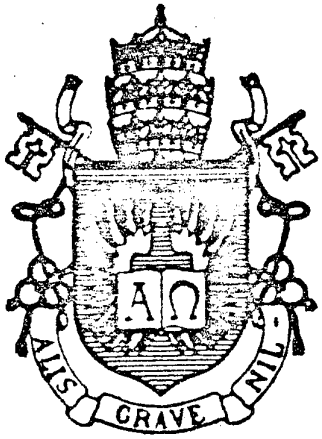

PUC

Série: Monografias em Ciência da Computação,
No.15/89

UM METODO AUTOMATICO DE PROVA PARA A LOGICA POLISORTIDA DEONTICA DE AÇÕES

Paulo Sérgio C. Alencar
Arthur R. V. Buchsbaum

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP-22453
RIO DE JANEIRO - BRASIL

Responsável por publicações:

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC RIO, Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453 - Rio de Janeiro, RJ
BRASIL

Tel.: (021) 529-9386 TELEX: 31078 FAX: (021) 274-4546
BITNET: userrtlc@lncc.bitnet

PUC RJ - Departamento de Informática

Série: Monografias em Ciência da Computação, Nº 15/89
Editor: Paulo A. S. Veloso Julho, 1989

UM MÉTODO AUTOMÁTICO DE PROVA PARA A LÓGICA POLISORTIDA DEÔNTICA DE AÇÕES

Paulo Sérgio C. Alencar
Arthur R. V. Buchsbaum

Trabalho parcialmente financiado pela FINEP

Um Método Automático de Prova para a Lógica Polisortida Deôntica de Ações

by
Paulo Sérgio C. de Alencar
Instituto de Ciências Exatas
Universidade de Brasília
Campus Universitário - Asa Norte
Brasília - DF 70090, Brazil and
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225
Rio de Janeiro 22453, Brazil
and
Arthur Ronald de Vallauris Buchsbaum
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225
Rio de Janeiro 22453, Brazil

Março 1989

Abstract

Neste artigo nós apresentamos um método de prova automática para a lógica polisortida deôntica com ações arbitrárias (constantes e não constantes) e domínio constante.

1 Introdução

Neste trabalho nós apresentamos um método de prova automática para uma lógica polisortida deôntica com ações arbitrárias (constantes e não constantes) e domínio constante, i. e., para a qual a fórmula de Barcan é válida. Com isso, os conjuntos de indivíduos nos diferentes mundos possíveis são idênticos.

O objetivo do presente trabalho é a obtenção de um método de prova automática para a lógica em questão que viabilize o raciocínio dedutivo sobre o processo de mudança tanto da estrutura quanto da funcionalidade de sistemas de software. Um tal sistema lógico com modalidades para ações e componentes deônticos fornece uma abordagem apropriada para lidar com estes aspectos do estudo de sistemas evolutivos de software.

Este artigo está estruturado como segue. Primeiramente, na seção 2 apresentamos por completeza uma vista geral do sistema lógico que queremos considerar. Na seção 3 um método de prova automática baseado em cálculo de tableaux é apresentado. Também nesta seção apresentamos um algoritmo de unificação de índices para esta lógica e fornecemos alguns exemplos. Finalmente, na seção 4 apresentamos algumas conclusões.

2 Descrição do Sistema Lógico LDA

Nesta seção apresentamos mais detalhadamente uma versão da lógica polisortida de ações deôntica que denominamos LDA. Esta lógica tem dois aspectos: o sintático e o semântico. O aspecto sintático refere-se às fórmulas bem formadas admitidas pela gramática de uma linguagem formal, bem como à questões de teoria da prova. Sob este aspecto, uma teoria desta lógica consiste de um alfabeto, de uma linguagem, de um conjunto de axiomas e de um conjunto de regras de inferência. A linguagem consiste das fórmulas bem formadas da teoria lógica. Os axiomas são um subconjunto designado de fórmulas bem formadas. Os axiomas e regras de inferência são usados para derivar os teoremas da teoria. Posteriormente trataremos do aspecto semântico da lógica, que trata dos significados associados às fórmulas bem formadas e aos símbolos que estas fórmulas contém.

Definição Um alfabeto de uma teoria desta lógica consiste de dez classes de símbolos:

(a) Uma coleção finita de sortes denotados pela letra grega τ (possivelmente com índices subscritos) dada por

$$T = Act \cup Agt \cup T(Act) \cup \{\tau_1, \tau_2, \ldots\}$$

onde Act denota o sorte de ações , Agt representa o sorte de agentes e $T(Act)$ denota o sorte não vazio de sequências finitas de ações . Os elementos do sorte $T(Act)$ são denotados pela letra grega σ, possivelmente com índices subscritos, enquanto que os elementos do sorte Agt são denotados pelas letras maiúsculas A, B e C, possivelmente com índices subscritos.

(b) Símbolos constantes:

1. Constante lógica $\hat{n}$.
2. Constantes extra-lógicas: para cada sorte $\tau \in T$ existe um conjunto (possivelmente vazio) de símbolos constantes cada um dos quais é dito ser de sorte τ. Estas constantes são denotadas pelas letras a, b, e c, possivelmente com índices subscritos.

