

O Método dos Tableaux Generalizado e sua Aplicação ao Raciocínio Automático em Lógicas Não Clássicas

Arthur Buchsbaum e Tarcisio Pequeno¹

O método dos tableaux, concebido por Beth [2], aprimorado e divulgado principalmente por Hintikka [10] e Smullyan [13], é um método de prova por refutação, no qual um teorema é provado pelo insucesso na tentativa de construção sistemática de um modelo para a sua negação. O método tem fundamento semântico : ao tentar provar que α é um teorema lógico (ou que α é teorema em uma teoria Γ), o que o método, em sua concepção original, faz de fato é verificar a impossibilidade da satisfação de $\neg \alpha$ (ou da satisfação simultânea de Γ e $\neg \alpha$).

Além da sua utilidade na prova de teoremas, a forma exaustiva e sistemática com que o espaço de construção de modelos é percorrido faz do método dos tableaux um valioso instrumento na determinação de resultados negativos em cálculos decidíveis (ou em fragmentos decidíveis de cálculos indecidíveis). Nesses casos, a falha na tentativa de refutação de $\neg \alpha$ (ou alguma variante dessa expressão de acordo com a generalização aqui introduzida) pode ser efetivamente decidida após um número finito de passos. Uma tal falha indica, na realidade, a impossibilidade dessa refutação, permitindo estabelecer que α não é teorema do cálculo. Além disso, em decorrência da tentativa infrutífera, um contra-exemplo é construído.

Resultados negativos são, via de regra, difíceis de obter e o apelo a critérios semânticos se faz em geral necessário. Veremos que, embora de inspiração semântica, o método dos tableaux generalizado é aplicável mesmo a cálculos que não disponham de semântica. Sistemas de tableaux podem ser derivados e/ou justificados diretamente das regras dos cálculos.

1 Departamento de Informática, PUC-Rio.

O método dos tableaux foi concebido originalmente para o cálculo proposicional e o cálculo de predicados clássicos. Posteriormente, foram desenvolvidas variantes aplicáveis a outros sistemas lógicos, como por exemplo em Fitting [9] para lógicas modais e em Buchsbaum & Pequeno [3], [4] e [5] para lógicas paraconsistentes e/ou paracompletas. Na realidade, uma das qualidades do método é exatamente a sua adaptabilidade a uma grande variedade de lógicas (ele não requer conversão para formas normais, por exemplo), como pretendemos aqui demonstrar. Essa flexibilidade o torna extremamente atraente ao projeto de automatização do raciocínio, o qual consiste na modelagem lógica do raciocínio, seguido do desenvolvimento de métodos de prova para a lógica empregada e da sua efetiva implementação na forma de um raciocinador automático.²

O projeto moderno de automatização do raciocínio é o sucessor do projeto de mecanização do pensamento concebido ao final da Idade Média por Ramón Lull (1235-1315), sob o nome de *ars magna*, e que teve outros importantes proponentes em Hobbes e na *ars combinatoria* de Leibniz. A idéia fundamental em suporte ao projeto de Lull/Hobbes/Leibniz é a de que o pensamento é uma manipulação de símbolos que pode ser reduzido a algum tipo de cálculo. Essa idéia permanece como hipótese de trabalho do projeto moderno sob a forma mais fraca de que o *raciocínio*, ou pelo menos certas formas de raciocínio, podem ser *simuladas* em um sistema de manipulação de símbolos. Outras características importantes do projeto original que permanecem na versão moderna são a percepção da necessidade do uso de linguagens artificiais formais e a compreensão de que a mecanização envolve problemas de natureza combinatória.

As chances de êxito do projeto moderno de mecanização do raciocínio, em oposição a seus prematuros precursores, repousam em duas conquistas essenciais : o progresso no conhecimento de sistemas simbólicos adquirido pela lógica matemática, teoria das linguagens formais e teoria da computação, e a disponibilidade de máquinas poderosas de manipulação de símbolos.

A lógica matemática desempenha nesse projeto um papel fundamental, tanto como instrumento conceitual quanto heurístico. O termo « lógica » designa aqui não um sistema lógico específico, mas a disciplina que estuda sistemas simbólicos com determinadas características. No âmbito da ciência da computação cunhou-se o termo « engenharia lógica » para designar a arte de elaboração e utilização de sistemas lógicos em modelagem e na especificação de sistemas. No nosso caso a modelagem em questão é a de certos padrões de raciocínio requeridos na construção de sistemas de inteligência artificial e/ou encontráveis no « raciocínio do senso comum ». Esses esquemas de

2 Um raciocinador automático é um programa que, munido de uma base de dados contendo « conhecimento » codificado em uma linguagem lógica (base de conhecimento), pode responder a perguntas efetuando raciocínio sobre esse conhecimento, de acordo com a lógica adotada.

raciocínio podem apresentar em suas lógicas subjacentes características não clássicas tais como não monotonicidade (a adição de novas informações que tornem o conhecimento mais acurado pode revogar conclusões previamente estabelecidas) e paraconsistência (a convivência de crenças contraditórias sem que isso acarrete a trivialização do sistema de crenças). Veja Pequeno [12].

