Cálculo automático do grau de expressividade de álgebras finitárias, e aplicações

Talis Lincoln
Aluno IC, UFRN
talislincoln@engcomp.ufrn.br

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

Resumo Por *álgebra finitária* entenderemos, no presente artigo, álgebras recursivamente construídas sobre um conjunto finito de elementos por meio de um conjunto finito de operações de aridade finita. Se entendemos tais álgebras como realizações semânticas de tipos abstratos de dados, como é usual em computação, a álgebra de termos correspondente fornece um modelo concreto da declaração abstrata do tipo correspondente, e representa todas as operações definíveis ou exprimíveis a partir da álgebra finitária tomada de partida. Note-se que, para cada aridade pré-determinada, o conjunto de operações *distintas* que se pode expressar numa álgebra finitária é também necessariamente finito: com efeito, do ponto de vista extensional, o número máximo de operadores n-ários distintos que podem ser definidos sobre um universo de m elementos é $n^{(n^m)}$. Implementaremos a seguir um procedimento que permite calcular, de forma automática, o conjunto de operadores m-ários distintos definíveis a partir de uma álgebra cujo universo possui n elementos. Algumas aplicações interessantes deste procedimento ao estudo da Lógica (semântica e teoria da demonstração) serão mencionadas.

Palavras-chave álgebra universal, lógica, expressividade linguística e semântica.

Introdução e objetivos

Diremos que uma álgebra $\mathcal{A} = \langle A, \Sigma\rangle$ é *finitária* caso $\#(A)$, a cardinalidade de seu universo A, seja finita e caso sua assinatura Σ consista de um conjunto finito de operações com aridades finitas. Neste caso podemos dizer também que $\mathcal{A}$ é uma álgebra *n-valorada*. Uma situação bastante prática na qual álgebras n-valoradas têm lugar em computação é na especificação formal em BNF de linguagens definidas por gramáticas livres de contexto por meio dos construtores em Σ. Dado um conjunto enumerável de variáveis, At, a álgebra de termos livremente gerada por Σ sobre At representa todas as expressões definíveis a partir da linguagem subjacente à álgebra $\mathcal{A}$. Do ponto de vista estrutural, toda expressão deste tipo pode ser entendida como descrevendo uma árvore cujos nós interiores representam os construtores n-ários da assinatura Σ, para $n > 0$, e do ponto de vista semântico uma interpretação desta expressão consiste na ligação (*binding*) de valores do universo A às variáveis de At que comparecem nas folhas da árvore em questão (as folhas que não contêm variáveis contêm necessariamente construtores 0-ários de Σ). Assim, a linguagem de uma álgebra n-valorada $\mathcal{A}$ descreve um tipo abstrato de dado, e cada interpretação das expressões desta linguagem através de $\mathcal{A}$ descreve um modelo concreto deste mesmo tipo de dado.

Quais operações concretas sobre o universo A são de fato exprimíveis a partir da assinatura Σ de uma álgebra n-valorada $\langle A, \Sigma\rangle$? Será possível definir por exemplo operações 0-árias correspondendo a cada elemento de A? Operações triviais tais como a identidade (a função 1-ária $\lambda x.x$) ou as projeções (as funções k-árias $\lambda x_1 \ldots \lambda x_i \ldots \lambda x_k.x_i$) são sempre definíveis, pois não necessitam de qualquer auxílio linguístico. Para cada aridade m, o número máximo de operações extensionalmente distintas que poderiam teoricamente ser definidas é $n^{(n^m)}$. Quantas destas operações podem de fato ser descritas por meio de expressões envolvendo símbolos de Σ?

Seguramente, a resposta para tal questão depende do grau de expressividade da própria assinatura Σ. Por exemplo, se tomamos a álgebra usual correspondente à aritmética sobre os números naturais, podemos usar a função sucessor para definir a adição, e usar a adição para definir a multiplicação, mas não é possível definir a adição exclusivamente a partir da multiplicação. No caso finitário, considere o seguinte exemplo simples. Considere o universo $\mathbb{N}_m = \{i \in \mathbb{N} : 0 \leq i < m\}$ e uma única operação 1-ária. Quantas operações unárias serão definíveis, em geral? O princípio da casa do pombo, da combinatória finita, nos garante que o número total de tais operações não é superior a m: mais precisamente, a operação s^m_j deve ser extensionalmente idêntica a alguma entre as operações $s^0_j, \ldots, s^{m-1}_j$, onde s^0_j denota a função identidade e $s^{k+1}_j = s \circ s^k_j$.

A caracterização do poder expressivo de uma álgebra nos permite determinar, assim, quais operações sobre o seu universo podem ser descritas pela linguagem que temos disponível. O obje-

tivo principal do presente trabalho será o de implementar um procedimento que permita calcular, para uma dada álgebra n-valorada $\mathcal{A}$ com assinatura Σ, e uma dada aridade m, o conjunto de todas as operações m-árias n-valoradas extensionalmente distintas que podem ser definidas com o auxílio dos construtores disponibilizados por Σ.

