ARGUMENTAÇÃO MONOLÓGICA: RACIOCÍNIO COERENTE NA PRESENÇA DE INCONSISTÊNCIAS

Luiz Felipe Zarco dos Santos[1], Lucio Nunes de Lira[2], Silvio do Lago Pereira[3]
[1] Aluno de graduação em Análise e Desenvolvimento de Sistemas – DTI / FATEC-SP
[2] Aluno de pós-graduação em Análise e Projeto de Sistemas– DTI/ FATEC-SP
[3] Prof. Dr. do Departamento de Tecnologia da Informação – FATEC-SP
luiz.santos58@fatec.sp.gov.br, lucio.lira@fatec.sp.gov.br, slago@fatecsp.br

Resumo

A argumentação é um processo que visa avaliar a coerência de uma conclusão, com base em conhecimento disponível. Normalmente, a argumentação envolve pelo menos dois agentes com pontos de vista divergentes. Na argumentação monológica, porém, um único agente tem todo o conhecimento necessário para gerar argumentos a favor ou contra uma conclusão. Como esses argumentos são conflitantes, para raciocinar coerentemente, o agente precisa simular um diálogo consigo mesmo, pesando prós e contras. Assim, o objetivo deste artigo é definir os fundamentos lógicos de um sistema computacional de argumentação monológica, capaz de simular raciocínio coerente na presença de inconsistências, bem como descrever um protótipo do sistema que foi desenvolvido em Prolog e relatar resultados empíricos obtidos com ele.

1. Introdução

Em todas as áreas do conhecimento, a tomada de decisões envolve algum tipo de argumentação em que são avaliados prós e contras a respeito de uma determinada conclusão. Em todos os casos, a argumentação é feita a partir de informações imperfeitas (incompletas ou conflitantes) e requer algum tipo de raciocínio cujas conclusões possam ser anuladas em face de novas evidências.

A *argumentação* [1] é um processo que normalmente envolve dois agentes, com conhecimentos conflitantes, que visa verificar a coerência de uma determinada conclusão. Na argumentação monológica, porém, um único agente detém todo o conhecimento necessário para gerar argumentos a favor ou contra uma conclusão. Como esses argumentos são conflitantes, para raciocinar de forma coerente, o agente precisa simular um diálogo consigo mesmo (*refletir*), pesando prós e contras.

Particularmente, em Inteligência Artificial (IA), a argumentação é um processo que *constrói* argumentos (a partir do conhecimento disponível), *identifica* conflitos entre eles, *resolve* estes conflitos e, finalmente, *decide* se uma determinada conclusão é ou não coerente. De fato, o objetivo deste artigo é definir os fundamentos lógicos da argumentação monológica, bem como descrever um protótipo de sistema criado para automatizar esse processo.

O restante deste artigo está organizado do seguinte modo: a Seção 2 define os fundamentos lógicos do sistema de argumentação monológica proposto neste artigo; a Seção 3 descreve as principais funcionalidades de um protótipo desse sistema, que foi desenvolvido em linguagem Prolog; a Seção 4 relata resultados empíricos obtidos com esse protótipo; e, por fim, a Seção 5 apresenta as conclusões finais do trabalho.

2. Fundamentos do Sistema Desenvolvido

Em IA, uma base de conhecimentos é um conjunto de regras descrevendo o conhecimento de um especialista em um domínio específico. Tradicionalmente, assume-se que uma base de conhecimentos é *consistente* (i.e., não tem contradições); caso contrário, ela é inútil para um sistema de raciocínio automatizado. Essa suposição é motivada pelo princípio *ex falso quodlibet* [2], segundo o qual qualquer coisa pode ser inferida de uma contradição.

A despeito disso, há diversas aplicações práticas de raciocínio automatizado em que, devido à existência de *exceções*, bases *inconsistentes* devem ser usadas (e.g., direito, política e medicina) [3]. Por exemplo, seja Δ uma base de conhecimentos com as informações♣: '*Tina is a bird*', '*Birds fly*' e '*Chickens do not fly*'. Sem evidência contrária, é coerente inferir '*Tina flies*' de Δ. Agora, suponha que Δ seja atualizada com a informação '*Tina is a chicken*', resultando numa nova base Δ'. Nesse caso, ambas as conclusões '*Tina flies*' e '*Tina does not fly*' podem ser inferidas de Δ' e isso não é coerente (pois é contraditório). Um modo de restaurar a consistência de Δ' seria remover a regra '*Birds fly*' (ou, sua exceção '*Chickens do not fly*'), mas isto destruiria conhecimento importante. Uma alternativa melhor seria, então, dar *precedência* às exceções. Neste caso, apenas '*Tina does not fly*' poderia ser coerentemente inferida de Δ'. De fato, usando relações de precedência, raciocínio coerente na presença de inconsistência torna-se possível.