(c) Nomes de ações : para cada $n > 0$ e cada n-upla $< \tau_1, \ldots, \tau_n >$ tal que $\tau_i \in T$, $i = 1, \ldots, n$, existe um conjunto (possivelmente vazio) de nomes de ações de aridade n pertencentes ao sorte *Act*, cada um dos quais é dito ser do tipo $< \tau_1, \ldots, \tau_n >$. Os nomes de ações são denotados pelas letras gregas α, β, γ, e δ, possivelmente com índices subscritos.

(d) Símbolos de predicados:

1. Lógicos: *per* do tipo *Act*, *obl* do tipo *Act*, *pref* do tipo *Act* e *obls* do tipo *Act*.
2. Extra-lógicos: para cada $n > 0$ e cada n-upla $< \tau_1, \ldots, \tau_n >$ tal que $\tau_i \in T$, $i = 1, \ldots, n$, existe um conjunto (possivelmente vazio) de símbolos de predicados de aridade n cada um dos quais é dito ser do tipo $< \tau_1, \ldots, \tau_n >$. Estes símbolos de predicados são denotados pelas letras p, q e r, possivelmente com índices subscritos.

(e) Símbolos funcionais:

1. Lógicos:
   $<>$ do tipo $< Act, T(Act) >$
   , do tipo $< T(Act), Act, T(Act) >$
2. Extra-lógicos: para cada $n > 0$ e cada $n+1$-upla $< \tau_1, \ldots, \tau_{n+1} >$ tal que $\tau_i \in T$, $i = 1, \ldots, n$, existe um conjunto (possivelmente vazio) de símbolos funcionais de aridade n cada um dos quais é dito ser do tipo $< \tau_1, \ldots, \tau_{n+1} >$. Denotamos estes símbolos funcionais pelas letras f, g e h, possivelmente com índices subscritos.

(f) Símbolos de igualdade: para certos sortes $\tau \in T$ (possivelmente todos) existe um símbolo predicado especial $=_\tau$ de sorte $< \tau, \tau >$ que representa a igualdade entre objetos de sorte τ.

(g) Variáveis: o conjunto infinito usual de variáveis distintas para cada sorte τ. As variáveis são denotadas pelas letras x, y, z, u, v e w, possivelmente com índices subscritos.

(h) Quantificadores: para cada sorte $\tau \in T$ temos um quantificador universal básico $\forall_\tau$ e um quantificador existencial $\exists_\tau$ definido em têrmos do quantificador universal.

(i) Operadores lógicos: $\neg$, $\rightarrow$, basicos e $\wedge$, $\vee$, $[\_,\_]$, $[[\_,\_]]$ definidos.

(j) Pontuação : "(", ")" e ",".

Passamos agora a definir a liguagem da teoria dada por um alfabeto.

Definição Um têrmo é definido indutivamente como segue:

(a) para cada sorte $\tau \in T$ uma variável ou uma constante de sorte τ é um têrmo.

(b) se $t_1, \ldots, t_n$ são têrmos de sortes $\tau_1, \ldots, \tau_n$, respectivamente e $\tau_i \in T$, $i = 1, \ldots, n$, e f é um símbolo funcional do tipo $< \tau_1, \ldots, \tau_{n+1} >$ então $f(t_1, \ldots, t_n)$ é um têrmo de sorte s_{n+1}.

(c) se $t_1, \ldots, t_n$ são têrmos de sortes $\tau_1, \ldots, \tau_n$, respectivamente e $\tau_i \in T$, $i = 1, \ldots, n$, e a é um simbolo de ação do tipo $< s_1, \ldots, s_n >$ então $a(t_1, \ldots, t_n)$ é um têrmo de sorte Act.

(d) nada mais é um têrmo.

Definição Uma fórmula atômica (ou um átomo) é definida indutivamente como segue:

(a) se $t_1, \ldots, t_n$ são têrmos de sortes denotados por $\tau_1, \ldots, \tau_n$ respectivamente, $\tau_i \in T$, $i = 1, \ldots, n$ e p é um simbolo de predicado de tipo $< \tau_1, \ldots, \tau_n >$ então $p(t_1, \ldots, t_n)$ é uma fórmula atômica.

(b) para cada sorte $\tau \in S$, dados dois termos de s, a saber, t_1 e t_2, $t_1 =_\tau t_2$ é uma fórmula atômica, se $=_\tau$ está no alfabeto.

(c) nada mais é uma fórmula atômica.

Definição Uma fórmula (bem-formada) é denotada pelas letras gregas ϕ, χ e ψ, possivelmente com índices subscritos, e é definida indutivamente como segue:

(a) Uma fórmula atômica é uma fórmula.

(b) Se ϕ é uma fórmula então $\neg\phi$ também é uma fórmula.

(c) Se ϕ e ψ são fórmulas então $(\phi \vee \psi), (\phi \wedge \psi), (\phi \rightarrow \psi), (\phi \leftrightarrow \psi)$ também são fórmulas.

(d) Se α é uma fórmula de sorte *Act*, A é um têrmo de sorte *Agt* e ϕ é uma fórmula, então $[A, \alpha]\phi$ é uma fórmula.