Ilustração do procedimento

Tomemos um exemplo da Lógica. Considere uma álgebra 2-valorada sobre o universo dos valores de verdade clássicos, $\{0, 1\}$, e considere as seguintes operações binárias bem conhecidas, $\{\leftrightarrow, \vee\}$. Podemos naturalmente então nos perguntar: **quais conectivos binários podem ser definidos a partir destes**? Se em geral sabemos que há $16 (= 2^{(2^2)})$ operações binárias distintas sobre dois valores de verdade, quantas destas operações poderão ser definidas usando apenas a bi-implicação e a disjunção? O nosso procedimento fará os seguintes passos até descobrir que apenas 8 destas operações são exprimíveis a partir deste fragmento da linguagem clássicas.

O algoritmo começa a partir das fórmulas de menor grau de complexidade, isto é, menor número de construtores, e prossegue pela adição paulatina de novos construtores a expressões anteriormente formadas. Há duas operações de projeção óbvias que prescindem de construtores: (1) $\lambda p_1\, p_2.p_1$ e (2) $\lambda p_1\, p_2.p_2$, e nada pode ser menos complexo do que isto. Como não há outras operações com complexidade zero, o algoritmo toma um símbolo de função da assinatura, digamos $\leftrightarrow$, e compõe a operação 2-ária correspondente com as operações que já foram definidas. O resultado são as operações $\lambda p_1\, p_2.p_1 \leftrightarrow p_1$, $\lambda p_1\, p_2.p_1 \leftrightarrow p_2$, $\lambda p_1\, p_2.p_2 \leftrightarrow p_1$, $\lambda p_1\, p_2.p_2 \leftrightarrow p_2$. No entanto, a terceira e a quarta expressões definem operações extensionalmente idênticas à segunda e à primeira, respectivamente, e por isso são descartadas. Guardamos assim apenas (3) $\lambda p_1\, p_2.p_1 \leftrightarrow p_1$ e (4) $\lambda p_1\, p_2.p_1 \leftrightarrow p_2$. Na próxima etapa fazemos o mesmo para o outro símbolo da assinatura, $\vee$, que também corresponde a uma operação 2-ária. Mais uma vez, quatro novas expressões são definíveis, $\lambda p_1\, p_2.p_1 \vee p_1$, $\lambda p_1\, p_2.p_1 \vee p_2$, $\lambda p_1\, p_2.p_2 \vee p_1$, $\lambda p_1\, p_2.p_2 \vee p_2$. A primeira e a quarta expressões geram operações que coincidem extensionalmente com as operações de projeção que já foram armazenadas, e a segunda e a terceira expressões geram operações extensionalmente idênticas, portanto basta guardarmos (5) $\lambda p_1\, p_2.p_1 \vee p_2$. Não há agora novos símbolos na assinatura, então a idéia é repetir a escolha dos símbolos anteriores, e compô-los com as operações já guardadas nos passos precedentes, procurando por novas combinações sintáticas. Muitas são as combinações resultantes, neste estágio, mas as únicas que geram operações extensionalmente distintas daquelas já definidas são (6) $\lambda p_1\, p_2.p_1 \leftrightarrow (p_1 \leftrightarrow p_2)$ e (7) $\lambda p_1\, p_2.p_1 \leftrightarrow (p_1 \leftrightarrow p_2)$. No próximo estágio, compondo mais uma vez as operações correspondentes aos símbolos primitivos da assinatura com as operações já guardadas, obtemos somente uma 'nova' operação, a saber, (8) $\lambda p_1\, p_2.p_1 \leftrightarrow (p_2 \leftrightarrow (p_1 \vee p_2))$. O nosso procedimento converge e pára quando já não é possível definir novas operações, do ponto de vista extensional, pela composição dos operadores primitivos com as operações já calculadas e guardadas.

Aplicações à Lógica

Já vimos que o procedimento acima tem um forte apelo semântico, indicando aqueles objetos e operações que se podem exprimir a partir de um conjunto de operações primitivas. Uma outra aplicação interessante e particular do mesmo procedimento, contudo, se dá na Teoria da Demonstração, em Lógica. Em [1] exibe-se um algoritmo capaz de produzir axiomatizações corretas e completas em termos de tableaux para lógicas multi-valoradas finitárias (um caso particular de álgebras finitárias). A condição necessária e suficiente para a aplicação deste algoritmo em cada caso, contudo, é a definibilidade de uma certa classe de operadores 1-ários, isto é, o algoritmo está condicionado à possibilidade de se exprimir um certo conjunto de funções a partir da assinatura original das lógicas dadas, e estas funções são usadas na axiomatização em questão. O pré-requisito para a aplicação deste algoritmo, assim, pode ser verificado facilmente através da saída do nosso presente procedimento, para o caso 1-ário, saída esta que também devolve as definições explícitas dos operadores dos quais se fará uso na citada axiomatização. Uma implementação deste algoritmo de extração de axiomas será apresentada em [2], no presente evento.

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. Preprint disponível em http://wslc.math.ist.utl.pt/ftp/pub/CaleiroC/05-CCCM-dyadic.pdf

[2] Dalmo Mendonça, J. Marcos, e M. Oliveira. Extração automática de axiomatizações em termos de sistemas de tableaux bi-rotulados para lógicas multi-valoradas finitárias. Em *Encontro Regional de Matemática Aplicada e Computacional* (ERMAC 2007). Recife:2007.