Em tal ambiente de pluralidade lógica o método dos tableaux pode ser útil de diversas formas. Na modelagem do raciocínio tem se mostrado útil no desenho e na avaliação de novos cálculos (que em geral não dispõem, ou pelo menos ainda não dispõem de semânticas). A sua propriedade na determinação de resultados negativos o faz extremamente valioso na calibragem do cálculo de forma a se banir teoremas não desejáveis e incluir outros que o sejam. Pode ainda contribuir na avaliação de certas propriedades dos cálculos como redutibilidade (por exemplo, qual a natureza das fórmulas que não pertencem ao domínio de nenhuma das regras de decomposição?) e complexidade (qual a predominância das regras que dependem apenas das fórmulas às quais se aplicam em relação àquelas que dependem também das « histórias » das fórmulas nos tableaux?). Na construção de raciocinadores o método dos tableaux pode ser utilizado diretamente como veículo da implementação. Sua grande vantagem aí é novamente a flexibilidade. Por outro lado, já existem evidências da possibilidade de implementações eficientes de provadores de teoremas baseados em tableaux (veja Oppacher & Suen [11]).

A fim de atualizar a potencialidade do método dos tableaux procuramos abstrair suas características essenciais em uma generalização cujas instâncias podem ser aplicadas a uma grande variedade de lógicas. Aparentes limitações do método tais como a sua dependência de propriedades da negação clássica podem ser removidas pela utilização de extensões conservativas adequadas dos cálculos a serem tratados. No presente artigo apresentamos o método generalizado dos tableaux conforme o concebemos. Além disso, esboçamos esquemas gerais para provas de correção e completude de sistemas de tableaux com respeito aos cálculos lógicos correspondentes. O nosso próprio exercício na arte de elaborar sistemas de tableaux para diferentes lógicas nos permitiu divisar algumas heurísticas para a confecção de regras que são aqui oferecidas. À guisa de ilustração, incluímos o tratamento do cálculo de predicados clássico e do cálculo para completo P1 (da Costa & Marconi [8]). Outros exemplos podem ser encontrados em Buchsbaum & Pequeno [4].

Sistemas de tableaux

Um sistema de tableaux é um método de prova por refutação que consiste na geração de uma árvore (*tableau*), a partir de um *tableau inicial*, o qual é em muitos casos um nó consistindo na negação do teorema a ser demonstrado.

Passo a passo, após a escolha de um nó ainda não utilizado relacionado a alguma *regra*, o desenvolvimento da árvore se dá a partir de todas as folhas descendentes deste nó, localizadas em *ramos abertos*, onde, a partir de cada uma destas folhas, é acrescentada uma subárvore obtida pela aplicação da regra ao nó. A proliferação dos ramos é contida por uma *operação de fechamento*, pela qual um ramo é *fechado* conforme a ocorrência de uma certa condição, dada por um *critério de fechamento*. O objetivo do procedimento é fechar todos os ramos, provando daí o teorema dado. Semanticamente, o fechamento de todos os ramos do tableau estabelece a insatisfabilidade da fórmula figurando no nó raiz do tableau.

O método dos tableaux é dito ser *analítico*, visto que este decompõe gradualmente a fórmula dada, em oposição aos métodos convencionais de prova automática por resolução (Chang & Lee [6]). Este método lida diretamente com a fórmula dada, sem apelar, por exemplo, para a normalização em cláusulas.

A fim de definir um sistema de tableaux seis elementos devem ser fornecidos: linguagem inicial, linguagem de trabalho, domínio, função de inicialização, critério de fechamento e coleção de regras. Já vimos de modo informal nesta seção três destes elementos. A seguir daremos uma descrição rigorosa do método dos tableaux.

Um *tableau* em uma linguagem L é uma árvore finita cujos nós são n -uplas possuindo pelo menos dois componentes, onde um deles é uma fórmula de L e o outro é uma *marca* indicada por 0 ou 1. Um *ramo* em L é uma seqüência finita de tais n -uplas. Cada nó de um tableau pertence a um único *nível*, o qual é rotulado por algum número natural. Há exatamente um nó de nível 0, que é o *nó raiz* do tableau, chamado também de *nó inicial*. Cada nó de nível $n + 1$ é um *filho* (ou *sucessor*) de um único nó, que é de nível n . Dizemos que um nó é *terminal* se este não tem sucessores. Se um tableau tem ao menos um nó de nível p mas nenhum nó de nível maior, dizemos que p é a *profundidade* do tableau. Um *ramo de um tableau* é uma seqüência finita de nós N_0, \dots, N_k , tal que N_0 é o nó inicial do tableau e, para cada $i = 1, \dots, k$, N_i é um sucessor de N_{i-1} e N_k é terminal.