Segundo Dung [4], argumentação deve ser baseada no princípio de que '*quem dá a última palavra vence*'. Por exemplo, seja Δ'' uma base de conhecimentos com as informações inconsistentes: '*Tina is a chicken*', '*Tina is scared*', '*Chickens are birds*', '*Birds fly*', '*Chickens do not fly*' e '*Scared chickens fly*'. Então, a partir de Δ'', podemos construir os seguintes argumentos conflitantes:

- A: *'Tina is a chicken. Chickens are birds. Birds fly. Therefore, Tina flies.'*
- B: *'Tina is a chicken. Chickens do not fly. Therefore, Tina does not fly.'*
- C: *'Tina is a chicken. Tina is scared. Scared chickens fly. Therefore, Tina flies'.*

Considerando que '*Chickens do not fly*' é uma exceção à regra '*Birds fly*', e que exceções têm precedência, B ataca A. Portanto, se a argumentação terminasse neste ponto, a conclusão final seria '*Tina does not fly*'. Porém, uma argumentação só termina quando não há argumentos

♣ *Textos descrevendo conhecimento estão em inglês visando à leitura mais fluente das regras codificadas no sistema criado.*

mais fortes, e como '*Scared chickens fly*' é uma exceção à regra '*Chickens do not fly*', C derrota B (e restabelece A). Logo, a única conclusão coerente inferida de Δ'' é '*Tina flies*'. Este processo argumentativo é sumarizado no grafo da Figura 1, em que os vértices A e C indicam argumentos cujas conclusões são coerentes com Δ''.

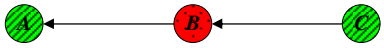


Figura 1 – Argumentos coerentes com Δ''.

2.1. Representação de Conhecimento

Neste trabalho, o conhecimento é representado em lógica proposicional [5]. Um *átomo* denota uma proposição. Um *literal* λ é um átomo α ou sua negação $\neg\alpha$. Os literais α e $\neg\alpha$ são *complementares*. O literal especial $\top$ denota uma proposição verdadeira e não tem complementar. Uma *conjunção* é uma expressão $\lambda_1 \wedge ... \wedge \lambda_k$, em que cada λ_i ($1 \le i \le k$) é um literal. Usamos $\Lambda(\lambda_1 \wedge ... \wedge \lambda_k)$ para denotar $\{\lambda_1,...,\lambda_k\}$. Particularmente, $\Lambda(\top) = \varnothing$.

Uma *regra revogável* é uma expressão $\ell : \varphi \to \lambda$, em que o *rótulo* ℓ é o identificador da regra, o *antecedente* φ é uma conjunção e o *consequente* $\lambda \neq \top$ é um literal. Se $\varphi = \top$, a regra revogável ℓ é chamada *presunção*. Duas regras revogáveis ℓ e ℓ' são *conflitantes*, denotado por $\ell \lozenge \ell'$, se elas têm consequentes complementares.

Uma *regra de precedência* é uma expressão $\ell \prec \ell'$, declarando que a regra revogável ℓ tem precedência sobre a regra revogável ℓ'. Regras de precedência são usadas para resolver conflitos entre regras revogáveis. Se ℓ e ℓ' são regras revogáveis conflitantes e $\ell' \not\prec \ell$, então ℓ *ataca* ℓ' (mas ℓ' não ataca ℓ). Note que, se $\ell \not\prec \ell'$ e $\ell' \not\prec \ell$, então as regras revogáveis ℓ e ℓ' *bloqueiam* uma à outra.

Uma *base de conhecimentos* Δ é um conjunto de regras revogáveis e regras de precedência.

2.2. Construção de Argumentos

Uma *árvore de derivação* de um literal λ a partir de uma base Δ, denotada por Υ_Δ^λ, é uma árvore tal que:

- A raiz de Υ_Δ^λ é rotulada com o literal λ.
- Para cada nó em Υ_Δ^λ rotulado com um literal λ', há uma regra revogável $\varphi \to \lambda' \in \Delta$.
- Se $\varphi = \top$, então o nó rotulado com λ' é um nó folha em Υ_Δ^λ; senão, se $\varphi = \lambda_1 \wedge ... \wedge \lambda_k$, esse nó tem k nós filhos rotulados com $\lambda_1,...,\lambda_k$, respectivamente.

Usamos $\Delta \vdash \lambda$ para indicar que existe uma árvore Υ_Δ^λ.

