusula P) não é uma cláusula de entrada. De fato, é fácil ver que não existe uma prova unitária para este conjunto de cláusulas. Assim, de acordo com o Teorema 7.1, também não existe uma prova por entrada. Isto é, o uso de uma cláusula central como uma cláusula lateral é inevitável.

Seria interessante se fosse possível encontrar uma condição necessária e suficiente, para a qual uma cláusula lateral deve ser uma cláusula central previamente gerada. Isto é, uma cláusula lateral é uma cláusula central previamente gerada quando e somente quando a condição é satisfeita.

Vai-se mostrar que se a informação dos literais resolvidos é apropriadamente armazenada e o conceito de cláusulas ordenadas é usado, está condição pode ser definida. Ficará óbvio, mais adiante, que o uso desta condição irá diminuir o número de resoluções possíveis. Na verdade, isto irá permitir ainda mais, se for determinado que uma cláusula lateral deve ser uma cláusula central previamente gerada, não será necessário nem determinar qual cláusula central utilizar. Portanto, não há a necessidade de se armazenar as cláusulas centrais previamente geradas. Isto simplifica enormemente a implementação computacional de resolução linear.

Vai-se discutir agora o mecanismo de armazenar a informação dos literais resolvidos. Suponha que C_1 e C_2 são duas cláusulas ordenadas como segue: $C_1 = P \vee Q$

$C_2 = \sim Q \vee R$.

Existe um resolvente ordenada (ou seja, $P \vee R$) de C_1 contra C_2, com Q e ~Q sendo os literais resolvidos. Uma vez que Q e ~Q são complementares um em relação ao outro, necessita-se somente armazenar um deles. Suponha que se convenciona armazenar Q, o último literal de

C_1. Então, o resolvente ordenado pode ser representado por $\boxed{Q}$, onde o literal emoldurado é o literal resolvido. Esta é a maneira de como se armazena informação dos literais resolvidos. Isto é, os literais resolvidos são indicados através de literais emoldurados em uma cláusula ordenada. Os literais emoldurados servem apenas para armazenar aqueles literais que foram resolvidos, ou seja, eles não participam da resolução.

No esquema acima, se um literal emoldurado não é seguido por nenhum literal não emoldurado, deve-se eliminar este literal emoldurado. O porquê de se fazer isto será explicado mais adiante.

O algoritmo que aplica os conceitos de cláusulas ordenadas e informação dos literais resolvidos é chamado de OL-dedução (dedução linear ordenada). Antes de apresentar o algoritmo completo, vai-se usar um exemplo para descrevê-lo.

Exemplo 7.4: Considere o conjunto ordenado de cláusulas definido como: S = {P ∨ Q, P ∨ ~Q, ~P ∨ Q, ~P ∨ ~Q}. Uma refutação linear usando informação de literais resolvidos e o conceito de cláusulas ordenadas é mostrado na figura 7.6, onde as cláusulas centrais são obtidas como segue:

1. Inicia-se com a cláusula ordenada P ∨ Q como cláusula de partida. O último literal desta cláusula ordenada é Q, que pode ser resolvido com P ∨ ~Q sobre ~Q. Portanto, resolver P ∨ Q contra P ∨ ~Q e armazenando o literal Q, obtém-se o resolvente que é representado por $P \vee \boxed{Q}$. Uma vez que $\boxed{Q}$ não é seguido por nenhum literal emoldurado, ele é eliminado. Portanto, obtém-se P, como mostrado na figura 7.6.
2. O último literal da cláusula ordenada obtida em (1) é P, que pode ser resolvido contra ~P ∨ Q sobre o literal ~P. Portanto, resolvendo P contra ~P ∨ Q e emoldurando o literal P, obtem-se $\boxed{P} \vee Q$.
3. O último literal de $\boxed{P} \vee Q$ é Q, que pode ser resolvido com ~P ∨ ~Q. Assim, resolvendo $\boxed{P} \vee Q$ contra ~P ∨ ~Q e armazenando Q, obtem-se $\boxed{P} \vee \boxed{Q} \vee {\sim}P$.
4. Agora obteve-se a cláusula $\boxed{P} \vee \boxed{Q} \vee {\sim}P$, que apresenta uma importante característica. Observe que o último literal de $\boxed{P} \vee \boxed{Q} \vee {\sim}P$ é ~P, que é complementar a um dos literais emoldurados, ou seja $\boxed{P}$. Esta é a condição que se estava tentando encontrar antes, em outras palavras, a cláusula lateral neste passo deve ser uma cláusula central, ou seja, a cláusla P. Resolvendo $\boxed{P} \vee \boxed{Q} \vee {\sim}P$ com P sobre o literal ~P, obtem-se $\boxed{P} \vee \boxed{Q} \vee \boxed{{\sim}P}$. Entretanto, nenhum destes literais emoldurados é seguido por literais não emoldurados, portanto, eles são eliminados obtendo-se a cláusula vazia □.

   Como enfatizado antes, a cláusula $\boxed{P} \vee \boxed{Q} \vee {\sim}P$ é especial, pois seu último literal é complementar a um dos literais emoldurados da cláusula. Toda vez que isto acontecer, deve-se usar uma cláusula central, anteriormente gerada, como cláusula lateral. Denomina-se este tipo de cláusula de cláusula redutível.

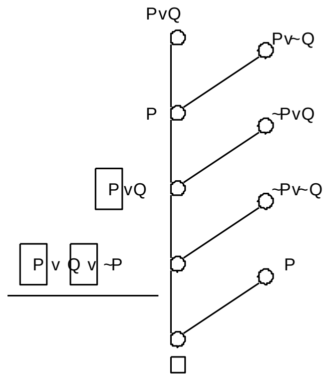

Figura 7.6 Refutação linear para S = {P ∨ Q, P ∨ ~Q, ~P ∨ Q, ~P ∨ ~Q} usando informação dos literais resolvidos e o conceito de cláusulas ordenadas.

Definição: Uma cláusula ordenada C é uma *cláusula ordenada redutível* se e somente se o último literal de C é unificável com a negação de um literal emoldurado de C.

Na verdade, sempre que uma cláusula ordenada reduzida é gerada, não é necessário fazer a busca na memória das cláusulas centrais deduzidas para resolvê-la. Ao invés disto, simplesmente elimina-se o último literal desta cláusula ordenada. Por exemplo, quando □Q□ ∨ é gerada, simplesmente elimina-se ~P de □Q□ ∨ e obtem-se □∨□. Uma vez que $\boxed{P}$ e $\boxed{Q}$ não são seguidos por nenhum literal não emoldurado, eles são eliminados e obtem-se a cláusula vazia □. Denomina-se esta operação de *redução* de uma cláusula ordenada redutível.

Definição: Sendo C uma cláusula ordenada redutível e o último literal L unificável com algum literal emoldurado tendo como unificador mais geral σ. A *cláusula ordenada reduzida de C* é a cláusula ordenada obtida de Cσ eliminando Lσ e todo os literais emoldurados subsequentes que não são seguidos por nenhum literal não emoldurado.

Uma vez que a redução de cláusulas redutíveis é incorporada à OL-dedução, não será mais necessário armazenar cláusulas intermediárias. Este importante aspecto da OL-dedução torna-a bastante propícia a implementação computacional. Note que a eliminação de uma cláusula ordenada redutível, por si só, já diminui o número de possibilidades de resoluções. A introdução deste mecanismo de redução efetivamente reduz este número a apenas uma resolução.

Agora va-se definir formalmente a OL-dedução. Na seqüência, uma cláusula ordenada pode conter tanto literais emoldurados quanto não emoldurados. Para levar isto em consideração, repete-se as definições de um fator ordenado, um resolvente binário ordenado e um resolvente ordenado, dados na seção 6.6.

Numa cláusula ordenada, se existe mais de uma ocorrência do mesmo literal não emoldurado, sempre se mantém somente o literal mais à esquerda na cláusula, eliminando os outros literais idênticaos não emoldurados. Esta operação é chamada de *agrupamento à esquerda* de literais idênticos não emoldurados. Por exemplo, agrupando à esquerda os literais idênticos não emoldurados em $R \vee S$, obtem-se $Q \vee R$.