Seja L uma linguagem formal, L' uma extensão de L , \mathcal{F} a coleção de todas as fórmulas de L , \mathcal{F}' a coleção de todas as fórmulas de L' , τ a coleção de todos os tableaux em L' , $\mathcal{P}\mathcal{F}(\tau)$ a coleção de todos os subconjuntos finitos de τ , θ a coleção de todos os ramos em L' , Γ um subconjunto de \mathcal{F} , I uma função de Γ em τ , F uma função de θ em {aberto, fechado}, e finalmente R uma coleção de funções de $\mathcal{F}'_0 \times \theta$ em $\mathcal{P}\mathcal{F}(\tau)$, onde \mathcal{F}'_0 é um subconjunto de \mathcal{F}' variável em função dos elementos de R . Dizemos então que a sêxtupla ordenada $S = \langle L, L', \Gamma, I, F, R \rangle$ é um *sistema de tableaux*, onde L é a *linguagem inicial* de S , L' é a *linguagem de trabalho* de S , Γ é o *domínio* de S , I é a *função de inicialização* de S , F é o *critério de fechamento* de S e R é a *coleção de regras* de S . No resto desta seção, usaremos as letras S, L, L', Γ, I, F e R no sentido que acabamos de definir.

Seja N um nó de algum tableau em L' , A a sua fórmula e r uma regra de S (isto é, um elemento de R) tal que o domínio de r é $\mathcal{F}'_0 \times \theta$; dizemos então que r é aplicável a N se $A \in \mathcal{F}'_0$, e neste caso dizemos também que r é aplicável a A . Dizemos que A é uma *fórmula excluída* de S se não existe regra de S aplicável a A . Se $A \in \Gamma$, dizemos que $I(A)$ é o *tableau inicial* para A em S . Se ρ é um ramo em L' , dizemos que ρ está fechado em S se $F(\rho) = \text{fechado}$, caso contrário dizemos que ρ está *aberto* em S . Dado um nó N de um tableau, denotamos a fórmula de N por $\text{form}(N)$.

Dizemos que um nó de um tableau está marcado se o seu segundo componente é 1. Essa definição desempenha um papel equivalente à de *nó usado* em Bell & Machover [1]. Do ponto de vista algorítmico, é muito mais simples verificar se um nó é marcado do que se ele foi usado. Por exemplo, em um sistema de tableaux para a lógica clássica, dizemos que a fórmula $A \wedge B$ foi usada em um determinado ramo se neste ramo figurarem também as fórmulas A e B . Para verificar então se $A \wedge B$ foi usado em todos os ramos em que esta fórmula figura, devemos observar se as fórmulas A e B ocorrem nestes ramos. Por outro lado, para verificar se o nó contendo $A \wedge B$ foi marcado, basta observar se há uma marca, previamente estabelecida por convenção, neste nó. Na rotina do algoritmo do tableau, a marcação se dá no momento em que $A \wedge B$ é escolhida para o desenvolvimento da árvore. Dizemos que um tableau em L' está *exaurido* em S se todos os seus nós contendo fórmulas não excluídas estão marcados. De modo análogo definimos *ramo exaurido* em S . É evidente que um tableau está exaurido se, e somente se, todos os seus ramos estão exauridos.

Sejam T, T' tableaux em L' . Dizemos então que T' é uma *extensão imediata* de T em S se existe um nó N de T tal que N não está marcado e existe uma regra r de S aplicável a N tal que T' pode ser obtido de T marcando-se N e acrescentando-se $r(\text{form}(N), \rho)$ (note que uma regra é uma função que associa à fórmula de um nó e ao ramo no qual o nó figura uma coleção finita de tableaux) em cada ramo aberto ρ de T onde N figure. Dizemos que T' é um *desenvolvimento* de T em S se há uma seqüência de tableaux T_0, \dots, T_n tal que T_0 é T , T_n é T' e, para cada $i = 1, \dots, n$, T_i é uma extensão imediata de T_{i-1} em S . Dizemos que T é um *tableau para uma fórmula* A em S ($A \in \Gamma$) se T é um desenvolvimento do tableau inicial para A . Dizemos que T é uma *confutação* para A em S se T é um tableau para A em S e todos os seus ramos estão fechados em S . Dizemos que T é uma *confutação* se T é uma confutação para alguma fórmula em S .

As duas linguagens de um sistema de tableaux são seu ponto de referência básico, sem o qual não poderíamos, pelo menos de um modo simples, definir adequadamente os domínios das regras do sistema. A linguagem inicial é a linguagem da lógica à qual o sistema se aplica. A linguagem de trabalho é aquela efetivamente usada no sistema de tableaux. Existem duas razões pelas quais a linguagem de trabalho é necessária: 1ª) para o tratamento dos quantificadores é bastante conveniente o uso de novas constantes; 2ª) às vezes é preciso adicionar novos sinais lógicos ao cálculo dado para obter-se uma

extensão conservativa mais adequada ao trabalho com tableaux. Por exemplo, em certos cálculos não possuindo fórmulas insatisfatíveis, segundo as suas semânticas, é preciso procurar extensões conservativas possuindo fórmulas insatisfatíveis.

Uma regra é uma função que, no caso geral, depende do nó ao qual esta é aplicada e do ramo em que o nó figura. Por exemplo, se estivermos trabalhando com um sistema para a lógica clássica de primeira ordem, então uma regra aplicada a um nó contendo uma fórmula quantificada dá resultados que dependem dos ramos aos quais este nó pertence.

