Extração automática de axiomatizações em termos de sistemas de tableaux bi-rotulados para lógicas multi-valoradas finitárias

Dalmo Mendonça
Bolsista PROPESQ - UFRN
dalmo_3@yahoo.com.br

João Marcos **Marcel Oliveira**
DIMAp/CCET, UFRN
jmarcos@dimap.ufrn.br marcel@dimap.ufrn.br

Resumo A Lógica Clássica admite dois valores de verdade: verdadeiro e falso. Lógicas multi-valoradas ou n-valoradas possuem um conjunto com n valores de verdade, com níveis de verdade entre o falso e o verdadeiro. É de se perceber, contudo, que há uma certa bivalência sobre esses níveis — há valores que são *designados* e outros que não o são. Sabe-se (cf. [3]) que esse caráter bivalente permite uma representação de lógicas multi-valoradas em termos de semânticas bivalentes, com valores lógicos 'T' e 'F'.

Nesse trabalho iremos mostrar como 'traduzir' cláusulas obtidas pela redução apresentada em [1] para axiomas de um sistema de tableaux com rótulos T e F. Os algoritmos de tradução serão implementados na linguagem de programação *ML*, com o objetivo de criar de maneira automática arquivos de teorias a serem processados pelo software *Isabelle* (cf. [4]), através do qual será possível verificar futuramente meta-resultados e teoremas sobre as lógicas estudadas. O estudo de diferentes lógicas multi-valoradas sobre um mesmo ponto de vista — sistemas de tableaux bi-rotulados — facilitará comparações entre propriedades dessas lógicas e também a definição de suas semelhanças e particularidades.
Palavras-chave lógica multi-valorada, bi-rotulação, axiomatização, tableaux, automatização.

Reduzir Lógicas?

Na década de 1970, o lógico polonês Roman Suszko afirmou: *"Não há senão dois valores lógicos: verdadeiro e falso"*. Ele defendia que em lógicas multi-valoradas os valores de verdade, apesar de terem diferentes valores algébricos, pertenciam exclusivamente a duas 'classes': os verdadeiros e os falsos. De fato, a definição dos *valores designados* de uma lógica já atribui a eles o papel de "verdadeiros". Suszko então mostrou, em [3], ser possível representar a lógica trivalorada Ł$_3$ de Łukasiewicz em termos de uma semântica bivalente.

Caleiro et al., em [1], apresentam uma maneira construtiva de reduzir uma ampla classe de lógicas multi-valoradas finitárias para semânticas com apenas dois valores de verdade. A pré-condição para o processo de redução funcionar é que essas lógicas sejam suficientemente expressivas, isto é, que seja possível definir fórmulas unárias capazes de distinguir cada valor de verdade (ver também o procedimento definido em [2], apresentado neste mesmo evento). Os resultados redutivos obtidos são representados como cláusulas escritas na linguagem da Lógica Clássica de Primeira Ordem, cláusulas estas que impõem restrições sobre o conjunto de funções de valoração com dois valores de verdade de tal forma que este conjunto forneça uma semântica correta e completa para a lógica que é objeto da redução. Em [1] mostra-se ainda que estas mesmas cláusulas podem definir sistemas de tableaux bi-rotulados, que apresentam funcionamento similar ao sistema correspondente para o caso da Lógica Clássica.

Resultados Redutivos

Vamos tomar como exemplo a lógica Ł$_3$ $= \langle \{0, \frac{1}{2}, 1\}, \{\neg, \rightarrow, \wedge, \vee\}, \{1\} \rangle$. Note que apenas o valor 1 é designado, o que, por questão de simplicidade, vamos denotar por $T(1)$. Para os outros valores, denotamos $F(0)$ e $F(\frac{1}{2})$. É de imediato perceber que na nossa semântica bivalente o valor 1 está "separado" do 0 e do $\frac{1}{2}$, mas como diferenciar os dois últimos? A resposta é encontrar os conectivos "separadores".