Definição: Se dois ou mais literais não emoldurados (com o mesmo sinal) de uma cláusula ordenada C, tem um unificador mais geral σ, então a cláusula ordenada é obtida da seqüência $C\sigma$, pelo agrupamento à esquerda de qualquer literal idêntico em $C\sigma$ e pela eliminação de todo literal emoldurado que não é seguido por nenhum literal não emoldurado na cláusula restante e é chamada de *fator ordenado* de C.

Definição: Sendo C_1 e C_2 duas cláusulas ordenadas, sem variáveis em comum e L_1 e L_2 dois literais não emoldurados em C_1 e C_2 , respectivamente. Se L_1 e $\sim L_2$ tem um unificador mais geral σ e se C^* é a cláusula ordenada obtida concatenando-se as seqüências $C_1\sigma$ e $C_2\sigma$, emoldurando $L_1\sigma$, removendo $L_2\sigma$ e agrupando à esquerda todo literal idêntico na seqüência restante. Sendo C obtido a partir de C^* pela remoção de todo literal emoldurado não seguido por literais não emoldurados em C^* , então C é chamado de *resolvente binário ordenado* de C_1 contra C_2. Os literais L_1 e L_2 são chamados de literais resolvidos.

Definição: Um *resolvente ordenado* de uma cláusula ordenada C_1 contra uma cláusula ordenada C_2 é um dos seguintes resolventes binários ordenados:

1. um resolvente binário ordenado de C_1 contra C_2,
2. um resolvente binário ordenado de C_1 contra um fator ordenado de C_2,
3. um resolvente binário ordenado de um fator ordenado de C_1 contra de C_2,
4. um resolvente binário ordenado de um fator ordenado de C_1 contra um fator ordenado de C_2 .

Definição: Dados um conjunto de cláusulas ordenadas S e uma cláusula ordenada C_0 em S, uma OL-dedução de C_n a partir de S tendo como cláusula ordenada inicial C_0 é uma dedução da forma mostrada na figura 7.7 que satisfaz as seguintes condições:

1. Para i = 0, 1, 2, ... , n-1, C_{i+1} é um resolvente ordenado de C_i (chamada *cláusula central ordenada*); o literal resolvido em C_i (ou uma fator ordenado de C_i) é o último literal.
2. Cada B_i é uma cláusula ordenada em S ou uma instância de algum C_j, j < i. B_i é uma instância de algum C_j, j < i, se e somente se C_i é uma cláusula ordenada redutível. Neste caso, C_{i+1} é a cláusula ordenada reduzida de C_i .
3. Não existem tautologias na dedução.

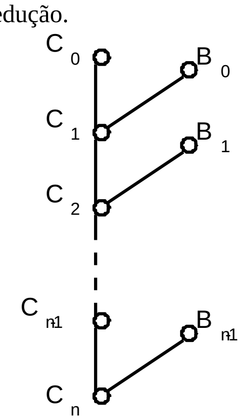

Figura 7.7 OL-dedução de C_n a partir de S considerando como cláusula de partida C_0

A definição de cláusula ordenada pode ser usada para provar o seguinte Lema, cuja prova será deixada como exercício.

Lema 7.2: Em uma OL-dedução, se C é uma cláusula ordenada redutível, então existe uma cláusula ordenada central C_j, $j < i$, tal que a cláusula ordenada reduzida C_{i+1} de C_i é um resolvente ordenado de C_i contra uma instância de C_j.

Deste modo as condições em uma OL-dedução podem ser fixadas como:

1. Todo B_i está em S ou é uma instância de algum C_j, $j < i$.
2. Se C_i é uma cláusula ordenada redutível, então C_{i+1} é a cláusula ordenada reduzida de C_i. Caso contrário, C_{i+1} é um resolvente ordenado de C_i com B_i em S onde o literal resolvido em C_i é o último literal.
3. Nenhuma tautologia está presente na dedução.

Definição: Uma OL-refutação é uma OL-dedução de □.

Exemplo 7.5: Considere o conjunto de cláusulas ordenadas S = {P ∨ Q, ~Q ∨ R, R ∨ ~P, ~Q ∨ ~R, ~P ∨ ~R}. Então a figura 7.8 é uma OL-dedução de □ a partir de S tendo como cláusula ordenada de partida P ∨ Q. Observe que C_2 e C_5 são cláusulas ordenadas redutíveis. Apesar da resolução não ser necessária nestes passos, as cláusulas laterais se encontram representadas através de linhas tracejadas. Por exemplo, B_2 é P ∨ Q, que é a cláusula de partida C_0, e B_5 é P, que é C_3 deduzido nesta refutação.

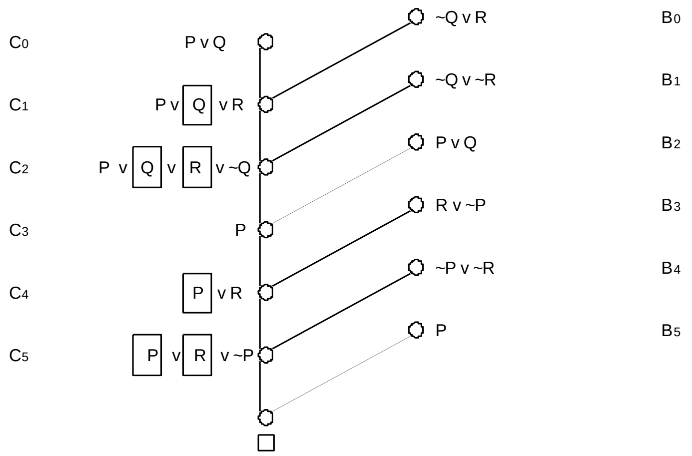

Figura 7.8 OL-dedução de □ a partir de S tendo como cláusula ordenada de partida P ∨ Q

Exemplo 7.6: No exemplo 7.2, foi apresentado um exemplo de dedução linear sem o conceito de cláusulas ordenadas e informação sobre os literais resolvidos. Na figura 7.9, prova-se o mesmo exemplo usando OL-dedução.

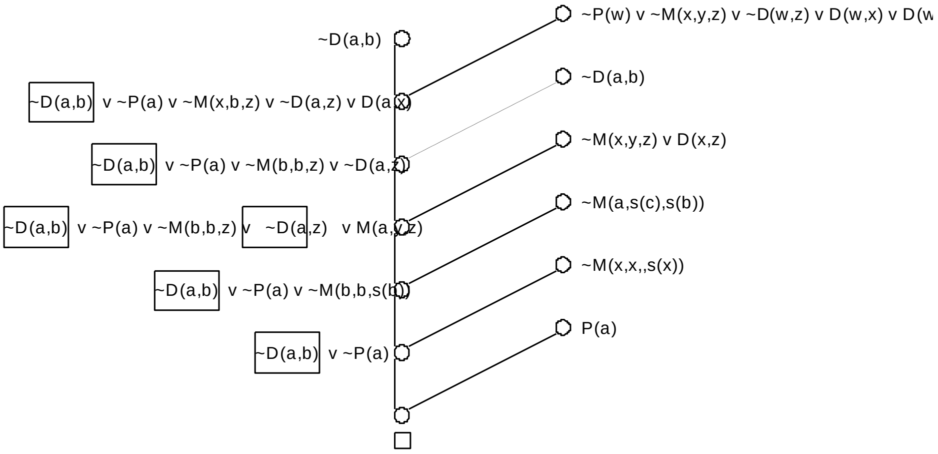

Figura 7.9 OL-dedução das cláusulas do Exemplo 7.2

7.5 COMPLETETUDE DA RESOLUÇÃO LINEAR

Nesta seção vai-se provar a completude da OL-dedução, para isto prova-se, inicialmente, o seguinte lema.

Lema 7.3: Se C é uma cláusula ordenada base em um conjunto insatisfatível de cláusulas ordenadas base S e se S - {C} é satisfatível, então existe uma OL-refutação a partir de S tendo como cláusula ordenada de partida C.

Prova: Prova-se este lema por indução sobre o número de elementos no conjunto de átomos de S. Seja A o conjunto de átomos de S. Se A consiste de um único elemento, denominado Q, então entre os elementos de S existem as cláusulas ordenadas Q e ~Q. Certamente, o resolvente de Q e ~Q é □. Uma vez que S - {C} é satisfatível, Q ou ~Q deve ser C. Portanto o Lema 7.3 é válido para este caso.