O critério de fechamento de ramos varia conforme a lógica para a qual o sistema de tableaux é destinado. Para a lógica clássica um ramo é fechado se contém duas fórmulas das formas A e $\neg A$, para o cálculo C1 (da Costa [7]) um ramo é fechado se este contém duas fórmulas das formas A e $\neg^* A$ (Buchsbaum [3]), onde \neg^* é uma negação equivalente à negação clássica que pode ser definida em C1 por $\neg^* A \equiv \neg A \wedge A^0$, para o cálculo C1* há uma outra variação na definição de ramo fechado, e assim por diante.

O domínio indica as fórmulas para as quais o sistema efetivamente funciona. Por exemplo, o primeiro sistema que definimos para C1* só trabalha com fórmulas fechadas.

Finalmente, o tableau inicial para uma fórmula dada também varia conforme a lógica para a qual o sistema foi projetado. Por exemplo, na lógica clássica sem igualdade o tableau inicial para uma fórmula A é uma árvore com um único nó, não marcado, cuja fórmula é A ; para a lógica clássica com igualdade, adaptando-se a abordagem de Bell & Machover [1] para o nosso contexto, temos que o tableau inicial é uma árvore com $n + 1$ nós, sendo n o número de sinais não lógicos de A .

O conceito de *nó composto*, definido abaixo, revela-se importante para provas de convergência a tableaux exauridos. Em nosso tratamento de tableaux, resolvemos considerar que cada nó contém exatamente uma fórmula. Esta abordagem apresenta certas desvantagens do ponto de vista teórico, mas reflete com mais fidelidade a realidade algorítmica do método de prova. Em Bell & Machover [1], os nós de um tableau contêm coleções de fórmulas, e a partir daí as provas de convergência de seqüências de tableaux tornam-se mais simples que no nosso caso. Procuramos contornar tais dificuldades tratando também destas coleções de fórmulas, às quais denominamos *nós compostos*.

Sejam r e s dois ramos em L tais que r é N_0, \dots, N_m , s é N'_0, \dots, N'_n ; então denotamos o ramo $N_0, \dots, N_m, N'_0, \dots, N'_n$ por $r + s$, e chamamos tal ramo de *concatenação* de r com s . Dizemos que dois ramos r e r' em L são *equivalentes* se eles têm o mesmo tamanho e possuem as mesmas fórmulas para os nós correspondentes, isto é, se há um n natural tal que r é N_0, \dots, N_n , r' é N'_0, \dots, N'_n e, para cada $i = 0, \dots, n$, a fórmula de N_i é igual à fórmula de N'_i . Um *segmento de um ramo* r de L é uma seqüência não vazia de nós consecutivos de r , e um *segmento inicial* de r é um segmento de r que inicia r .

Dizemos que um tableau T é de S se T é um tableau para alguma fórmula A em L . Dizemos que um ramo r em L' é de S se r é ramo de algum tableau de S . Dados dois ramos r e r' de S , dizemos que r' é uma *extensão imediata* de r se existem dois tableaux T e T' de S tal que T' é uma extensão imediata de T em S , r é um ramo de T , r' é um ramo de T' , r e r' são distintos e r é equivalente a um segmento inicial de r' .

Dado um ramo r de S , tal que cada fórmula de r é aplicável a no máximo uma regra, definiremos recursivamente $\eta_i(r)$. Temos que há uma seqüência de ramos r_0, \dots, r_n , tal que r_0 é um ramo de algum tableau inicial de S , para cada $i = 1, \dots, n$, r_i é uma extensão imediata de r_{i-1} , e r_n é r . $\eta_0(r)$ é o segmento inicial de r equivalente a r_0 e, para cada $i = 1, \dots, n$, $\eta_i(r)$ é o segmento de r tal que $\eta_0(r) + \dots + \eta_{i-1}(r) + \eta_i(r)$ é equivalente a r_i . Dizemos que μ é um nó composto de r se há $i \in \{0, \dots, n\}$ tal que $\mu = \eta_i(r)$. É evidente que r é a concatenação de seus nós compostos.

O conceito de *seqüência completa de tableaux*, definido abaixo, é essencial para provas de completude de um sistema de tableaux com respeito a alguma semântica dada. Em Buchsbaum [3], capítulo 5, apresentamos uma prova detalhada de completude do sistema $SC1^*$ com respeito à semântica de $C1^*$.

Seja $(T_i)_{i \in I}$ uma seqüência de tableaux, onde I é \mathbb{N} ou I é da forma $\{0, 1, \dots, n\}$, onde $n \in \mathbb{N}$, de modo que, para cada $i \in I$, se $i > 0$, então T_i é uma extensão imediata de T_{i-1} em S ; dizemos então que $(T_i)_{i \in I}$ é uma *seqüência de desenvolvimento de tableaux* em S .

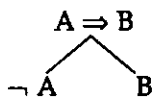
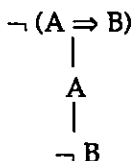
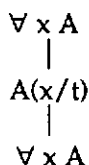
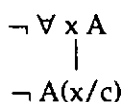
Dizemos que uma seqüência $(T_i)_{i \in I}$ de desenvolvimento de tableaux é *completa* se as condições (a) e (b) dadas abaixo forem satisfeitas :

- (a) Para cada $i \in I$, T_i possui um sucessor se, e somente se, T_i não é uma confutação nem possui um ramo exaurido aberto;
- (b) Se I é um conjunto infinito, então $(T_i)_{i \in I}$ tende para uma árvore-limite com um ramo infinito em que cada nó está marcado ou a sua fórmula é excluída.