Por exemplo, a Figura 2 mostra uma árvore de derivação de f a partir da base de conhecimentos Δ^* definida a seguir (onde os átomos c, s, d, b e f denotam '*chicken*', '*scared*', '*dead*', '*bird*' e '*fly*', respectivamente).

$$\Delta^* = \left\{ \begin{array}{l} 1 : \top \to c, \quad 2 : \top \to s, \quad 3 : \top \to d, \quad 4 : c \to b, \\ 5 : b \to f, 6 : c \to \neg f, 7 : c \wedge s \to f, 8 : c \wedge d \to \neg f, \\ 6 \prec 5, \quad 7 \prec 6, \quad 8 \prec 7 \end{array} \right\}$$

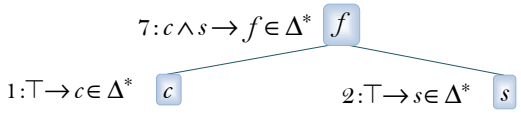


Figura 2 – Árvore de derivação de f a partir de Δ^*.

Uma árvore de derivação de λ a partir de Δ pode ser automaticamente gerada por um algoritmo de *busca regressiva em profundidade* [6]. Uma vez gerada essa árvore, o argumento correspondente pode ser facilmente extraído dela (e.g., coletando numa lista toda fórmula cujo consequente rotula um nó dessa árvore). Por exemplo, para a árvore na Figura 2, o argumento construído é:

$$\{1 : \top \to c, 2 : \top \to s, 7 : c \wedge s \to f\} \vdash f$$

2.3. Identificação de Conflitos entre Argumentos

Podemos usar o algoritmo citado na Seção 2.2 para obter todos os argumentos que podem ser construídos a partir de uma base de conhecimentos Δ. Por exemplo, o conjunto de todos os argumentos que podem ser automaticamente construídos a partir de Δ^* é o seguinte:

$$A_1 = \{1 : \top \to c\} \vdash c$$
$$A_2 = \{2 : \top \to s\} \vdash s$$
$$A_3 = \{3 : \top \to d\} \vdash d$$
$$A_4 = \{1 : \top \to c, 4 : c \to b\} \vdash b$$
$$A_5 = \{1 : \top \to c, 6 : c \to \neg f\} \vdash \neg f$$
$$A_6 = \{1 : \top \to c, 4 : c \to b, 5 : b \to f\} \vdash f$$
$$A_7 = \{1 : \top \to c, 2 : \top \to s, 7 : c \wedge s \to f\} \vdash f$$
$$A_8 = \{1 : \top \to c, 3 : \top \to d, 8 : c \wedge d \to \neg f\} \vdash \neg f$$

O *grafo de conflitos* correspondente a esse conjunto de argumentos é ilustrado na Figura 3. Observe que, sem regras de precedência, os *conflitos* entre A_5, A_6, A_7 e A_8 não podem ser resolvidos. Consequentemente, nem f, nem $\neg f$, pode ser aceito como conclusão coerente de Δ^*.

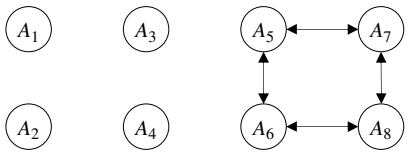


Figura 3 – Conflitos entre argumentos de Δ^*.

2.4. Relações de Precedência

Seja L_Δ o conjunto de rótulos ℓ usados numa base Δ. Uma relação binária $\prec$ é uma *ordem parcial estrita* sobre L_Δ se ela é *antirreflexiva* ($\ell \not\prec \ell$) e *transitiva* (se $\ell \prec \ell'$ e $\ell' \prec \ell''$, então $\ell \prec \ell''$). Claramente, se $\prec$ é antirreflexiva e transitiva, ela é *antissimétrica* (se $\ell \prec \ell'$, então $\ell' \not\prec \ell$).

Seja $\Pi_\Delta^e = \{\ell \prec \ell' \in \Delta\}$ o conjunto de regras de precedência *explicitamente* declaradas em Δ. Assumimos que Π_Δ^e é antirreflexiva e antissimétrica. Logo, o fecho transitivo de Π_Δ^e, denotado por $\mathcal{E}_\Delta^c$, é uma ordem parcial estrita sobre L_Δ. Ademais, como regras de precedência entre regras revogáveis não conflitantes são inúteis, definimos $\mathcal{E}_\Delta = \{\ell \prec \ell' \in \mathcal{E}_\Delta^c : \ell \lozenge \ell'\}$. De fato, o conjunto $\mathcal{E}_\Delta$ é uma *relação de precedência explícita* sobre regras revogáveis declaradas em Δ. Por exemplo, para Δ^* temos:

- $\Pi_{\Delta^*}^e = \{6 \prec 5, 7 \prec 6, 8 \prec 7\}$
- $\mathcal{E}_{\Delta^*}^c = \{6 \prec 5, 7 \prec 5, 7 \prec 6, 8 \prec 5, 8 \prec 6, 8 \prec 7\}$
- $\mathcal{E}_{\Delta^*} = \{6 \prec 5, 7 \prec 6, 8 \prec 5, 8 \prec 7\}$

Na Figura 4, regras de precedência em $\Pi_{\Delta^*}^e$ e regras de precedência *deduzidas por transitividade* são indicadas por linhas contínuas e pontilhadas, respectivamente.

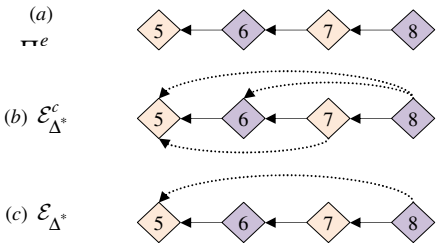


Figura 4 – Síntese da relação de precedência explícita $\mathcal{E}_{\Delta^*}$.

Uma relação de precedência implícita sobre regras revogáveis em Δ também pode ser definida com base em *especificidade* [7]. Neste trabalho, adotamos um critério de especificidade que favorece dois aspectos de uma regra revogável: *precisão* (quantidade de informação no antecedente da regra) e *concisão* (número de passos necessários para derivar o antecedente da regra).

Sejam $\ell_1 : \varphi_1 \to \lambda_1$ e $\ell_2 : \varphi_2 \to \lambda_2$ regras revogáveis conflitantes em Δ e $\Delta_0 = \{\varphi \to \lambda \in \Delta : \varphi \neq \top\}$ o conjunto de regras revogáveis de Δ que não são presunções. Então, ℓ_1 é *mais específica* que ℓ_2, denotado por $\ell_1 \lhd \ell_2$, se e somente se $\Delta_0 \cup \{\top \to \lambda : \lambda \in \Lambda(\varphi_1)\} \vdash \lambda$, para *todo* literal $\lambda \in \Lambda(\varphi_2)$, e $\Delta_0 \cup \{\top \to \lambda : \lambda \in \Lambda(\varphi_2)\} \nvdash \lambda$, para *algum* literal $\lambda \in \Lambda(\varphi_1)$. Intuitivamente, $\ell_1 \lhd \ell_2$ indica que o antecedente de ℓ_2 pode ser derivado do antecedente de ℓ_1, mas não o contrário (i.e., ℓ_1 é uma exceção de ℓ_2). Por exemplo, em Δ^*, a regra $7 : c \wedge s \to f$ é mais específica que $6 : c \to \neg f$ (pois c é derivável de $c \wedge s$, mas $c \wedge s$ não é derivável de c). Neste caso, a regra 7 é mais *precisa* do que a regra 6 porque seu antecedente é mais informativo. Analogamente, a regra $6 : c \to \neg f$ é mais específica do que $5 : b \to f$ (pois b é derivável de c, via $4 : c \to b$, mas c não é derivável de b). Nesse caso, a regra 6 é mais *concisa* que a regra 5 porque seu antecedente é mais direto.

Seja $\mathcal{I}_\Delta = \{\ell \prec \ell' : \ell, \ell' \in L_\Delta \text{ e } \ell \lhd \ell'\}$ o conjunto de regras de precedência implícitas, *sintetizado por especificidade*, a partir das regras revogáveis em Δ. Então, $\mathcal{I}_\Delta$ é uma relação *antirreflexiva* (pois especificidade vale apenas para regras conflitantes), *antissimétrica* (pois se $\ell \lhd \ell'$, o antecedente de ℓ' é derivável do antecedente de ℓ, mas não vice-versa), e *transitiva*, sob o aspecto de conflito (pois se $\ell \lhd \ell'$, $\ell' \lhd \ell''$ e $\ell'' \lhd \ell'''$, então $\ell \lozenge \ell'''$). Logo, $\mathcal{I}_\Delta$ é uma *relação de precedência implícita* sobre as regras revogáveis declaradas em Δ. Assim, por exemplo, as regras de precedência $6 \prec 5$ e $7 \prec 6$ não precisariam ser declaradas em Δ^*; pois, usando o critério de especificidade que definimos, é possível sintetizar o conjunto $\mathcal{I}_{\Delta^*} = \{6 \prec 5, 7 \prec 6\}$, a partir das regras revogáveis em Δ^*.