Assuma que o Lema 7.3 é válido qaundo A consiste de i elementos, $1 \leq i \leq n$. Para completar a indução, considera-se A tal que A consiste de exatamente n + 1 elementos.

CASO 1: C é uma cláusula unitária. Seja C $\triangleq$ L, onde L é um literal. Sendo S' o conjunto obtido a partir de S pela eliminação das cláusulas ordenadas contendo L e pela eliminação de ~L das cláusulas ordenadas restantes, S' deve ser instatisfatível. Seja T' um subconjunto insatisfatível de S' tal que todo subconjunto próprio de T' é satisfatível (esta subconjunto pode ser obtido considerando, exaustivamente, todos os subconjuntos possíveis de S'). T' deve conter alguma cláusula ordenada E' que foi obtida a partir de uma cláusula em S pela eliminação do literal ~L, pois caso contrário T' seria um subconjunto de S - {C} que, nesta caso, seria insatisfatível, violando o fato de que é satisfatível. Tem-se agora uma cláusula ordenada E' no conjunto insatisfatível de cláusulas T' e T' - {E} é satisfatível. Uma vez que T' contém n ou menos átomos, pela hipótese induzida, existe uma OL-refutação D' a partir de T' com cláusula de partida E' como mostrado na figura 7.10 (a). Adicionando novamente o literal ~L, no seu lugar correto, em todas as cláusulas ordenadas das quais ele foi eliminado, com exceção da cláusula ordenada de partida E' e obtendo E' pela resolução de L contra E, obtem-se a dedução D de □ ou (~L) como mostrado na figura 7.10 (b).

Na figura 7.10 (b), o símbolo ($\vee$ ~L) significa a adição do literal ~L no lugar de onde ele foi originalmente eliminado. Mostra-se agora que D também é uma OL-dedução.

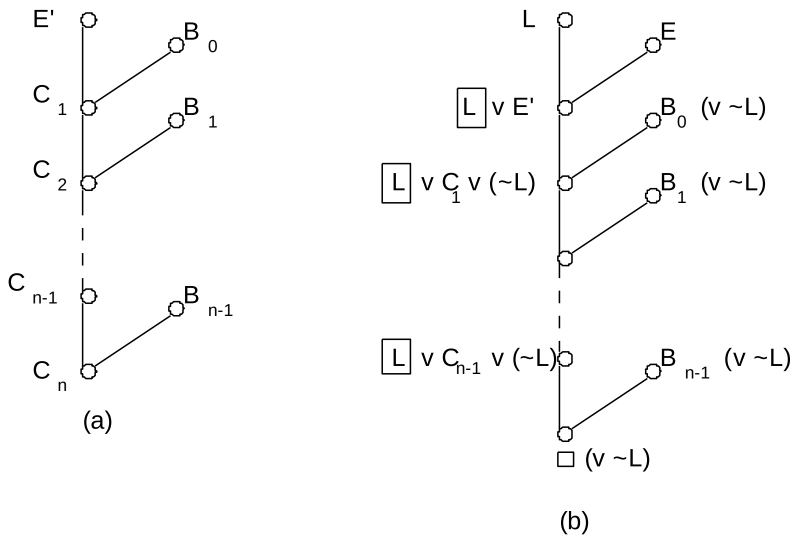

Figura 7.10 (a) OL-refutação D' a partir de T' com cláusula de partida E'
(b) Dedução D de □ ou (~L)

Note que nenhuma cláusula em D é uma tautologia, uma vez que D é obtido a partir de D', que não contém nem uma tautologia nem o literal L. Observe também que se ~L não aparece como último literal de uma cláusula central, então a resolução em D é exatamente igual a resolução em D'. Se ~L aparece como último literal de uma cláusula central, então ~L deve ser resolvido. Neste caso, substitui-se a parte de D mostrada na figura 7.11 (a) pela dedução mostrada na figura 7.11 (b). Fica óbvio que $\boxed{L} \vee C_i \vee$ é redutível e $\boxed{L} \vee C_i$ é obtido pela operação de redução.

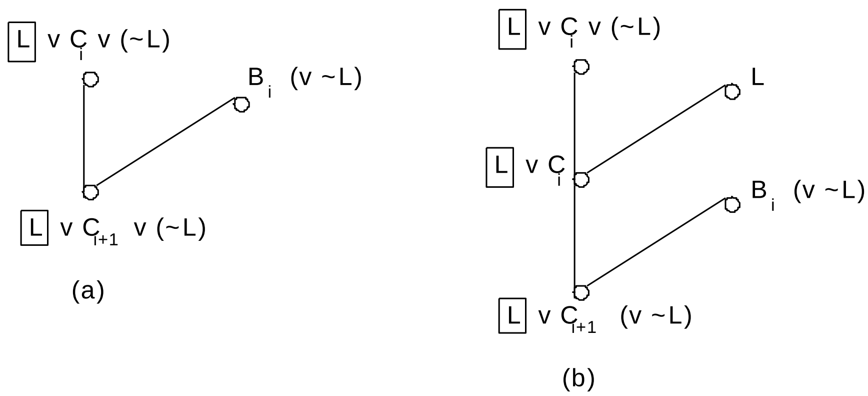

Figura 7.11 (a) Parte de D onde ~L aparece como último literal de uma cláusula central
(b) Substituição da parte de D onde ~L aparece como último literal de uma cláusula central, utilizando-se a operação de redução

Após fazer estas substituições sempre que requerido na figura 7.10 (b), obtem-se uma OL-dedução D* de □ ou $\boxed{L} \vee \sim L$ a partir de S com cláusula ordenada de partida C. Se D* é

uma dedução de $\square$, a prova está pronta. Caso contrário, D^*, juntamente com a resolução de $\boxed{L} \vee \sim L$ contra L, é uma OL-refutação a partir de S com cláusula ordenada de partida C.
CASO 2: C não é uma cláusula unitária. Neste caso, sendo L o primeiro (mais a esquerda) literal em C, isto é, $C \triangleq L \vee C'$, onde C' é uma cláusula ordenada não vazia e S' o conjunto obtido a partir de S pela eliminação das cláusulas ordenadas contendo ~L e eliminando L das cláusulas ordenadas restantes, então S' é insatisfatível. Deve-se mostrar que S' - {C'} é satisfatível. Seja I uma interpretação que satisfaz S - {C} (este I existe, pois S - {C} é satisfatível). Uma vez que S é insatisfatível, C deve ser falso em I, portanto L é falso em I. Consequentemente, S' - {C'} é verdadeiro em I. Deste modo S' - {C'} é satisfatível. Uma vez que S' contém n ou menos átomos, pela hipótese de indução existe uma OL-dedução D' de $\square$ a partir de S' com cláusula ordenada de partida C'. Adicionando o literal L, em seu lugar correto, nas cláusulas das quais ele foi eliminado, obtem-se uma OL-dedução D_1 de L a partir de S com cláusula ordenada de partida C. Agora, $\{L\} \cup (S - \{C\})$ é insatisfatível e S - {C} é satisfatível. Como provado na Caso 1, existe uma OL-dedução D_2 de $\square$ a partir de $\{L\} \cup (S - \{C\}$ com cláusula ordenada de partida {L}. Colocando D_1 no topo de D_2, obtem-se uma OL-refutação a partir de S com cláusula ordenada de partida C. Isto completa a prova.

Teorema 7.2 (Completude da OL-dedução): Se C é uma cláusula ordenada em um conjunto insatisfatível S de clásulas ordenadas e se S - {C} é satisfatível, então existe uma OL-refutação a partir de S com cláusula ordenada de partida C.
<u>Prova</u>: Uma vez que S é insatisfatível e S - {C} é satisfatível, pelo Teorema de Herbrand (versão II) existe um conjunto finito S' de instâncias base das cláusulas ordenadas de S e uma instância base C' de C tal que S' é insatisfatível, C' está em S' e S' - {C'} é satisfatível. Pelo Lema 7.3, existe uma OL-refutação D' a partir de S' com cláusula ordenada de partida C'. Usando o Lema 5.1 (Lifting Lemma), a partir de D', pode-se obter uma OL-refutação a partir de S com cláusula ordenada de partida C. Isto completa a prova.