Pode-se provar que, para qualquer tableau dado, existe uma seqüência de desenvolvimento completa cujo primeiro tableau é o próprio tableau dado (veja Buchsbaum [3], capítulo 5).

Exemplo 1 : o cálculo de predicados clássico

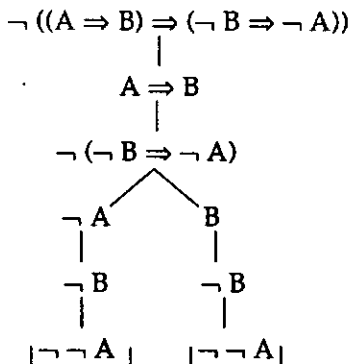
Seja L uma linguagem de primeira ordem cujos conectivos lógicos são \Rightarrow e \neg , e o único quantificador é \forall ; L' uma extensão de L obtida acrescentando-se uma infinidade de novas constantes; Γ a coleção das fórmulas fechadas de L ; I uma função que associa a cada fórmula A de Γ um tableau possuindo um único nó, não marcado, cuja fórmula é A ; F uma função definida por $F(r) =$ fechado sss r possui duas fórmulas contraditórias, onde r é um ramo em L' ; e finalmente R uma coleção de cinco regras dadas pelos diagramas abaixo :

Regra $A \Rightarrow B$ Regra $\neg(A \Rightarrow B)$ Regra $\neg\neg A$ Regra $\forall x A$ Regra $\neg\forall x A$ 

A expressão $A(x/t)$ na regra para fórmulas quantificadas universalmente representa a fórmula obtida de uma fórmula dada A substituindo-se todas as ocorrências livres de x em A por t , onde t é o primeiro termo de L' tal que $A(x/t)$ não figura no ramo em que a regra estiver sendo aplicada. A constante c na expressão $\neg A(x/c)$ na regra para negações de fórmulas quantificadas universalmente é a primeira constante de L' que não ocorre no ramo em que a regra estiver sendo aplicada.

A regra para implicações (regra $A \Rightarrow B$), por exemplo, é uma função que associa a cada fórmula de L' da forma $A \Rightarrow B$ e a cada ramo no qual esta fórmula figure uma coleção de dois tableaux, onde o primeiro tableau possui um único nó, não marcado, cuja fórmula é $\neg A$, e o segundo tableau também possui um único nó, não marcado, cuja fórmula é B .

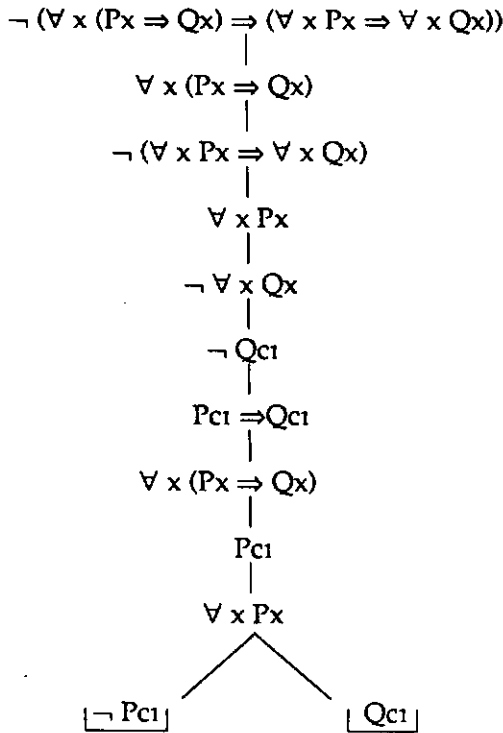
O sistema de tableaux $S = \langle L, L', \Gamma, I, F, R \rangle$, dado acima, é um sistema adequado para a lógica clássica. Por exemplo, se quisermos verificar se a fórmula $(A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$ é um teorema da lógica clássica, devemos observar se o tableau inicial para a sua negação converge para uma confutação:



Devemos proceder de um modo análogo para verificar se a fórmula

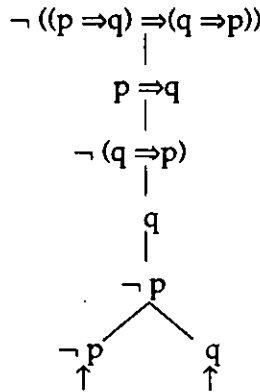
$$\forall x (Px \Rightarrow Qx) \Rightarrow (\forall x Px \Rightarrow \forall x Qx)$$

é um teorema da lógica clássica :

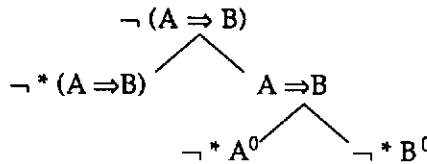


Observe que, no desenvolvimento deste segundo tableau, não escolhemos os nós na mesma ordem em que estes foram surgindo, mas sim na ordem mais adequada para obter o fechamento dos ramos mais rapidamente. Em geral, é melhor deixar por último os nós com fórmulas quantificadas universalmente. A mecanização deste processo depende de certos critérios heurísticos (veja Oppacher & Suen [11]).