(e) Se σ é um têrmo de sorte $T(Act)$, A é um têrmo de sorte *Agt* e ϕ é uma fórmula, então $[[A, \sigma]]\phi$ é uma fórmula.

(f) Para cada sorte $\tau \in T$, se x é uma variável de sorte τ e ϕ é uma fórmula então $\forall_\tau x\phi$ e $\exists_\tau x\phi$ são fórmulas.

(g) Nada mais é uma fórmula.

Definição A linguagem polisortida deôntica de ações dada por um alfabeto consiste do conjunto de todas as fórmulas bem formadas construídas com símbolos do alfabeto.

Definição Achamos mais conveniente usar a notação $\forall x/\tau\phi$ no lugar de $\forall_\tau x\phi$. Similarmente, usaremos a notação $\exists x/\tau\phi$ no lugar de $\exists_\tau x\phi$. $\forall(\phi)$ denota o fechamento universal da fórmula ϕ e $\exists(\phi)$ denota o seu fechamento existencial. Estes fechamentos podem ser obtidos prefixando-se a fórmula ϕ com quantificadores de tipos apropriados.

Definição Se ϕ, ψ e χ são fórmulas, A é um termo de sorte *Agt*. α é um termo de sorte *Act* e σ é um termo de sorte $T(Act)$, então as sguintes expressões são axiomas:

1. $\phi \rightarrow (\psi \rightarrow \phi)$
2. $(\chi \rightarrow (\phi \rightarrow \psi)) \rightarrow ((\chi \rightarrow \phi) \rightarrow (\chi \rightarrow \psi))$
3. $(\neg\phi \rightarrow \neg\psi) \rightarrow ((\neg\phi \rightarrow \psi) \rightarrow \phi)$
4. $\forall x\phi(x) \rightarrow \phi(t)$ onde t é livre para x em ϕ
5. $(\forall x(\phi \rightarrow \psi)) \rightarrow (\varphi \rightarrow \forall x\psi)$ onde x não é livre em ϕ

6. $[A,\alpha]\top$
7. $([A,\alpha](\phi \rightarrow \psi)) \rightarrow (([A,\alpha]\phi) \rightarrow ([A,\alpha]\psi))$
8. $([A,\alpha]\neg\phi \rightarrow (\neg[A,\alpha]\phi)$
9. $\forall x([A,\alpha]\phi) \leftrightarrow ([A,\alpha]\forall x\phi)$ onde x não é livre em α ou A
10. $\exists x([A,\alpha]\phi) \rightarrow [A,\alpha]\exists x\phi)$ onde x não é livre em α ou A
11. $(([A,\alpha]\phi) \wedge ([A,\alpha]\psi)) \leftrightarrow ([A,\alpha]\phi \wedge \psi)$
12. $(([A,\alpha]\phi) \vee ([A,\alpha]\psi)) \rightarrow ([A,\alpha]\phi \vee \psi)$
13. $[[A,< \alpha >]]\phi \leftrightarrow [A,\alpha]\phi$
14. $[[A,\alpha,\sigma]]\phi \leftrightarrow [A,\alpha]\ [[A,\sigma]]\ \phi$
15. $\hat{n} \rightarrow (per(A,\alpha) \leftrightarrow [A,\alpha]\hat{n})$
16. $\neg[A,\alpha]\hat{n} \rightarrow [A,\alpha]\neg\hat{n}$
17. $pref(A,\alpha) \leftrightarrow \exists\beta(\neg(\beta = \alpha) \wedge per(A,\beta))$
18. $obl(A,\alpha) \rightarrow per(A,\alpha)$
19. $obl(A,\alpha) \rightarrow \neg pref(A,\alpha)$
20. $obls(A,< \alpha >) \leftrightarrow obl(A,\alpha)$
21. $obls(A,\alpha,\sigma) \leftrightarrow (obl(A,\alpha) \wedge [A,\sigma]obls(A,\sigma))$

Definição As regras de inferência sao dadas por:

1. Generalização :

$$\frac{\vdash \phi}{\vdash \forall x\,.\,\phi}$$

2. Modus Ponens:

$$\frac{\vdash \phi, \vdash (\phi \rightarrow \psi)}{\vdash \psi}$$

3. Necessidade:

$$\frac{\vdash_{sp} \phi}{\vdash_{sp} [\alpha]\,\phi}$$

Os conectivos não básicos são introduzidos por definição .

Definição Os conectivos $\wedge$, $\vee$ e $\leftrightarrow$ são definidos por:

($\wedge$-def) $(\phi \wedge \psi) =_{def} \neg(\phi \rightarrow \neg\psi)$

($\vee$-def) $(\phi \vee \psi) =_{def} (\neg\phi) \rightarrow \psi)$

($\leftrightarrow$-def) $(\phi \leftrightarrow \psi) =_{def} (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$

O significado de $\wedge$-def, por exemplo, é que para quaisquer fórmulas bem formadas ϕ e ψ, "$(\phi \wedge \psi)$" é uma abreviação para "$\neg(\phi \rightarrow \neg\psi)$". Também por definição , temos que:

($false$-def) $\neg\, true =_{def} false$

Para a componente de primeira orderm desta lógica tomamos, como usualmente, o quantificador $\forall_\tau$ como básico e definimos o quantificador $\exists_\tau$ como segue, onde x é uma variável do sorte τ e ϕ é uma fórmula bem formada:

($\exists_\tau$-def) $\exists_\tau\, x\, \phi =_{def} \neg \forall_\tau\, x \neg \phi$

ou, opcionalmente,

($\exists_\tau$-def) $\exists\, x / \tau\, \phi =_{def} \neg \forall\, x / \tau \neg \phi$

Além disso, assumimos as seguintes propriedades para a igualdade:

(Ref) $\forall\, x / \tau\, x =_\tau x$

(Sim) $\forall\, x / \tau \forall\, y / \tau\, (x =_\tau y \rightarrow y =_\tau x)$

(Trans) $\forall\, x / \tau \forall\, y / \tau \forall\, z / \tau\, (x =_\tau y \wedge y =_\tau z \rightarrow x =_\tau z)$

Observamos que existe uma transformação de fórmulas polisortidas em fórmulas monosortidas, o que mostra que a aparente generalidade fornecida pelas lógicas polisortidas é apenas relativa ([Enderton 72]). Esta transformação permite que se reduza a prova de um teorema em uma lógica polisortida no teorema correspondente em uma lógica sem sortes. Usaremos esta transformação como uma etapa do processo de prova automática de teoremas da lógica polisortida deôntica de ações .

3 Prova Automática de Teoremas em LDA

Nesta seção descreveremos um cálculo de tableaux para a lógica polisortida deôntica de ações LDA que serve de base para a construção de um provador automático para esta lógica. Tal provador se constitue o núcleo de um sistema para a descrição da arquitetura de sistemas de software, dos seus aspectos dinâmicos e prescritivos e para a investigação através do aparato dedutivo da lógica a respeito da viabilidade de certas mudanças tanto sobre a estrutura quanto sobre a funcionalidade de sistemas de software serem realizadas e de suas possíveis consequências.

Um tableau semântico é um método para a obtençao da forma normal disjuntiva de uma dada fórmula de tal modo a fazer cada componente desta forma disjuntiva ocupar um ramo de uma estrutura de árvore. Na construção da árvore procuramos subfórmulas insatisfatíveis em cada ramo da árvore. Dizemos que esses ramos são fechados e e um tableaué um tableau fechado se todos os ramos da árvore são fechados. O método de tableau aplicado á lógica clássica tem seu locus classicus em [Smullyan 68]. A descrição de sistemas de tableau para as lógicas não clássicas modais e intuicionista pode ser encontrada em [Fitting 83] e [Bell, Machover 77].

Antes de apresentarmos as regras do sistema de tableaux para a componente de primeira ordem polisortida de LDA, seja a seguinte definição .

Definição Seja ϕ uma fórmula e x_τ uma variável do sorte τ e t_τ um têrmo do mesmo sorte τ. Por $\phi(x_\tau/t_\tau)$ nós denotamos a expressão obtida substituindo-se toda ocorrência livre de x_τ e, ϕ por t_τ.

As regras do sistema de tableau para o componente polisortido de primeira ordem de LDA podem ser divididas em quatro tipos:

Definição Sejam ϕ e ψ fórmulas polisortidas de primeira ordem,

Regra do tipo *I*:

$$\frac{\neg\neg\phi}{\phi}$$

$$\frac{\phi \wedge \psi}{\begin{array}{c}\phi \\ \psi\end{array}}$$

$$\frac{\neg(\phi \vee \psi)}{\begin{array}{c}\neg\phi \\ \neg\psi\end{array}}$$

$$\frac{\neg(\phi \rightarrow \psi)}{\begin{array}{c}\phi \\ \neg\psi\end{array}}$$

Regra do tipo *II*:

$$\frac{(\phi \vee \psi)}{\phi \mid \psi}$$

$$\frac{(\phi \rightarrow \psi)}{\neg\phi \mid \psi}$$

$$\frac{\neg(\phi \wedge \psi)}{\neg\phi \mid \neg\psi}$$

Regra de tipo *III*:

$$\frac{\forall_r\, x_r\, \phi}{\phi(x_r/b_r)}$$

$$\frac{\neg\exists_r\, x_r\, \phi}{\neg\phi(x_r/b_r)}$$

onde b é qualquer constante (ou têrmo aterrado).

Regra do tipo *IV*:

$$\frac{\exists_r\, x_r\, \phi}{\phi(x_r/b_r)}$$

$$\frac{\neg\forall_r\, x_r\, \phi}{\neg\phi(x_r/b_r)}$$

onde b é uma constante nova no ramo.

Para obtermos um procedimento sistemático correto e completo para este método de tableaux extendemos o procedimento para o caso clássico de primeira ordem contido em [Smullyan 68]. Este procedimento estendido pode ser sumarizado como segue:

1. Enumere todos os símbolos constantes.

2. Dê prioridade às fórmulas mais próximas da raiz do tableau.

3. Use cada fórmula somente uma vez.

4. Mude a regra do tipo III para:

$$\frac{\forall_\tau\, x_\tau\, \phi}{\begin{array}{c}\forall_\tau\, x_\tau\, \phi \\ \phi(x_\tau/b_\tau)\end{array}}$$

$$\frac{\neg\, \exists_\tau\, x_\tau\, \phi}{\begin{array}{c}\neg\, \exists_\tau\, x_\tau\, \phi \\ \neg\, \phi(x_\tau/b_\tau)\end{array}}$$

onde, na primeira (segunda) regra b é a primeira constante tal que $\phi(x_\tau/b_\tau)$ $(\neg\, \phi(x_\tau/b_\tau))$ não esteja ainda no ramo.