Neste caso, o conectivo primitivo $\neg$, definido pela tabela a seguir, já resolve o problema da separação dos valores não-designados:

	0	$\frac{1}{2}$	1
$\neg$	1	$\frac{1}{2}$	0

Conclui-se que $T(\neg 0)$, $F(\neg \frac{1}{2})$ e $F(\neg 1)$. Podemos então distinguir os valores de verdade da seguinte maneira:

$$\begin{array}{ll} x = 0 \text{ sse } F(x) \text{ e } T(\neg x); & \\ x = \frac{1}{2} \text{ sse } F(x) \text{ e } F(\neg x); & \text{(I)} \\ x = 1 \text{ sse } T(x). & \end{array}$$

Agora considere a tabela de verdade da implicação, definida por $(\alpha \to \beta) := Min(1, 1-\alpha+\beta)$, onde Min é a função Mínimo:

$\to$	0	$\frac{1}{2}$	1
0	1	1	1
$\frac{1}{2}$	$\frac{1}{2}$	1	1
1	0	$\frac{1}{2}$	1

Para obter uma semântica bivalente correta e completa, é necessário definir novas regras para os conectivos da linguagem. A tabela acima nos permite afirmar que $(\alpha \to \beta) = 0$ sse $\alpha = 1$ e $\beta = 0$. Utilizando as expressões de (I), é equivalente dizer que

$F(\alpha \to \beta)$ e $T(\neg(\alpha \to \beta))$ sse $T(\alpha)$, $F(\beta)$ e $T(\neg(\beta))$.

Particularmente, $T(\neg(\alpha \to \beta))$ só é o caso quando $T(\alpha)$, $F(\beta)$ e $T(\neg(\beta))$ também valerem. Isso é expresso na Lógica Clássica por

$$T(\neg(\alpha \to \beta)) \to T(\alpha) \wedge F(\beta) \wedge T(\neg(\beta)).$$

A partir desta cláusula, é possível definir a seguinte regra de tableau:

$$\frac{T(\neg(\alpha \to \beta))}{T(\alpha), F(\beta), T(\neg(\beta))}$$

Regras semelhantes podem ser obtidas se repetirmos este procedimento para o restante da tabela da implicação, bem como para os outros conectivos da lógica em questão. De acordo com [1], um conjunto correto e completo de regras de tableaux pode então ser obtido se adicionarmos a estas regras a seguinte *regra de bifurcação*:

$$\frac{}{T(\alpha) | F(\alpha)}$$

ML e Isabelle

Utilizamos a linguagem de programação funcional *ML* para automatizar o processo de extração dos axiomas. O *ML* nos fornece, entre outras vantagens, uma sintaxe elegante, de fácil compreensão e a garantia da correção do programa em tempo de compilação.

O *Isabelle* é um ambiente de demonstrações genérico, construído também em *ML*, no qual é possível rapidamente desenvolver teorias contendo regras e axiomas em variados sistemas dedutivos, bem como construir funções e demonstrar teoremas sobre essas teorias. Eis uma ilustração de como seria a regra obtida na última seção reescrita no *Isabelle*:

```
TNegImp
    "[$H, T:A, F:B, T:~B, $E]
                ==> [$H, T:~(A-->B), $E]"
```

Na expressão acima, fórmulas do tipo `T:X` e `F:X` representam $T(\mathrm{x})$ e $F(\mathrm{x})$ respectivamente; `$H` e `$E` são sequências de fórmulas; e cada sequência entre colchetes representa um ramo de tableaux. Na estratégia de demonstração do *Isabelle*, a aplicação de uma regra indica que é possível alcançar o objetivo (conteúdo à direita da meta-implicação, representada por `==>`), desde que seja possível demonstrar a hipótese (à esquerda da meta-implicação), a qual passa a constituir o novo objetivo (ou, em geral, a nova coleção de sub-objetivos).

O nosso programa recebe como entrada uma lógica multi-valorada finitária (seus valores de verdade e conectivos), realiza a extração dos axiomas, a tradução para regras de tableau e então gera um arquivo de texto contendo a teoria a ser utilizada com o *Isabelle*.

Referências

[1] C. Caleiro, W. A. Carnielli, M. E. Coniglio, e J. Marcos. Two's company: "The humbug of many logical values". Em *J.-Y. Béziau, editor, Logica Universalis*, pp 169–189. Birkhäuser Verlag, 2005. Cópia em http://wslc.math.ist.utl.pt/ftp/pub/CaleiroC/05-CCCM-dyadic.pdf

[2] T. Lincoln, J. Marcos, e M. Oliveira. Cálculo automático do grau de expressividade de álgebras finitárias, e aplicações. Em *Encontro Regional de Matemática Aplicada e Computacional* (ERMAC 2007). Recife:2007.

[3] R. Suszko, Remarks on Łukasiewicz's three-valued logic, Bulletin of the Section of Logic **4** (1975), pp 87–90.

[4] Site oficial do software Isabelle (em inglês): http://www.cl.cam.ac.uk/research/hvg/Isabelle/