Pode ser provado rapida e facilmente que a resolução por conjunto de apoio é implicada pela OL-resolução, isto é, o Teorema 7.2 implica no Teorema 6.2, dexa-se esta prova como exercício.

7.6 DEDUÇÃO LINEAR E BUSCA EM ÁRVORE

Nas seções subsequentes, vai-se discutir um problema bastante importante da dedução linear, ou seja, como implementar dedução linear eficientemente.

Escolhendo a cláusula C_0 como cláusula de partida, pode-se encontrar todas as cláusulas laterais possíveis. Após resolver C_0 com estas possíveis cláusulas laterais, obtem-se os resolventes R_1 , ... , R_m. Todo R_i, $1 \leq i \leq m$, é uma possível cláusula central que pode levar a uma prova. Se algum R_i é a cláusula vazia, chegou-se ao objetivo. Caso contrário, para todo i, encontra-se todas as possíveis cláusulas laterais que podem ser resolvidas com R_i e continua-se o processo até que a cláusula vazia é gerada. De maneira a melhor descrever o processo acima, uma árvore de dedução linear mostrada na figura 7.7 será representada por um caminho como mostrado na figura 7.12, isto é, a cláusula lateral B_i é associada a um arco no caminho e C_i é um resolvente de C_{i-1} e B_{i-1} . Na seqüência, vai-se implementar a OL-dedução.

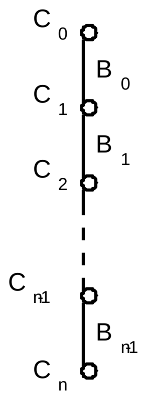

Figura 7.12 Caminho onde a cláusula lateral B_i é associada a um arco no caminho e C_i é um resolvente de C_{i-1} e B_{i-1}

Vai-se primeiro considerar um exemplo, no qual tautologias são incluídas com propósitos ilustrativos. Considere o seguinte conjunto de cláusulas:

(1) $P \vee Q$
(2) $\sim P \vee Q$
(3) $P \vee \sim Q$
(4) $\sim P \vee \sim Q$

Escolhendo a cláusula (1) como sendo a cláusula de partida C_0. As cláusulas ordenadas que podem ser consideradas cláusulas laterais de C_0 são (3) e (4). Resolvendo C_0 com as cláusulas (3) e (4), obtem-se a árvore mostrada na figura 7.13. Foram geradas duas cláusulas centrais. Para cada uma destas cláusulas, pode-se encontrar todas as cláusulas laterais a serem resolvidas e continuar a resolução mostrada na árvore da figura 7.13, gerando a árvore da figura 7.14.

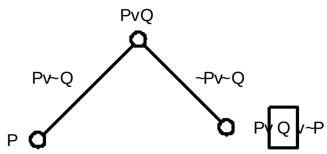

Figura 7.13 Árvore de Dedução considerando a cláusula (1) como sendo a cláusula de partida C_0 e como cláusulas laterais (3) e (4)

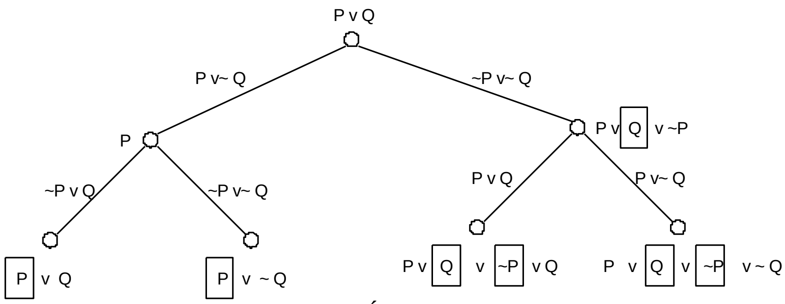

Figura 7.14 Continuação da Árvore de Dedução da figura 7.13

Quando um resolvente ordenado é gerado, sempre se checa se ele é redutível. Se for, sempre se reduz a cláusula ordenada reduzida correspondente. Na figura 7.14, é redutível. Sua cláusula ordenada reduzida é P. Portanto a árvore da figura 7.14 pode ser simplificada para a árvore mostrada na figura 7.15.

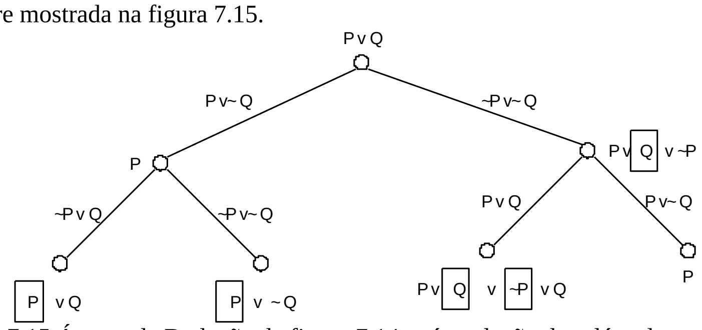

Figura 7.15 Árvore de Dedução da figura 7.14 após redução das cláusulas redutíveis

Agora existem quatro cláusulas centrais possíveis. Novamente, para cada cláusula central, encontra-se todas as cláusulas possíveis de se resolver. Deste modo obtem-se a árvore mostrada na figura 7.16.

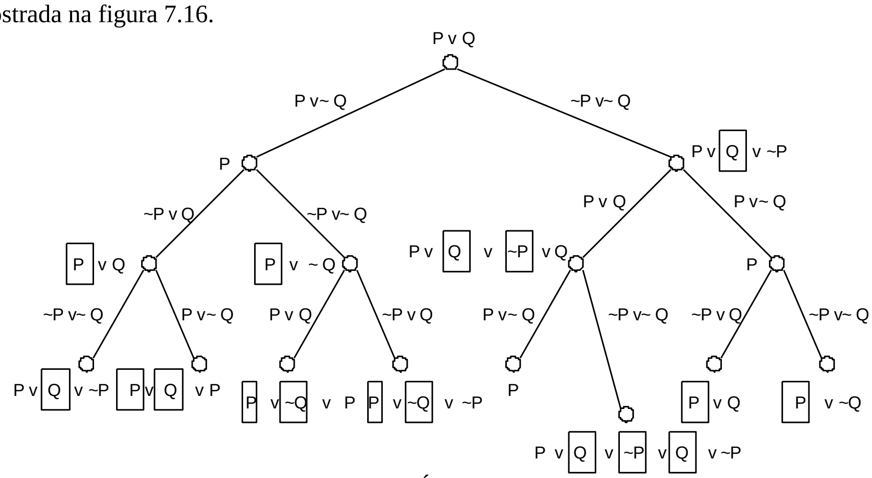

Figura 7.16 Continuação da Árvore de Dedução da figura 7.15

Os resolventes ordenados e são redutíveis. Fazendo a redução, obtem-se a árvore mostrada na figura 7.17.

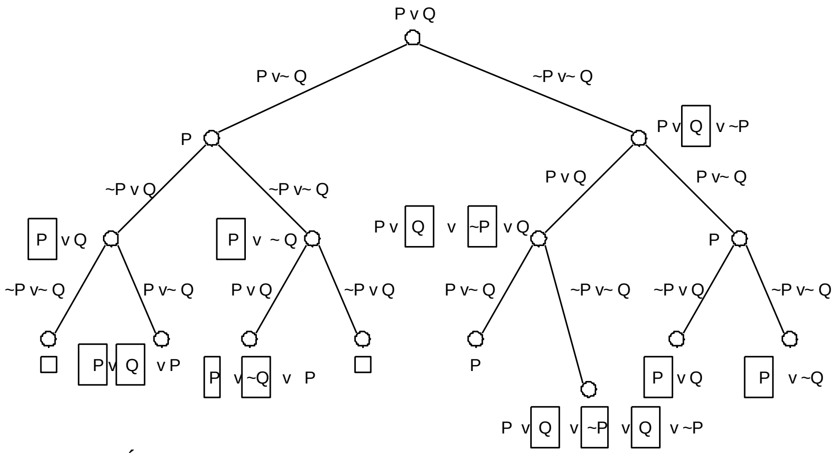

Figura 7.17 Árvore de Dedução da figura 7.16 após redução das cláusulas redutíveis

Uma vez que a cláusula vazia foi gerada, o processo é encerrado. O ramo mais a esquerda da árvore da figura 7.17 corresponde a OL-refuração mostrada na figura 7.18.