Como a síntese da relação de precedência implícita é baseada apenas na sintaxe das regras revogáveis em Δ, ela tem a vantagem de ser independente do domínio de aplicação. Porém, como nem toda regra de precedência pode ser definida em termos de especificidade (como, por exemplo, é o caso da regra $8 \prec 7$ em Δ^*), em geral, uma base contém regras de precedência explícitas dadas por um especialista do domínio. Então, uma *relação de precedência mista* (combinando precedências explícitas e implícitas) pode ser usada. O problema é que o fecho transitivo de $\mathcal{E}_\Delta \cup \mathcal{I}_\Delta$ nem sempre é uma ordem parcial estrita sobre L_Δ, pois $\mathcal{E}_\Delta$ e $\mathcal{I}_\Delta$ podem discordar sobre a precedência relativa de duas regras revogáveis. Por exemplo, se $\{\ell_1 \prec \ell_2, \ell_2 \prec \ell_3\} \subseteq \mathcal{E}_\Delta$ e $\{\ell_3 \prec \ell_4, \ell_4 \prec \ell_1\} \subseteq \mathcal{I}_\Delta$, o fecho transitivo de $\mathcal{E}_\Delta \cup \mathcal{I}_\Delta$ não é antissimétrico (pois contém $\ell_1 \prec \ell_4$ e $\ell_4 \prec \ell_1$), nem antirreflexivo (pois se ele contém $\ell_1 \prec \ell_4$ e $\ell_4 \prec \ell_1$, então também contém $\ell_1 \prec \ell_1$). Ou seja, é possível que $\mathcal{E}_\Delta \cup \mathcal{I}_\Delta$ contenha *ciclos*.

Como solução, propomos um algoritmo que integra $\mathcal{E}_\Delta$ e $\mathcal{I}_\Delta$, dando preferência às precedências explícitas ao eliminar ciclos. Esse algoritmo inicia com $\Pi^m_\Delta := \mathcal{E}_\Delta \cup \mathcal{I}_\Delta$ e, enquanto Π^m_Δ for uma relação *cíclica*, ele obtém o conjunto W de *elos mais fracos* num ciclo mínimo C em Π^m_Δ e faz $\Pi^m_\Delta := \Pi^m_\Delta - W$ (para um ciclo $C = \langle \ell_1 \prec \ell_2, \ldots, \ell_k \prec \ell_1 \rangle$, W é $\{\ell \prec \ell' \in C : \ell \prec \ell' \notin \mathcal{E}_\Delta, \ell' \prec \ell'' \in C \text{ e } \ell' \prec \ell'' \in \mathcal{E}_\Delta\}$). No fim, Π^m_Δ é uma relação *acíclica* e $\mathcal{E}_\Delta \subseteq \Pi^m_\Delta$. Logo, o fecho transitivo $\mathcal{M}^c_\Delta$ de Π^m_Δ é uma relação de ordem parcial estrita sobre L_Δ e $\mathcal{M}_\Delta = \{\ell \prec \ell' \in \mathcal{M}^c_\Delta : \ell \lozenge \ell'\}$ é uma *relação de precedência mista* para regras revogáveis em Δ.

Por exemplo, seja $\Delta^- = \Delta^* - \{6 \prec 5, 7 \prec 6\}$ uma base de conhecimentos idêntica a Δ^*, exceto pelo fato de que as regras de precedência $6 \prec 5$ e $7 \prec 6$, que podem ser deduzidas por especificidade, não estão explicitamente declaradas em Δ^-. Então, o conjunto $\mathcal{E}_{\Delta^-} \cup \mathcal{I}_{\Delta^-}$ não tem ciclos e, portanto, a saída $\Pi^m_{\Delta^-}$ do algoritmo proposto para eliminação de ciclos é o próprio conjunto $\mathcal{E}_{\Delta^-} \cup \mathcal{I}_{\Delta^-}$. Quando o fecho transitivo $\mathcal{M}^c_{\Delta^-}$ de $\Pi^m_{\Delta^-}$ é computado, e as regras de precedência entre regras revogáveis não conflitantes são removidas de $\mathcal{M}^c_{\Delta^-}$, temos um conjunto $\mathcal{M}_{\Delta^-}$ idêntico a $\mathcal{E}_{\Delta^*}$, que é computado a partir de Δ^*.

No sistema desenvolvido neste trabalho, o próprio usuário pode escolher a relação de precedência $\mathfrak{R}$ que ele deseja usar (*explícita*, *implícita* ou *mista*).