Para extendermos este sistema para incluir ações não constantes associamos à cada fórmula um rótulo que nomeia o estado definido como segue.

Definição Um rótulo é uma sequência cujos elementos podem ser nomes de ação α ou $n : \alpha$ onde n é um inteiro positivo.

Com isto, as regras de tableau que tratam ações , incluindo agentes como parte do têrmo para a modalidade de ação são dadas abaixo.

Regra do tipo V:

$$\frac{\sigma\, [A,\, \alpha]\, \phi}{\sigma'\, \phi}$$

onde o rótulo σ' é obtido a partir do rótulo σ e do têrmo de ação α através da concatenação da sequência σ com a sequência $< \alpha >$, isto é, $\sigma' = < \alpha > \,\|\, \sigma$ onde $\|$ denota concatenação de sequências.

Regra do tipo VI:

$$\frac{\sigma\, \neg\, [A,\, \alpha]\, \phi}{\sigma''\, \neg\, \phi}$$

onde $\sigma' = < n : \alpha > \,\|\, \sigma$ e n é o primeiro inteiro positivo tal que $n > j$ para todo j para o qual $j : \alpha$ aparece no ramo.

Com a introdução de rótulos como definidos anteriormente, o critério de fechamento de ramos do tableau é extendido de tal modo que um ramo é dito fechado se contém as fórmulas $\sigma\, \phi$ e $\sigma'\, \neg\, \phi$ onde σ e σ' devem poder se unificar de acordo com a definição seguinte.

Definição Dois elementos de sequências de rótulos e_1 e e_2 são unificáveis se um dêles for α onde α é um têrmo de ação e o outro for $n : \alpha$ para qualquer inteiro positivo n ou for o próprio α. Caso contrário, dizemos que tais elementos não são unificáveis. Esta unificação dá como resultado $e_r = n : \alpha$ se $e_1 = \alpha$ e $e_2 = n : \alpha$ (ou vice-versa) ou $e_r = \alpha$ se $e_1 = e_2 = \alpha$.

Definimos ainda um algoritmo de unificação de rótulos σ e σ' a seguir.

Definição Algoritmo de unificação para os rótulos σ e σ':

1. Faça $k = 0$ e $\eta = <>$ e $\sigma_0 = \sigma$ e $\sigma'_0 = \sigma'$.

2. Se $\sigma_k = <>$ e $\sigma'_k = <>$ então σ e σ' são unificáveis e η_k é o unificador mais geral de rótulos.

3. Se no par σ_k, σ'_k um dêles é $<>$ e o outro é diferente de $<>$ (sequência vazia), então pare; σ e σ' não são unificáveis.

4. Sejam $\sigma_k = <\overline{e} \mid \overline{\sigma}>$ e $\sigma'_k = <\overline{e}' \mid \overline{\sigma}'>$ onde com isso obtemos os elementos frontais $\overline{e}$ e $\overline{e}'$ das listas σ_k e σ'_k, respectivamente. Se os elementos $\overline{e}$ e $\overline{e}'$ podem ser unificados e obtivermos e_r como resultado de tal unificação então faça

$$\begin{aligned} \sigma_{k+1} &= \overline{\sigma} \\ \sigma'_{k+1} &= \overline{\sigma}' \\ \eta_{k+1} &= <e_r> \parallel \eta_k \end{aligned}$$

e vá para 2. Caso contrário, pare; σ e σ' não são unificáveis.

Convém que façamos aqui uma comparação do presente sistema de tableau para lógica polisortida de primeira ordem com ações variáveis, com o sistema de tableau para a lógica monosortida de primeira ordem com ações constantes proposto em [Costa 89], especialmente no que concerne às regras envolvendo ações . Para isso explicitaremos as suas regras envolvendo ações e expressas através de um operador gerador de tableau γ:

Regra do tipo E:

$$\frac{[\alpha]\phi}{\gamma(\sigma', \phi)}$$

1.	$<\omega>$	$\neg([\beta][\alpha]\phi \rightarrow [\alpha]\phi)$	
2.	$<\omega>$	$[\beta][\alpha]\phi$	1, *regra I*
3.	$<\omega>$	$\neg[\alpha]\phi$	1, I
4.	$<\omega\,1:\alpha>$	$\neg\phi$	3, VI
5.	$<\omega\,\beta>$	$[\alpha]\phi$	2, V
6.	$<\omega\,\beta\,\alpha>$	ϕ	5, V

not closed

Note que o nosso sistema de tableau não permite que os rótulos das linhas 4 e 6 sejam unificados e isto implica qua esta fórmula não é teorema do nosso sistema. Além disso, não se torna necessário que uma regra (no caso a regra VI) seja aplicada antes de outra (no caso a regra V). A ordem de aplicação de tais regras não afeta o fechamento do tableau. Note ainda que a notação para os rótulos reflete de modo natural a acessibilidade entre estados. Esta notação explicita tanto a ação envolvida na transição quanto o tipo de regra envolvido (regra V ou regra VI).