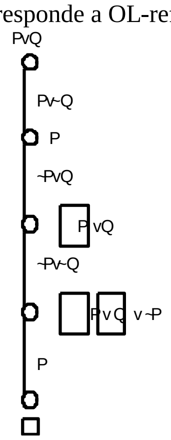

Figura 7.18 OL-refutação obtida a partir da figura 7.17

Nota-se que o problema de encontrar uma OL-refutação pode ser visto como uma problema de busca em árvore. Várias técnicas desenvolvidas originalmente para busca em árvore podem ser aplicadas à OL-dedução.

No exemplo anterior o método utilizado para gerar a árvore da figura 7.17 é chamado de método *breadth-first* (busca em largura). Na terminologia de busca em árvore, C_0 é o *nó inicial* da árvore. As cláusulas laterais podem ser consideradas como *operadores*, que podem ser usados para gerar os nós sucessores de um nó. Sendo C uma cláusula ordenada que representa um nó, encontre todas as cláusulas laterais possíveis de resolver com C e gere os resolventes ordenados R_1, ... ,R_m. R_1 , ... , R_m são chamados de *nós sucessores imediatos* de C. Quando todos os possíveis sucessores imediatos de um nó C são gerados, diz-se que o nó C foi *expandido*. Isto é, expandir um nó significa gerar todos os seus sucessores imediatos.

Supondo que S é um conjunto de cláusulas a ser provado e C_0 é uma cláusula ordenada em S, selecionada como cláusula de partida, então o método breadth-first pode ser descrito como segue:

Método de Busca em Largura (Breadth-First)

Passo 1: Faça CLIST = (C_0).
Passo 2: Se CLIST é vazio, pare sem obter uma prova. Caso contrário, continue.
Passo 3: Faça C ser a primeira cláusula ordenada em CLIST. Elimine C de CLIST.
Passo 4: Encontre todas as cláusulas ordenadas em S que podem ser cláusulas laterais em C. Se não existir nenhuma destas cláusulas, vá para o Passo 2. Caso contrário, resolva C com todas estas cláusulas laterais. Faça com que R_1 , ... , R_m representem estes resolventes ordenados. Faça R_i^* ser a cláusula ordenada reduzida de R_i se ela for redutível. Se R_i não for redutível, faça $R_i^*= R_i$.
Passo 5: Se algum R_q^* é uma cláusula vazia, $1 \leq q \leq m$, pare, pois a prova foi obtida; caso contrário, continue.
Passo 6: Coloque R_1^* , ... , R_m^* (em uma ordem aleatória) no final de CLIST e vá para o Passo 2.

Uma prova mínima para S com cláusula de partida C_0 é uma OL-refutação (a partir de S com cláusula de partida C_0) que envolve o menor número de resoluções. O método breadth-first irá encontrar sempre a prova mínima se esta prova existir. Entretanto, obviamente, muitas cláusulas irrelevantes e redundantes podem ser geradas pelo método de busca em largura antes da cláusula vazia ser gerada. Isto fica evidente noexemplo anterior. Outra maneira de busca em árvore é o *método de busca em profundidade* (*depth first method*). A idéia é que, de modo inverso ao método de busca em largura que expande os nós de cima para baixo, o método de busca em profundidade expande os nós da esquerda para a direita. O método de busca em profundidade é quase o mesmo que o de busca em largura, exceto que no Passo 6 coloca-se R_1^* , ... , R_m^* no início de CLIST, ao invés de colocá-los no final. Em adição, deve-se fornecer ao método de busca em profundidade um *limitante de profundidade* com o intuito de evitar expandir nós de um caminho infrutífero (ramo), isto é, um caminho que não pode levar à cláusula vazia. A profundidade de uma cláusula em um OL-dedução é definida como segue:

Definição: Em uma OL-dedução com uma cláusula de partida C_0, a *profundidade* de C_0 é 0. Se a profundidade de uma certa cláusula ordenada é k e R é um resolvente ordenado de C e alguma cláusula lateral, então a *profundidade* de R é (k + 1). O *tamanho* de uma prova (refutação) com a cláusula de partida C_0 é a profundidade da cláusula vazia.

Sendo d^* um limiar especificado anteriormente, o algoritmo de busca em profundidade pode ser descrito como segue:

Método de Busca em Profundidade (Depth-First)

Passo 1: Faça CLIST = (C_0).
Passo 2: Se CLIST é vazio, pare sem obter uma prova. Caso contrário, continue.
Passo 3: Faça C ser a primeira cláusula ordenada em CLIST. Elimine C de CLIST. Se a profundidade de C é maior que d^*, vá para o Passo 2. Caso contrário, continue.
Passo 4: Encontre todas as cláusulas ordenadas em S que podem ser cláusulas laterais em C. Se não existir nenhuma destas cláusulas, vá para o Passo 2. Caso contrário, resolva C com todas estas cláusulas laterais. Faça com que R_1 , ... , R_m representem estes resolventes ordenados. Faça R_i^* ser a cláusula ordenada reduzida de R_i se ela for redutível. Se R_i não for redutível, faça $R_i^*= R_i$.
Passo 5: Se algum R_q^* é uma cláusula vazia, $1 \leq q \leq m$, pare, pois a prova foi obtida; caso contrário, continue.

Passo 6: Coloque R_1^* , ... , R_m^* (em uma ordem aleatória) no início de CLIST e vá para o Passo 2.

É óbvio que aplicando o método de busca em profundidade com $d^* = 2$ na árvore da figura 7.17 será gerada a árvore mostrada na figura 7.19. Fica claro que a árvore da figura 7.19 é menor que a da figura 7.17.

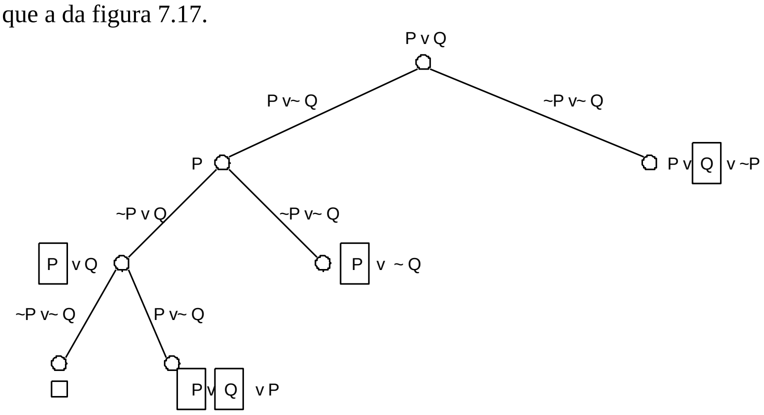

Figura 7.19 Aplicação o método de busca em profundidade com $d^* = 2$ na árvore da figura 7.17

Apesar de, geralmente, o método de busca em profundidade explorar uma árvore menor que o método de busca em largura faria, pode-se melhorá-lo ainda mais. Nota-se que no método de busca em profundidade, quando um nó é escolhido para expansão, ele é completamente expandido. Isto é, todos os possíveis sucessores são gerados. Entretanto, pode-se modificá-lo de maneira que ele gere somente um sucessor de cada vez. Considere a figura 7.19. O nó de partida da árvore é P ∨ Q. Existem dois sucessores deste nó de partida, que são, P e . Suponha que somente P seja gerado. Agora, P é uma cláusula central. Esistem dois sucessores do nó P. Suponha que somente é gerado, novamente tem somente dois sucessores, que são, □ e . Se somente □ for gerada, obtém-se a cláusula vazia. O processo como um todo é mostrado na figura 7.20, que é ainda menor que a árvore da figura 7.19. Devido a consideração acima, mostra-se um método de busca em profundidade modificado, onde d^* é um limiar previamente especificado.