2.5. Resolução de Conflitos entre Argumentos

Sejam Δ uma base de conhecimentos e $\mathfrak{R}$ uma relação de precedência sobre regras revogáveis em Δ. Sejam $A = \{\ell_1 : \varphi_1 \to \lambda_1, \ldots, \ell_m : \varphi_m \to \lambda\} \vdash \lambda$ e $A' = \{\ell'_1 : \varphi'_1 \to \lambda'_1, \ldots, \ell'_n : \varphi'_n \to \lambda'\} \vdash \lambda'$ dois argumentos construídos a partir de Δ. Então:

- A e A' são *conflitantes* se e só se há regras revogáveis $\ell_i : \varphi_i \to \lambda_i \in A$ e $\ell'_j : \varphi'_j \to \lambda'_j \in A'$ tais que $\ell_i \lozenge \ell'_j$.
- A *ataca* A' se e só se existe uma regra revogável $\ell'_j : \varphi'_j \to \lambda'_j \in A'$ tal que $\ell_m \lozenge \ell'_j$ e $\ell'_j \prec \ell_m \notin \mathfrak{R}$.

Por exemplo, para os argumentos conflitantes

$A_5 = \{1 : \top \to c, 6 : c \to \neg f\} \vdash \neg f$,
$A_6 = \{1 : \top \to c, 4 : c \to b, 5 : b \to f\} \vdash f$,
$A_7 = \{1 : \top \to c, 2 : \top \to s, 7 : c \wedge s \to f\} \vdash f$ e
$A_8 = \{1 : \top \to c, 3 : \top \to d, 8 : c \wedge d \to \neg f\} \vdash \neg f$,

construídos na Seção 2.3, e $\mathfrak{R} = \{6 \prec 5, 7 \prec 6, 8 \prec 5, 8 \prec 7\}$, temos que A_8 ataca A_6 (mas A_6 não ataca A_8, pois $8 \in A_8$, $5 \in A_6$ e $8 \prec 5 \in \mathfrak{R}$); A_8 ataca A_7 (mas A_7 não ataca A_8, pois $8 \in A_8$, $7 \in A_7$ e $8 \prec 7 \in \mathfrak{R}$); A_7 ataca A_5 (mas A_5 não ataca A_7, pois $7 \in A_7$, $6 \in A_5$ e $7 \prec 6 \in \mathfrak{R}$); e A_5 ataca A_6 (mas A_6 não ataca A_5, pois $6 \in A_5$, $5 \in A_6$ e $6 \prec 5 \in \mathfrak{R}$).

A partir dessa definição de ataque, criamos um algoritmo que transforma um grafo de conflitos num *grafo de ataques*. Por exemplo, a aplicação desse algoritmo no grafo da Figura 3 resulta no grafo da Figura 5.

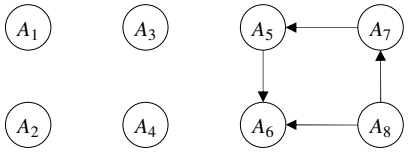


Figura 5 – Ataques entre argumentos de Δ^*.

2.6. Decisão sobre a Coerência dos Argumentos

Após a obtenção do grafo de ataques, a próxima etapa é decidir quais argumentos são coerentes com o conhecimento disponível (a partir do qual eles foram construídos). Para isso, precisamos definir o que é *derrota*.

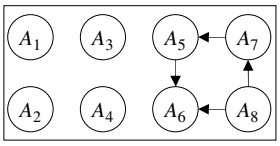
Sejam Δ uma base de conhecimentos e $\mathfrak{R}$ uma relação de precedência sobre regras revogáveis em Δ. Sejam $A = \{\ell_1 : \varphi_1 \to \lambda_1, ..., \ell_m : \varphi_m \to \lambda\} \vdash \lambda$ e $A' = \{\ell'_1 : \varphi'_1 \to \lambda'_1, ..., \ell'_n : \varphi'_n \to \lambda'\} \vdash \lambda'$ dois argumentos construídos a partir de Δ. Então, A *derrota* A' se e só se $(\ell_m \prec \ell'_i) \in \mathfrak{R}$ e:

- Não existe um argumento A'' que ataque A; ou
- Todo argumento A'' que ataca A é derrotado por um outro argumento A''' (construído a partir de Δ).