Segundo a expansão mostrada abaixo, podemos concluir que a fórmula $(p \Rightarrow q) \Rightarrow (q \Rightarrow p)$, onde p e q são fórmulas atômicas, não é teorema da lógica clássica, já que obtemos um tableau exaurido a partir do tableau inicial que não é uma confutação, onde todos os ramos são abertos :



A estrutura das regras no sistema que vimos acima é bem simples, no sentido de que cada regra acima é uma função que fornece ou uma coleção com um único tableau contendo no máximo dois nós, ou uma coleção de dois tableaux contendo cada um apenas um nó. Isto não ocorre em todos os casos. Por exemplo, a regra do sistema SC1, elaborado para o cálculo paraconsistente C1, para fórmulas da forma $\neg (A \Rightarrow B)$, é dada no diagrama abaixo :



Esta regra é uma função que associa a cada fórmula da forma $\neg (A \Rightarrow B)$ uma coleção de dois tableaux, onde o primeiro tableau possui um único nó não marcado, cuja fórmula é $\neg * (A \Rightarrow B)$, e o segundo tableau possui três nós, não marcados, com um nó inicial e dois sucessores, sendo que a fórmula do nó inicial é $A \Rightarrow B$, e as fórmulas dos dois nós sucessores são $\neg * A^0$ e $\neg * B^0$.

Existem regras que geram coleções de tableaux nas quais pelo menos um de seus nós é previamente marcado. Isto ocorre por exemplo em um sistema de tableaux para o cálculo DL (veja Buchsbaum & Pequeno [4]).

Exemplo 2 : o cálculo paracompleto P1

Uma lógica é dita paracompleta se nela o princípio do terceiro excluído não vigora, ou seja, não se tem em geral que, dadas duas fórmulas contraditórias A e $\neg A$, uma delas é necessariamente verdadeira. A lógica intuicionista e diversas lógicas polivalentes são paracompletas neste sentido. Este tipo de lógica pode ser adequado por exemplo a situações como a que descrevemos a

seguir. Ao olharmos para uma rosa localizada a uma grande distância, pode acontecer de não estarmos seguros com respeito à percepção de sua cor, ou seja, não necessariamente pode ser observada uma cor vermelha ou outra cor distinta do vermelho. Em uma lógica que desse conta do estado subjetivo de nossa percepção, ao invés da condição objetiva da rosa observada; não se poderia afirmar que a rosa seria vermelha ou não vermelha. Em geral, uma lógica paracompleta pode ser concebida como a lógica subjacente de uma teoria incompleta no sentido forte, isto é, de uma teoria na qual uma proposição e a sua negação podem ser ambas falsas (veja da Costa & Marconi [8]).

Apresentamos abaixo os axiomas e a regra de inferência do cálculo P1 :

$$\begin{aligned}
 & A \Rightarrow (B \Rightarrow A); \\
 & (A \Rightarrow B) \Rightarrow (A \Rightarrow (B \Rightarrow C)) \Rightarrow (A \Rightarrow C); \\
 & \frac{A, A \Rightarrow B}{B}; \\
 & A \wedge B \Rightarrow A; \\
 & A \wedge B \Rightarrow B; \\
 & A \Rightarrow (B \Rightarrow A \wedge B); \\
 & A \Rightarrow A \vee B; \\
 & B \Rightarrow A \vee B; \\
 & (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow (A \wedge B \Rightarrow C); \\
 & ((A \Rightarrow B) \Rightarrow A) \Rightarrow A; \\
 & A \Rightarrow \neg \neg A; \\
 & A \Rightarrow (\neg A \Rightarrow B); \\
 & \neg (A \wedge \neg A); \\
 & A^* \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow \neg B) \Rightarrow \neg A); \\
 & A^* \wedge B^* \Rightarrow (A \Rightarrow B)^* \wedge (A \wedge B)^* \wedge (A \vee B)^*.
 \end{aligned}$$

A^* é abreviatura para $A \vee \neg A$ em P1.

Em P1 não se tem, em geral, que uma fórmula da forma $A \vee \neg A$ é teorema. A negação clássica em P1 pode ser simulada através da abreviatura

$$\sim A \equiv A \Rightarrow \neg A.$$

Temos que o conectivo \sim , da forma como este foi definido acima, apresenta todas as propriedades da negação clássica.

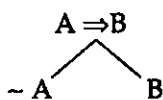
A seguir alguns teoremas de P1 que foram importantes para a construção do sistema SP1 :

$$\begin{aligned}
 & \vdash \neg A \Leftrightarrow \sim A \wedge A^*; \\
 & \vdash A^* \Rightarrow (\neg A)^*; \\
 & \vdash \sim \neg \neg A \Rightarrow \sim A; \\
 & \vdash \sim \neg \sim A \Leftrightarrow \sim A.
 \end{aligned}$$

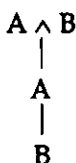
Apresentamos abaixo os componentes do sistema SP1. Usaremos o sinal # para representar qualquer um dos conectivos \Rightarrow , \wedge ou \vee .