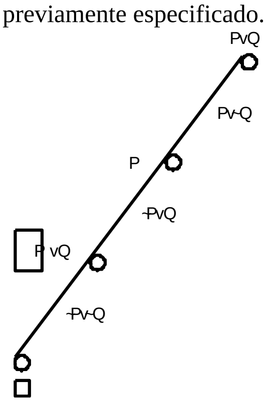

Figura 7.20 Aplicação o método de busca em profundidade gerando um sucessor de cada vez

Método de Busca em Profundidade Modificado (Modified Depth-First)

Passo 1: Encontre todas as cláusulas ordenadas em S que podem ser cláusulas laterais de C_0. Se não existir nenhuma destas cláusulas, pare sem obter uma prova. Caso contrário, seja B_0^1 , ... , B_0^r todas estas cláusulas laterais. Construa pares (C_0 , B_0^1) , ... , (C_0 , B_0^r) e faça CLIST ser a lista de todos estes pares arranjados em uma ordem aleatória.
Passo 2: Se CLIST é vazio, pare sem obter uma prova. Caso contrário, continue.
Passo 3: Faça (C , B) ser o primeiro par em CLIST. Elimine (C , B) de CLIST. Se a profundidade de C é maior que d^*, vá para o Passo 2. Caso contrário, continue.
Passo 4: Resolva C com B. Faça com que R_1 , ... , R_m representem todos os resolventes ordenados de C contra B. Faça R_i^* ser a cláusula ordenada reduzida de R_i se ela for redutível. Caso contrário, faça $R_i^* = R_i$.
Passo 5: Se algum R_q^* é uma cláusula vazia, $1 \leq q \leq m$, pare, pois a prova foi obtida; caso contrário, continue.
Passo 6: Para cada i = 1, 2, ... , m, encontre as cláusulas ordenadas em S que podem ser cláusulas laterais de R_i^*. Se não existir nenhuma destas cláusulas, elimine R_i^*. Caso contrário, faça B_{i1} ,B_{i2} , ... , B_{is_i} ser o conjunto de todas estas cláusulas laterais. Construa pares (R_i^*, B_{i1}) , ... , (R_i^* , B_{is_i}). Coloque todos estes pares (em uma ordem aleatória) no início de CLIST e vá para o Passo 2.

Na maioria das situações, tanto o básico quanto o método modificado de busca em profundidade são melhores que o método de busca em largura. Se o conjunto de cláusulas ordenado S tem uma prova com um tamanho menor que d^* , ambas as versões do método de busca em profundidade garantem encontrar esta prova. Apesar o método de busca em profundidade modificado ser mais difícil de implementar que o método básico, ele é bastante superior. Na próxima seção, assume-se que o método de busca em profundidade modificado é usado.

7.7 HEURÍSTICAS NA BUSCA EM ÁRVORE

Nesta seção, vai-se discutir "heurísticas" que podem ser usadas em conjunto com o método de busca em profundidade modificado no sentido de acelerar o processo de busca. O uso de uma heurística pode não garantir encontrar a prova, até mesmo se ela existir. Apesar disto, ela é, geralmente, uma ferramenta necessária para se provar teoremas grandes e difíceis, isto é, teoremas com provas longas. Existem várias heurísticas no campo de prova de teoremas e vai-se discutir apenas algumas delas, que servirão de base para que você, até mesmo desenvolva suas próprias heurísticas. Note que apesar destas heurísticas serem introduzidas para o uso com o método de busca em profundidade modificado, elas podem ser aplicadas a outros tipos de resolução, não necessariamente resolução linear.

7.7.1 Estratégia de Eliminação

Uma cláusula ordenada C_1 é dita *englobar* outra cláusula ordenada C_2 se e somente se a cláusula consistindo somente de literais não emoldurados de C_1 engloba a cláusula consistindo somente de literais não emoldurados de C_2. Uma cláusula ordenada é dita ser uma *tautologia* se e somente se contém um par complementar de literais não emoldurados. Uma vez que tautologias e cláusulas englobadas são irrelevantes e redundantes, elas devem ser eliminadas sempre que possível. No método de busca em profundidade modificado, no fim do Passo 4 pode-se checar se R_i^* é uma tautologia, i = 1, ... , m. Se R_i^* é uma tautologia, ele deve ser eliminado. Mais ainda, se existe um par (C , B) em CLIST, tal que R_i^* é englobado por C, então, R_i^* também deve ser eliminado.

7.7.2 Estratégia da Preferência por Poucos-Literais

No método de busca em profundidade modificado, os pares de CLIST são arranjados em uma ordem arbitrária. Na verdade, poder-se-ia arranjar os pares em CLIST de maneira que pares "potencialmente" bons apareçam antes dos "ruins". Assim, no Passo 3 do método, quando se pega o primeiro par de CLIST, pode-se estar certo que se tem nas mãos um dos "bons" que pode levar ao encontro de uma cláusula vazia rapidamente.

Sendo o tamanho de uma cláusula ordenada E, representado como tamanho(E) e definido como o número de literais não emoldurados em E. Suponha que (C , B) é um par em CLIST. Seja R representando um resolvente ordenado de C contra B. Uma vez que tamanho(R) indica o número mínimo de resoluções necessárias para levar R até a cláusula vazia, pode-se usar tamanho(R) como um indicardor de qualidade do par (C , B). Isto é, quanto menor o valor de tamanho(R), melhor será o par (C , B). Na verdade, na prática, não é necessário se calcular R para obter o tamanho(R). Uma estimativa de tamanho(R) será suficiente. De modo claro:

$$\textbf{tamanho(R)} \leq \textbf{tamanho(C)} + \textbf{tamanho(B)} - \mathbf{2} \qquad (7.1)$$

Mais ainda, uma vez que necessita-se somente de um número relativo, pode-se simplesmente usar (tamanho(C) + tamanho(B)) como medida de qualidade. Ou seja,

$$\boldsymbol{f}\textbf{(C,B)} = \textbf{tamanho(C)} + \textbf{tamanho(B)} \qquad (7.2)$$

Então f(C,B) será utilizada para avaliar a qualidade do par (C , B). No método de busca em profundidade modificado, antes do Passo 3 pode-se arranjar os pares de CLIST colocando os pares que tenham menores valores de f antes daqueles com maiores valores de f. Este esquema de ordenação é chamado de *Estratégia de Preferência por Poucos-Literais*.

7.7.3 Uso de Funções de Avaliação Heurística

Na discussão acima, a única medida de qualidade de uma cláusula é o número de literais que ela contém, o que é, convenhamos, muito simples. Para um par (C , B) em CLIST do método de busca em profundidade modificado, define-se h^*(C,B) como sendo o número mínimo de resoluções em uma prova mínima (refutação) tendo como cláusula de partida C e como primeira cláusula lateral B. Se h^*(C,B) é conhecido, então pode-se usar h^*(C,B) para ordenar os pares de CLIST, entretanto, geralmente, h^*(C,B) não é conhecido. Portanto, h^*(C,B) deve ser estimado. Sendo h(C,B) uma estimativa de h^*(C,B), que geralmente é obtida a partir de um conjunto de dados conhecidos. Na seqüência, vai-se considerar como obter h(C,B). Suponha que h(C,B) pode ser expresso como:

$$\boldsymbol{h}\textbf{(C,B)} = \boldsymbol{w_0} + \boldsymbol{w_1 f_1}\textbf{(C,B)} + \ldots + \boldsymbol{w_n f_n}\textbf{(C,B)} \qquad (7.3)$$

onde cada f_i, i = 1, ... , n, é uma função avalidada corretamente de C e B, e w_i é um peso associado a f_i. Cada f_i é chamado de *atributo* do par (C,B). Geralmente, atributos são selecionadas ad hoc pelo pesquisador que acredita que elas sejam relevantes para h^*(C,B). Por exemplo, pode-se usar alguns atributos sugeridos na tabela 7.1. Na verdade h(C,B) pode ser uma função linear ou não-linear de f_1(C,B), ... , f_n(C,B). Para os propósitos deste texto vai-se considerar somente funções lineares na seqüência. A função h(C,B) pode ser usada para ordenar os pares de CLIST, isto é, para colocar os pares com pequenos valores de h antes daqueles com valores maiores de h. Chama-se h(C,B) de *função de avaliação heurística*, ou *função de avaliação* de maneira resumida. Na próxima seção, ver-se-á um método para obter um h(C,B).