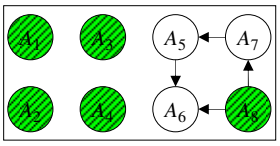
Claramente, essas condições para derrota podem ser facilmente verificadas num grafo de ataques. De fato, criamos um algoritmo de coloração de grafos de conflitos que decide coerência de argumentos do seguinte modo:

- Vértices que não têm predecessores no grafo de ataques, ou que só têm predecessores vermelhos, são coloridos de *verde* (indicando que os argumentos representados por eles são *coerentes*).
- Vértices que têm pelo menos um predecessor verde são coloridos de *vermelho* (indicando que os argumentos representados por eles são *incoerentes*).
- Quando os passos anteriores não puderem mais ser repetidos, os vértices que restarem são coloridos de *amarelo* (indicando que não é possível decidir se os argumentos correspondentes são ou não coerentes).

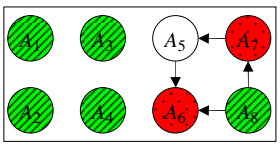
A Figura 6 mostra os passos necessários para colorir o grafo da Figura 5, usando esse algoritmo (nesse caso particular, não há vértices amarelos).



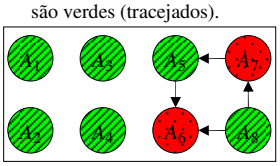
(*a*) Grafo de conflitos sem colorir.



(*b*) Vértices sem predecessores são verdes (tracejados).



(*c*) Vértices com predecessores verdes são vermelhos (pontilhados).

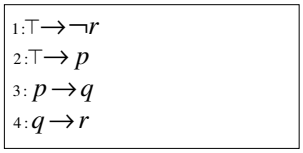


(*d*) Vértices que só têm predecessores vermelhos são verdes.

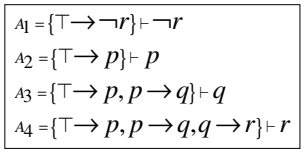
Figura 6 – Decisão de coerência para argumentos de Δ^*.

3. O Protótipo de Sistema Desenvolvido

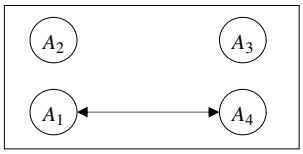
As etapas do processo de argumentação monológica, discutidas na Seção 2 e ilustradas na Figura 7, foram implementadas no sistema de argumentação *Socrates*.



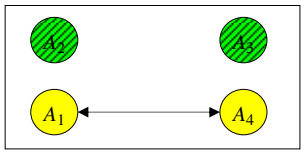
(*a*) Base de conhecimentos.



(*b*) Construção de argumentos.



(*c*) Identificação de ataques.



(*d*) Decisão de coerência.

Figura 7 – Etapas do processo de argumentação monológica.

O sistema *Socrates* é parte de um sistema maior♠, desenvolvido em SWI-PROLOG (*Multi-threaded*, 64 bits, versão 7.2.3) [8]. As principais funcionalidades implementadas no sistema *Socrates* são: editor para criação de uma base de conhecimentos, geração automática de todos os argumentos que podem ser construídos a partir dessa base; criação automática do grafo de ataques entre argumentos conflitantes (a partir da relação de precedência entre regras revogáveis – *explícita*, *implícita* ou *mista* – escolhida pelo usuário) e exibição do grafo de decisão de coerência dos argumentos.

Para usar o sistema, o usuário acessa sua interface gráfica (Figura 8) e cria uma base de conhecimentos ou carrega uma base previamente criada e salva em disco.

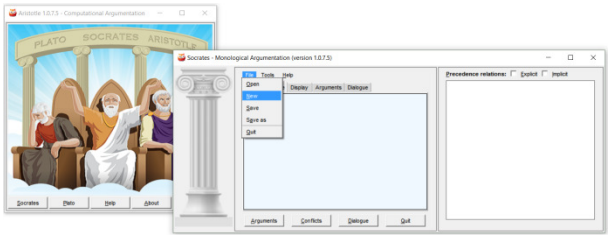


Figura 8 – Interface gráfica do sistema *Socrates*.

Na linguagem de representação usada para criar a base de conhecimentos neste sistema, os símbolos $\top$, $\neg$, $\wedge$, $\to$ e $\prec$ são codificados como `true`, `not`, `and`, `then` e `precedes`, respectivamente. Ademais, regras revogáveis com variáveis funcionam como *esquemas* de regras proposicionais. Por exemplo, a Figura 9 apresenta a codificação da base Δ^-, que foi definida na Seção 2.4.