Convém notarmos que em relação ao componente polisortido do sistema de tableau descrito até aqui, optamos por indiciar todos os termos com os índices de seus sortes respectivos. No caso de símbolos funcionais usamos como índice o sorte do resultado da aplicação funcional.

Alternativamente, poderíamos ter usado uma transformação de fórmulas polisortidas em fórmulas monosortidas (ver, por exemplo, [Enderton 72]). Esta transformação , que nos permite reduzir a prova de um teorema em uma lógica polisortida no teorema correspondente em uma lógica sem sortes, é dada a seguir.

Definição Seja ϕ uma fórmula da lógica polisortida de ações . Para cada sorte τ, associamos um novo símbolo de predicado unário também denotado por τ. Então a fórmula sem sortes ϕ^* de ϕ é a fórmula da lógica (monosortida) de primeira ordem com ações obtida a partir de ϕ pela aplicação das seguintes transformações à todas as subfórmulas de ϕ da forma $\forall\, x/\tau\ \psi$ e $\exists\, x/\tau\ \psi$:

1. Substitua $\forall\, x/\tau\ \psi$ por $\forall\, x\,(\psi \leftarrow \tau(x))$
2. Substitua $\exists\, x/\tau\ \psi$ por $\exists\, x\,(\psi \wedge \tau(x))$

onde σ' é um novo tableau ou um tableau previamente gerado pela aplicação da regra E ou F à uma fórmula do ramo por causa da mesma ação α.

Regra do tipo F:

$$\frac{\neg [\alpha] \phi}{\gamma(\sigma', \neg \phi)}$$

onde σ' é um novo tableau.

Portanto o operador γ gera um novo tableau cujo nome é σ' e cuja fórmula raiz é ϕ, ou ele adiciona ϕ a σ se σ é um tableau já existente). Se σ é fachado, então o ramo original também é considerado fechado.

Consideremos o seguinte exemplo simples abaixo, que consiste em provarmos através das regras de [Costa 89] se a fórmula

$$[\beta] [\alpha] \phi \rightarrow [\alpha] \phi$$

é válida. Mostramos a seguir que utilizando o seu sistema de tableau podemos mostrar que tal fórmula é válida:

1.	$\sigma 1$	$\neg ([\beta] [\alpha] \phi \rightarrow [\alpha] \phi)$	
2.	$\sigma 1$	$[\beta] [\alpha] \phi$	1, *regra* $(\neg \rightarrow)$
3.	$\sigma 1$	$\neg [\alpha] \phi$	1, $(\neg \rightarrow)$
4.	$\sigma 2$	$\neg \phi$	3, (F)
5.	$\sigma 3$	$[\alpha] \phi$	2, (E)
6.	$\sigma 2$	ϕ	5, (E)
closed			

Note que não é claro que os estados alcançados na linhas 4 e 6 possam ser iguais. Observe ainda que para obter o fechamento do tableau a regra F tem que ser utilizada antes da regra E. No caso geral, para se obter o fechamento deve-se forçar que para uma dada ação α a regra F seja utilizada antes da regra E. Note também que o nome dos tableaus não contém qualquer referência com relação à acessibilidade: qual o tableau anterior, qual a ação que foi realizada e qual a regra aplicada na transição (E ou F).

Se aplicarmos o sistema de tableau apresentado nesta seção , obtemos o seguinte, se utilizarmos $< \omega >$ como rótulo inicial (no caso retringirmos a aplicação das regras envolvendo ações à ações constantes):

Por completeza, devemos mencionar que outras abordagens para o tratamento de lógicas polisortidas de primeira ordem são dados em [Goguen et al. 85], [Cunninghan,Dick 85], [Walther 86] e [Cohn 87], por exemplo.

Para incorporar o predicado de igualdade á abordagem do presente sistema de tableaux, nós extendemos o método da unificação parcial proposto em [Reeves 87]. Para isso, definimos primeiramente o que é a diferença entre um par de têrmos aterrados.

Definição A diferença de um par de têrmos aterrados é dada por:

$$dif\,ere(t_a, t_b) = \begin{cases} [(a_{1_{r_1}}, b_{1_{r_1}}), \ldots, (a_{n_{r_n}}, b_{n_{r_n}})] & \text{se } f_r = g_r \\ [\,] & \text{caso contrário} \end{cases}$$

Em essência, o procedimento de se encontrar diferenças entre sentenças convenientes e adicionar condições de desigualdade ao tableau é o que chamamos de unificação parcial. As regras que descrevem a unificação parcial são dadas como segue.