Tabela 7.1 Sugestões de atributos f_i a serem usados na composição de h(C,B)

1. Número de literais não emoldurados em C
2. Número de literais emoldurados em C
3. Número de cláusulas laterais de C
4. Número de constantes no último literal de C
5. Número de funções no último literal de C
6. Número de literais emoldurados de C, que englobam o último literal de C
7. Número de variáveis diferentes em C e B
8. tamanho(C) + tamanho(B) - 2
9. Número de constantes em C / (1 + número de variáveis em C)
10. Número de constantes em C / (1 + número de variáveis distintas em C)
11. Profundidade de C
12. Número de literais não emoldurados que estão tanto em C quanto em B
13. Número de literais em B, que tem um complemento emoldurado em C
14. Número de predicados distintos em C e B

7.8 ESTIMATIVAS DE FUNÇÕES DE AVALIAÇÃO

Como discutido na seção 7.7.3, assume-se que:

$$h(C,B) = w_0 + w_1 f_1(C,B) + ... + w_n f_n(C,B).$$

Nesta seção, discute-se duas importantes questões:

1. Dado um conjunto de atributos, como pode-se determinar os valores apropriados para w_0, w_1, ... , w_n ?
2. Como pode-se saber se um conjunto de atributos é bom (relevantes com relação a h^*(C,B)) ou não ?

Vai-se agora responder a primeira questão. Assuma que se conhece os valores de $h^*(C_1,B_1)$, ... , $h^*(C_q,B_q)$. Então o problema é determinar w_0, w_1, ... , w_n tal que:

$$\sum_{i=1}^{q} \mathbf{[\ } \boldsymbol{h^*}\mathbf{(C_i,B_i) - (}\boldsymbol{w_0}\mathbf{ + }\boldsymbol{w_1 f_1}\mathbf{(C_i,B_i) + ... + }\boldsymbol{w_n f_n}\mathbf{(C_i,B_i))]^2} \qquad (7.4)$$

seja mínimo. Esta estimativa é chamada de *estimativa de quadrado-mínimo*. Seja as matrizes F, H e W definidas como segue:

$$F = \begin{bmatrix} 1 f_1(C_1,B_1) \cdots f_n(C_1,B_1) \\ 1 f_1(C_2,B_2) \cdots f_n(C_2,B_2) \\ \vdots \\ 1 f_1(C_q,B_q) \cdots f_n(C_q,B_q) \end{bmatrix}, \quad H = \begin{bmatrix} h^*(C_1,B_1) \\ h^*(C_2,B_2) \\ \vdots \\ h^*(C_q,B_q) \end{bmatrix}, \quad W = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

Portanto o valor de W que minimiza a expressão (7.4) é dado por:

$$\mathbf{W = (F'\ F)^{-1}\ F'\ H} \qquad (7.5)$$

onde F' é a transposta de F e $(F'\ F)^{-1}$ é a inversa de (F' F). Substituindo W em (7.3), uma função de avaliação pode ser obtida.

Exemplo 7.7: Considere a árvore mostrada na figura 7.17. Vai-se redesenhar a árvore na figura 7.21, onde cada cláusula ordenada é numerada. Para este exemplo, suponha que se usou os atributos definidos como segue:

f_1(C,B) = tamanho(C) + tamanho(B) - 2
f_2(C,B) = número de literais emoldurados em C
f_3(C,B) = número de literais não emoldurados que estão tanto em C quanto em B
f_4(C,B) = número de literais em B, que tem um complemento emoldurado em C
f_5(C,B) = número de literais emoldurados em C, que englobam o último literal de C

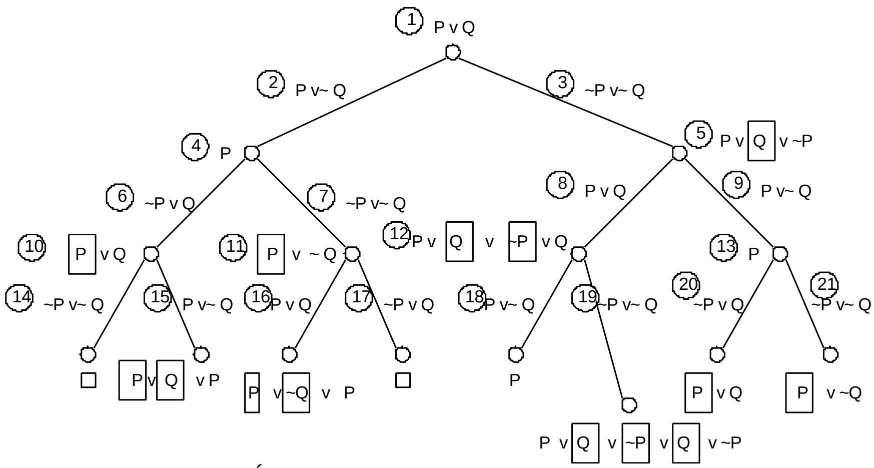

Figura 7.21 Árvore da figura 7.17 com as cláusulas numeradas

Calculando os valores dos atributos de cada par na árvore da figura 7.21, obtem-se a tabela 7.2.

Tabela 7.2 Valores para os atributos de cada para da árvore da figura 7.21

i	par(C_i,B_i)	$f_1(C_i,B_i)$	$f_2(C_i,B_i)$	$f_3(C_i,B_i)$	$f_4(C_i,B_i)$	$f_5(C_i,B_i)$	$h^*(C_i,B_i)$
1	(1,2)	2	0	1	0	0	3
2	(1,3)	2	0	0	0	0	4
3	(4,6)	1	0	0	0	0	2
4	(4,7)	1	0	0	0	0	2
5	(5,8)	2	1	1	0	0	4
6	(5,9)	2	1	1	1	0	3
7	(10,14)	1	1	0	1	0	1
8	(10,15)	1	1	0	0	0	3
9	(11,16)	1	1	0	0	0	3
10	(11,17)	1	1	0	1	0	1
11	(12,18)	2	2	1	2	1	3
12	(12,19)	2	2	0	1	1	4
13	(13,20)	1	0	0	0	0	2
14	(13,21)	1	0	0	0	0	2

Calculando a fórmula (7.5) em um computador, obtem-se:

w_0 = 0,30 w_1 = 1,68 w_2 = 0,76 w_3 = -0,24 w_4 = -1,44 w_5 = 0,60 (7.6)

Deste modo tem-se:

h(C,B) = 0,30 + 1,68f_1(C,B) + 0,76f_2(C,B) - 0,24f_3(C,B) - 1,44f_4(C,B) + 0,60f_5(C,B) (7.7)

No método de busca em profundidade modificado, logo antes do Passo 3, se for usada a função *h*(C,B) para ordenar os pares de CLIST, então a árvore de busca do método de busca em profundidade modificado será aquela mostrada pela figura 7.22 (a) ou (b).

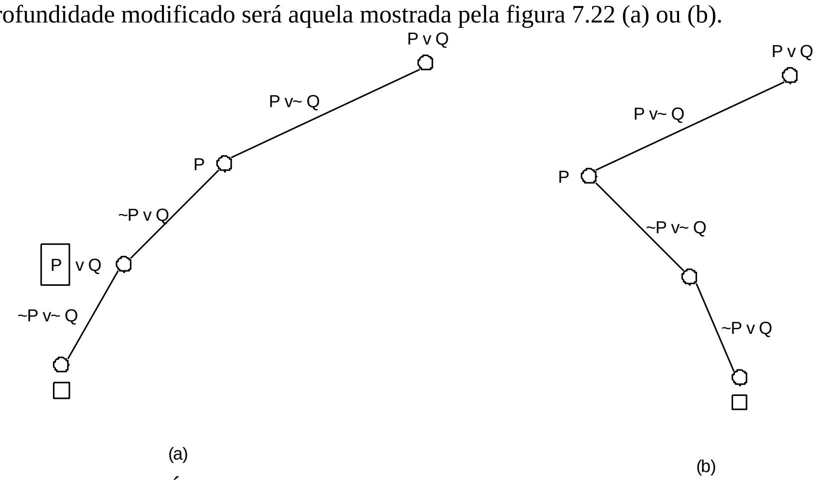

Figura 7.22 Árvores de busca usando as funções de avaliação da tabela 7.2

No exemplo acima, mostrou-se como obter uma função de avaliação. A função de avaliação obtida no exemplo 7.7 é somente para conjuntos de cláusulas *base*. Elas estão baseadas apenas em um exemplo, é claro que, quanto mais exemplos se usar para estimar uma função de avaliação, melhor ela será. Apesar disso, aplica-se a função de avaliação obtida no exemplo 7.7 a um novo exemplo para ver o quão boa esta estimativa é.