Seja L uma linguagem proposicional cujos conectivos lógicos são \Rightarrow , \wedge , \vee e \neg ; Γ a coleção de todas as fórmulas de L ; I uma função que associa a cada fórmula A de L um tableau possuindo um único nó, não marcado, cuja fórmula é A ; F uma função definida por $F(r) = \text{fechado}$ sss r possui duas fórmulas das formas A e $\sim A$, ou uma fórmula da forma $\sim \neg (A \wedge \neg A)$, onde r é um ramo em L , e finalmente R uma coleção de regras dadas pelos diagramas abaixo :

Regra $A \Rightarrow B$



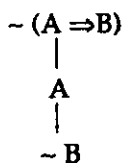
Regra $A \wedge B$



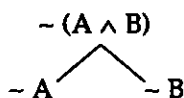
Regra $A \vee B$



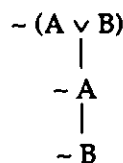
Regra $\sim (A \Rightarrow B)$



Regra $\sim (A \wedge B)$



Regra $\sim (A \vee B)$



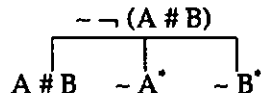
Regra $\sim \sim A$



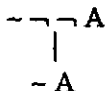
Regra $\neg A$



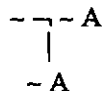
Regra $\sim \neg (A \# B)$



Regra $\sim \neg \neg A$



Regra $\sim \neg \sim A$

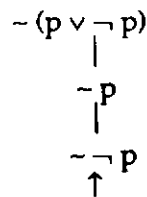


O sistema de tableaux SP1 = $\langle L, L, \Gamma, I, F, R \rangle$, dado acima, é um sistema adequado para o cálculo P1, no sentido de que, dada uma fórmula A , A é teorema de P1 se, e somente se, há uma confutação para $\sim A$. Note que as linguagens inicial e de trabalho de SP1 são idênticas; tal fato é muito freqüente nos cálculos proposicionais.

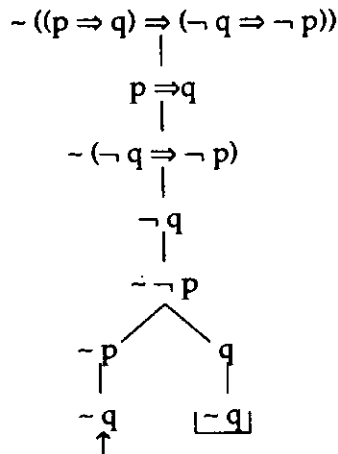
As fórmulas excluídas do sistema são de uma das formas p , $\sim p$ ou $\sim \neg p$, onde p é uma fórmula atômica.

As regras específicas para fórmulas de uma das formas $\sim \sim A$ ou $\sim \neg \sim A$ não são essenciais para a completude do sistema em relação ao cálculo P1.

Podemos observar que a negação clássica da fórmula $p \vee \neg p$, onde p é uma letra sentencial, não converge para uma confutação, daí temos que $p \vee \neg p$ não é teorema de P1 (o que verifica a paracompletude do cálculo) :



Temos também, conforme mostramos abaixo, que a fórmula $(p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$ (*modus tollens*) não é teorema em P1 :

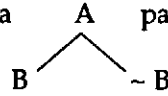


Heurísticas para a confecção de regras

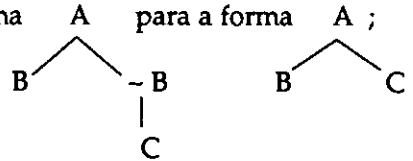
Damos abaixo algumas heurísticas que revelaram-se úteis na construção das regras de sistemas de tableaux :

- No caso da lógica considerada ser um desvio da lógica clássica, devemos procurar relações entre os conectivos com significados alterados e os demais conectivos;
- Devemos procurar eliminar circularidades no desenvolvimento das

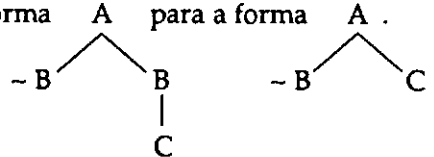
- árvores de prova, marcando e desenvolvendo previamente certos nós;
- Eliminar nós com fórmulas repetidas;
- Eliminar nós com fórmulas redundantes, observando as interações dos ramos em que estes nós figuram com nós com as mesmas fórmulas negadas classicamente;
- Transformar ramificações da forma



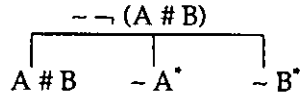
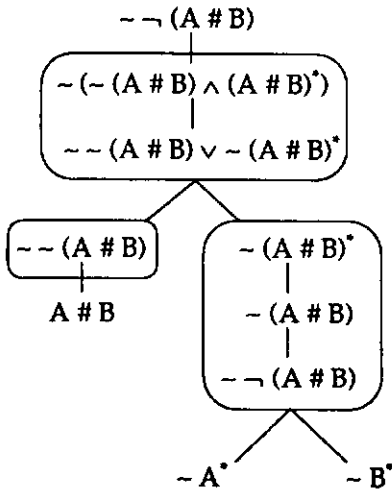
- Transformar ramificações da forma



- Transformar ramificações da forma



Não dispomos de espaço no presente artigo para dar explicações ou exemplos exaustivos do que afirmamos acima, mas daremos abaixo um exemplo, descrito sucintamente, de como elaboramos a regra para fórmulas do tipo $\neg\neg(A \# B)$ para SP1. A primeira árvore representa o processo de confecção da regra, e a segunda representa a regra em sua forma final. As fórmulas circundadas representam nós eliminados da árvore da regra.