Definição Suponha, para quaisquer literais aterrados L_1 e L_2 da forma $\sigma\ p(a_{1_{r_1}}, \ldots, a_{n_{r_n}})$ e $\sigma'\ p(b_{1_{r_1}}, \ldots, b_{n_{r_n}})$,respectivamente, que uma vez que estes apareçam em um ramo, podemos introduzir a seguinte desigualdade:

$$\neg\,(a_{1_{r_1}} = b_{1_{r_1}} \wedge \ldots \wedge a_{n_{r_n}} = b_{n_{r_n}})$$

Esta regra pode ser expressa por:

$$\frac{\sigma\ p(a_{1_{r_1}}, \ldots, a_{n_{r_n}}),\ \sigma'\ p(b_{1_{r_1}}, \ldots, b_{n_{r_n}})}{\eta\ \neg\,(a_{1_{r_1}} = b_{1_{r_1}} \wedge \ldots \wedge a_{n_{r_n}} = b_{n_{r_n}})}$$

Além disso, para qualquer literal aterrao da forma $\neg\,(t_a = t_b)$, onde t_a e t_b são dois têrmos (aterrados) quaisquer, podemos introduzir a desigualdade:

$$\neg\,(a_{1_{r_1}} = b_{1_{r_1}} \wedge \ldots \wedge a_{n_{r_n}} = b_{n_{r_n}})$$

onde

$$[(a_{1_{r_1}}, b_{1_{r_1}}), \ldots, (a_{n_{r_n}}, b_{n_{r_n}})] = dif\,ere(t_a, t_b).$$

Esta regra pode ser expressa por:

$$\frac{\sigma\ \neg\,(t_a = t_b)}{\sigma\ \neg\,(a_{1_{r_1}} = b_{1_{r_1}} \wedge \ldots \wedge a_{n_{r_n}} = b_{n_{r_n}})}$$

onde

$$dif ere(t_a, t_b) = [(a_{1_{r_1}}, b_{1_{r_1}}), \ldots, (a_{n_{r_n}}, b_{n_{r_n}})]$$

Note que restringimos aqui as regras que descrevem a unificação parcial à formulas associadas à rótulos unificáveis. Na primeira regra acima σ e σ' são rótulos unificáveis e η é o unificador mais geral de rótulos resultante do processo de unificação . Na segunda regra apenas mantivemos o mesmo rótulo da fórmula inicial.

Finalmente, o componente deôntico de LDA nos leva às seguintes regras de tableau:

Regra *VII*:

$$\frac{(\neg)\ \sigma\ [[<\alpha>]]\ \phi}{(\neg)\ \sigma\ [\alpha]\ \phi}$$

Na verdade estra regra denota duas: uma sem o conectivo de negação e outra com este conectivo. O mesmo ocorre para a notação das regras que se seguem.

Regra VIII:

$$\frac{(\neg)\ \gamma\ [[\sigma, \alpha]]\ \phi}{(\neg)\ \gamma\ [[\sigma]]\ [\alpha]\ \phi}$$

Regra IX:

$$\frac{\hat{n}}{\forall\, \alpha\ (per(\alpha) \leftrightarrow [\alpha]\ \hat{n})}$$

Regra X:

$$\frac{\sigma\ \neg\ [\alpha]\ \hat{n}}{\sigma \| <\alpha>\ \ \neg\ \hat{n}}$$

Regra XI:

$$\frac{(\neg)\ \ \sigma\ \ obls(<\alpha>)}{(\neg)\ \ \sigma\ \ obl(\alpha)}$$

Regra XII:

$$\frac{(\neg)\ \ \gamma\ \ obls(\sigma, \alpha)}{(\neg)\ \ \gamma\ \ obls(\sigma)\ \wedge\ [[\sigma]]\ obl(\alpha)}$$

Regra XIII:

$$\frac{(\neg)\ \sigma\ obl(\alpha)}{\begin{array}{c}(\neg)\ \sigma\ per(\alpha) \\ (\neg)\ \sigma\ \neg\exists\beta(\neg(\beta = \alpha) \wedge per(\beta))\end{array}}$$

Uma vez que ao longo deste trabalho temos usado a letra grega σ tanto para expressar rótulos de fórmulas quanto sequências de ações , nas regras acima quando σ aparece na regra denotando uma sequência de ações , nós usamos a letra γ para denotar o rótulo da fórmula e quando σ não aparece denotando uma sequência de ações , este σ denota o rótulo da fórmula.

Observamos finalmente que do mesmo modo que podemos obter uma versão extendida do sistema de tableau com unificação para a lógica clássica (monosortida) de primeira ordem, podemos também, a princípio, obter uma versão extendida do sistema de tableau para LDA usando unificação . A equivalência entre sistemas de resolução e de tableau significa que podemos buscar uma implementação eficiente de um sistema de tableau importando, por exemplo, estratégias e meios de tratar a igualdade de sistemas de resolução . Pode-se, a princípio, obter um procedimento para o sistema de tableau para LDA conforme, por exemplo, a estratégia proposta por [Schonfeld 85].

4 Conclusão

Neste trabalho nós apresentamos um método de prova automática para uma lógica polisortida deôntica com ações arbitrárias (constantes e não constantes) e domínio constante, i. e., para a qual a fórmula de Barcan é válida. Com isso, os conjuntos de indivíduos nos diferentes mundos possíveis são idênticos. Apesar de não considerarmos aqui o caso de domínios não constantes, é possível, a princípio, extender este trabalho para tratar este caso.