Exemplo 7.8: Considere o seguinte conjunto de cláusulas ordenadas:

(1) $\sim P_1 \vee \sim P_2 \vee P_3$

(2) P_1

(3) P_2

(4) $\sim P_1 \vee \sim P_3$

(5) $\sim P_3 \vee P_4$

(6) $\sim P_3 \vee P_5$

(7) $\sim P_4 \vee P_5$.

Sendo a cláusula (1) a cláusula de partida, existem três cláusulas ordenadas que podem ser cláusulas laterais de (1), são elas: (4), (5) e (6). Usa-se agora *h*(C,B) definido pela equação 7.7 para calcular o seguinte:

$h(1,4) = 0,30 + 1,68(3+2-2) + 0,76(0) - 0,24(1) - 1,44(0) + 0,60(0) = 3,42$

$h(1,5) = 0,30 + 1,68(3+2-2) + 0,76(0) - 0,24(0) - 1,44(0) + 0,60(0) = 3,66$

$h(1,6) = 0,30 + 1,68(3+2-2) + 0,76(0) - 0,24(0) - 1,44(0) + 0,60(0) = 3,66$

Uma vez que *h*(1,4) é o menor valor, escolhe-se a cláusula (4) como cláusula lateral da cláusula (1) e obtem-se a seguinte cláusula:

(8) $\sim P_1 \vee \sim P_2$ a partir de (1) e (4).

O restante da prova é trivial. Você pode facilmente ver que os seguintes passos completam a prova:

(9) $\sim P_1$ a partir de (8) e (3).

(10) $\square$ a partir de (9) e (2).

Para este exemplo, nenhuma cáusula irrelevante foi gerada. Nota-se que se uma função de avaliação fizesse com que fosse escolhida a cláusula (6) ou (5) como cláusula lateral da cláusula (1), cláusulas irrelevantes poderiam ser geradas. Assim, mostrou-se que a função de avaliação também é útil para este exemplo.

Pode-se agora responder a segunda questão: Como pode-se saber se um conjunto de atributos é bom (relevantes com relação a $h^*(C,B)$) ou não? Evidentemente, se um conjunto de atributos é bom, então após usá-los em um grande número de exemplos para estimar o valor da função de avaliação, a função deve se generalizar para novos exemplos. Isto é, a função de avaliação deve apresentar bom desempenho sobre novos exemplos. Se ela não se generalizar para novos problemas, então o pesquisador deve analisar os dados e considerar novos atributos.

7.9 EXERCÍCIOS

Seção 7.2

1. Use resolução linear para provar que os seguintes conjuntos de cláusulas são insatisfatíveis:

(a) S = {~P ∨ Q, P ∨ R, ~Q, ~R}
(b) S = {P ∨ Q ∨ R, ~P, ~Q, ~R ∨ W, ~W}
(c) S = {P ∨ Q, Q ∨ R, R ∨ W, ~R ∨ ~P, ~W ∨ ~Q, ~Q ∨ ~R}

2. Use resolução linear para provar a insatisfiabilidade do seguinte conjunto de cláusulas:

(a) ~E(x) ∨ V(x) ∨ S(x,f(x))
(b) ~E(x) ∨ V(x) ∨ C(f(x))
(c) P(a)
(d) E(a)
(e) ~S(a,y) ∨ P(y)
(f) ~P(x) ∨ ~V(x)
(g) ~P(x) ∨ ~C(x).

Use a cláusula (g) como cláusula de partida.

3. Prove a insatisfiabilidade do seguinte conjunto de cláusulas por resolução linear. Use a última cláusula como cláusula de partida.

(a) P(x,I(x),e)
(b) ~S(x) ∨ ~S(y) ∨ ~P(x,I(y),z) ∨ S(z)
(c) S(a)
(d) S(e).

Seção 7.3

4. Prove a insatisfiabilidade do seguinte conjunto de cláusulas por resolução por entrada.

(a) S = { P(I(x),x,e),
P(e,x,x),
~P(x,y,u) ∨ ~P(y,z,v) ∨ ~P(u,z,w) ∨ P(x,v,w),
~P(x,y,u) ∨ ~P(y,z,v) ∨ ~P(x,v,w) ∨ P(u,z,w),
~P(a,x,e) }

(b) S = { D(x,x),
~D(x,y) ∨ ~D(y,z) ∨ D(x,z),
P(x) ∨ D(g(x),x),
P(x) ∨ L(b,g(x)),
P(x) ∨ L(g(x),x),
L(b,a),
~P(x) ∨ ~D(x,a),
~L(b,x) ∨ ~L(x,a) ∨ P(f(x)),
~L(b,x) ∨ ~L(x,a) ∨ D(f(x),x) }

5. Refaça o exercício 4 usando resolução unitária.
6. Sendo S um conjunto insatisfatível de cláusulas base, L uma cláusula untária em S e S' o conjunto obtido a partir de S pela eliminação das cláusulas contendo L e pela eliminação de ~L das cláusulas restantes. Prove que se existe uma refutação unitária a partir de S, também deve existir uma refutação unitária a partir de S'.

Seção 7.4
7. Prove a insatisfiabilidade do seguinte conjunto ordenado de cláusulas por OL-dedução.

P(e,x,x)
P(x,e,x)
P(x,I(x),e)
P(I(x),x,e)
S(b)
~S(x) ∨ ~S(y) ∨ ~P(x,I(y),z) ∨ S(z)
~P(x,y,u) ∨ ~P(y,z,v) ∨ ~P(x,v,w) ∨ P(u,z,w)
~P(x,y,u) ∨ ~P(y,z,v) ∨ ~P(u,z,w) ∨ P(x,v,w),
~S(I(b)).

8. Prove a insatisfiabilidade do seguinte conjunto de cláusulas por OL-dedução:

P(x,I(x),e)
P(e,x,x)
~P(x,y,u) ∨ ~P(y,z,v) ∨ ~P(u,z,w) ∨ P(x,v,w),
~P(x,y,u) ∨ ~P(y,z,v) ∨ ~P(x,v,w) ∨ P(u,z,w)
~P(a,e,a).

9. Prove o Lema 7.2.
10. Prove que o Teorema 7.2 implica no Teorema 6.2.

Seção 7.6
11. Considere o seguinte conjunto de cláusulas ordenadas:

S = {P ∨ Q, P ∨ ~Q ∨ R, P ∨ ~Q ∨ ~R, ~P ∨ R, ~P ∨ ~R}.

Sendo P ∨ Q a cláusula ordenada de partida e $d^* = 6$. Prove que S é insatisfatível através do:
(a) método de busca em largura;
(b) método de busca em profundidade;
(c) método de busca em profundidade modificado.

Seção 7.8
12. Seja S o conjunto de cláusulas ordenadas abaixo:

P ∨ Q ∨ R
P ∨ Q ∨ ~R
P ∨ ~Q ∨ R
P ∨ ~Q ∨ ~R
~P ∨ Q ∨ R
~P ∨ Q ∨ ~R
~P ∨ ~Q ∨ R
~P ∨ ~Q ∨ ~R

Sendo P ∨ Q ∨ R a cláusula ordenada de partida e usando a função de avaliação definida pela equação 7.7 para ordenar os pares em CLIST, encontre a árvore de busca do método de busca em profundidade modificado.

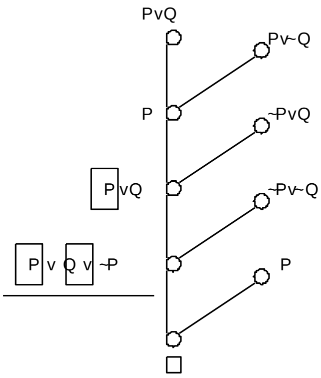

Figura 7.6 Refutação linear para S = {P ∨ Q, P ∨ ~Q, ~P ∨ Q, ~P ∨ ~Q} usando informação dos literais resolvidos e o conceito de cláusulas ordenadas.