As provas de correção e completude

Nesta seção apresentamos um esboço, valendo para todos os sistemas vistos aqui e em Buchsbaum & Pequeno [4], das provas de correção e completude em relação aos cálculos considerados. Este esboço segue, em linhas gerais, a prova apresentada em Bell & Machover [1] para a lógica clássica.

Dizemos que um conjunto ϕ de fórmulas é *trivial* com respeito a um dado cálculo se qualquer fórmula (da linguagem do cálculo) puder ser deduzida a partir de ϕ .

A letra grega ϕ e a letra A representam respectivamente, nos enunciados dos teoremas abaixo, um conjunto finito de fórmulas e uma fórmula do domínio do sistema considerado em cada caso. O sinal \sim usado abaixo representa a negação clássica (o mesmo que \neg no caso clássico, ou a sua simulação nos casos em que a negação é alterada).

Nas provas abaixo, consideraremos, para cada sistema de tableaux, um sistema equivalente no qual os nós são conjuntos finitos de fórmulas.

Lema da Correção : Se há uma confutação para ϕ , então ϕ é trivial.

Prova :

É paralela à prova do caso clássico, levando-se em conta a função de inicialização, o critério de fechamento e as regras de cada sistema.

Teorema da Correção : Se há uma confutação para ϕ , $\sim A$, então há uma dedução de A a partir de ϕ .

Prova :

Supõe-se que a negação clássica (\sim) deverá ser definível no cálculo considerado; no mais aplica-se a esta prova o mesmo comentário da prova anterior.

Lema da Eliminação : Se existem confutações para ϕ , A e para ϕ , $\sim A$, então existe uma confutação para ϕ .

Prova :

A prova é feita por indução sobre o grau de A . Caso as regras sejam conservativas (no sentido de produzirem decomposições equivalentes às fórmulas), a prova é paralela ao caso clássico; caso isso não se verifique, ela pode ser mais complexa.

Teorema da Eliminação : Se há uma confutação para ϕ obtida usando regras do terceiro excluído, então há uma confutação para ϕ obtida sem o uso de tais regras.

Prova :

É análoga à prova do caso clássico.

Teorema da Completude : Se há uma dedução de A a partir de ϕ , então há uma confutação para $\phi, \sim A$.

Prova :

Nesta prova, a função de inicialização, o critério de fechamento e as regras são relevantes, particularmente para mostrar que, para cada axioma A , há uma confutação para $\phi, \sim A$; no mais a prova assemelha-se ao caso clássico.

Referências

- [1] Bell, J. L. & Machover, M., *A Course in Mathematical Logic*, North Holland Publishing Company, 1977.
- [2] Beth, E. W., « Semantic Entailment and Formal Derivability », *Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen*, 18, pgs. 309-342, Amsterdam, 1955.
- [3] Buchsbaum, Arthur, *Um Método Automático de Prova para a Lógica Paraconsistente*, Dissertação de Mestrado, Pontifícia Universidade Católica, Rio de Janeiro, 1988.
- [4] Buchsbaum, Arthur & Pequeno, Tarcisio, « Raciocínio Automático em Lógicas Paraconsistentes e/ou Paracompletas », *Proceedings do 6º Simpósio Brasileiro de Inteligência Artificial*, pgs. 1-15, 1989.
- [5] Buchsbaum, Arthur & Pequeno, Tarcisio, « A Reasoning Method for a Paraconsistent Logic », a ser publicado no *Journal of Non-Classical Logic*, volume 7, número 1/2, 1990.
- [6] Chang, Chin-Liang & Lee, Richard Char-Tung, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.
- [7] da Costa, Newton C. A., « On the Theory of Inconsistent Formal Systems », *Notre Dame Journal of Formal Logic* 15, pgs. 497-510, 1974.
- [8] da Costa, Newton C. A. & Marconi, Diego, « A Note on Paracomplete Logic », *Rendiconti dell'Accademia Nazionale dei Lincei*, pgs. 504-509, 1986.
- [9] Fitting, M. C., « Tableau Methods of Proof for Modal Logics », *Notre Dame Journal of Formal Logic* 13, pgs. 237-247, 1972.
- [10] Hintikka, J., « Form and Content in Quantification Theory », *Acta Philosophica Fennica* 8, Helsinki, 1955.
- [11] Oppacher, F. & Suen, E., « HARP : A Tableau-Based Theorem Prover », *Journal of Automated Reasoning* 4, pgs. 69-100, 1988.
- [12] Pequeno, Tarcisio, « A Logic for Inconsistent Nonmonotonic Reasoning », *Technical Report 90/6*, Department of Computing, Imperial College, London, 1990.
- [13] Smullyan, Raymond M., « A Unifying Principle in Quantification Theory », *Proceedings of the National Academy of Sciences*, 49, pgs. 828-832, 1963.