O objetivo do presente trabalho é a obtenção de um método de prova automática para a lógica em questão que viabilize o raciocínio dedutivo sobre o processo de mudança tanto da estrutura quanto da funcionalidade de sistemas de software. Um tal sistema lógico com modalidades para ações e componentes deônticos fornece uma abordagem apropriada para lidar com estes aspectos do estudo de sistemas evolutivos de software ([Alencar 89]). Pode-se também pensar na aplicação de um provador de teoremas para esta lógica no contexto da expressão e raciocínio dedutivo sobre métodos de programação ([Alencar, Maibaum 90]).

Um sistema de tableux para esta lógica tem como uma vantagem o fato que todas as suas regras de inferência são regras de eliminação e não introduzem novos operadores conectivos. Além disso este sistema é bastante natural, podendo ser generalizado através da introdução de um algoritmo de unificação que reflita as várias condições pelas quais duas fórmulas são unificáveis. Esta generalização será um dos tópicos a ser descrito em um trabalho futuro.

5 Referências

[Alencar, Lucena 88] Alencar, P.S.C., Lucena, C.J.P. *Métodos Formais para o Desenvolvimento de Programas*, Editorial Kapelusz, IV EBAI, Buenos Aires, Argentina, 1988.

[Alencar 89] Alencar, P.S.C. *Uma Abordagem Lógica para Sistemas Evolutivos de Software*, Relatório Técnico, Departamento de Informática, 1989.

[Alencar, Maibaum 90] Alencar, P.S.C., Maibaum, T., *A Deontic Action Logic Metalanguage for Programming Methods*, submetido para publicação .

[Beth 59] Beth, E. W., *The foundations of Mathematics*, North-Holland, Amsterdam, 1959.

[Bittel,Alencar 88a] Bittel, O., Alencar, P.S.C., *Towards a Tableau Based Intuitionistic Theorem Prover*, RT no. 3, Departamento de Informática, PUC-RJ, 1988.

[Bittel, Alencar 88b] Bittel, O., Alencar, P. S. C., *Program Construction with an Intuitionistic Sequent Calculus*, Anais da IV Reunião de Trabalho do Projeto Estra, 1988.

[Bell,Machover 77] Bell, J., Machover, M., *A Course in Mathematical Logic*, North-Holland Pub. Co., 1977.

[Cohn 87] Cohn, A. J., *A More Expressive Formulation of Many-Sorted Logic*, Journal of Automated Reasoning, vol. 3, no. 2, 1987.

[Costa, Cunningham 88] Bell, J., Machover, M., *Mechanized Deduction and Modal Action Logic*, Forest Report 5, Imperial College, University of London.

[Cunningham, Dick 85] Cunningham, R. J., Dick, A. J. J., *Re-Write Systems in a Lattice of Types*, Acta Informatica, vol. 22, no. 2, pp. 149-169, 1985.

[Enderton 72] Enderton, H. B., *A mathematical Introduction to Logic*, Academic Press, 1972.

[Fitting 83] Fitting, M., *Proof Methods for Modal and Intuitionistic Logics*, D. Reidel Pub. Co., Dordrecht, 1983.

[Gabbay 84] Gabbay, D., Guenthner,F. (eds.), *Handbook of Philosophical Logic*, Vols. I, II,D. Reidel Pub. Co., 1984.

[Goguen et. al 85] Goguen,J. A., Jouannoud, J. P., Meseguer,J., *Operational Semantics for Order-Sorted Algebra*, SRI International, Menlo Park, CA 94305, 1985.

[Goldblatt 82] Goldblatt, R., *Axiomatizing the Logic of Computer Programming*, Lecture Notes in Computer Science 130, Springer-Verlag, 1982.

[Hintikka 55] Hintikka, J., *Form and Content in Quantification Theory*, Acta Philosophica Fennica, vol. 8, p. 7-55, 1955.

[Hughes, Cresswell 68] Hughes, G. E., Cresswell, M. J., *An Introduction to Modal Logic*, Menthuen, London, 1968. Segunda edição 1972.

[Khosla 88] Khosla, S., *System Specification: A Deontic Approach*, PhD Thesis, Department of Computing, Imperial College of Science and Technology, 1988.

[Kripke 63] Kripke, S. A., *Semantical Considerations on Modal Logic*, Acta Philosophica Fennica, vol. 16, p. 83-94, 1963.

[Prawitz 75] Prawitz,D., *Comments on Gentzen-style Procedures and the Classical Notion of Truth*, em Diller, J. e Muller, G. H., (eds.), Proof Theory Symposium, Lecture Notes in Mathematics 500, Springer, Berlin, p. 290-319, 1975.

[Reeves 87] Reeves, S. V., *Adding Equality to Semantic Tableau*, Journal of Automated Reasoning, vol. 3, 1987.

[Schonfeld 85] Schonfeld, W., *Prolog Extensions based on Tableau Calculus*, Proceedings of the IJCAI, 1985.

[Smullyan 68] Smullyan, R. M., *First Order Logic*, Springer-Verlag, Berlin, 1968.

[Walther 86] Walther, C., *A Classification of Many Sorted Unification Problems*, CADE 8, Springer Verlag LNCS 230, 1986.