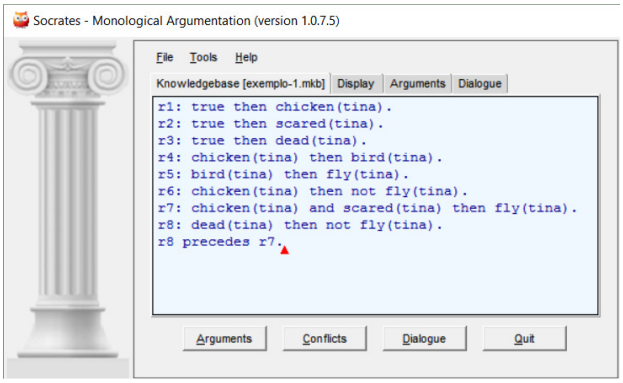


Figura 9 – Base de conhecimentos Δ^- no sistema *Socrates*.

♠ *Disponível em:* `www.ime.usp.br/~slago/aristotle.zip`

Caso não haja erros, quando o usuário clica no botão Arguments, o sistema gera e exibe todos os argumentos possíveis na aba *Arguments*, como mostra a Figura 10.

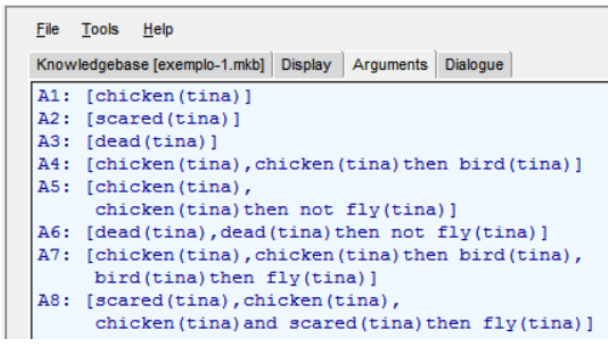
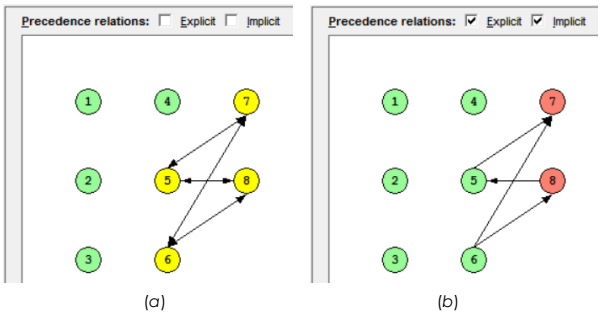


Figura 10 – Aba de exibição de argumentos.

Depois, quando o usuário clica no botão Conflicts, o sistema cria o grafo de ataques entre argumentos conflitantes e exibe o grafo de decisão de coerência (*sem* usar precedências), como mostra a Figura 11(*a*).



(*a*) (*b*)

Figura 11 – Grafos de decisão de coerência.

Para ver o grafo de decisão de coerência (usando precedências), o usuário deve selecionar o tipo de relação de precedência desejada (marcando as opções Explicit e Implicit ao mesmo tempo, ele seleciona uma relação mista). O resultado é exibido logo após a seleção, como mostra a Figura 11(*b*).

Além dessas funcionalidades básicas, o sistema também é capaz de apontar erros sintáticos e semânticos na base digitada pelo usuário, detectar erros de relação de precedência cíclica, mostrar as regras contidas em cada tipo de relação de precedência que o usuário pode escolher e simular um diálogo persuasivo [9].

4. Experimentos Realizados com o Protótipo

Nesta seção, descrevemos alguns dos experimentos que fizemos com o protótipo do sistema implementado e analisamos os resultados obtidos.

4.1. Experimento I: Base Consistente

Este primeiro experimento visa mostrar que não há conflitos entre argumentos que são construídos a partir de uma base de conhecimentos consistente. Portanto, todos eles devem ter conclusões coerentes (ou seja, todos os argumentos devem ser coloridos de verde).

Para tanto, usamos a base de conhecimentos dada na Figura 12. Note que esta base é consistente, pois ela não contém literais complementares.

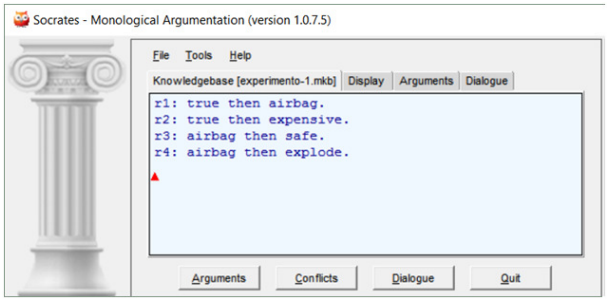


Figura 12 – Base de conhecimentos consistente.

Os argumentos que podem ser construídos a partir dessa base são exibidos na Figura 13 e o grafo de decisão de coerência correspondente é exibido na Figura 14.

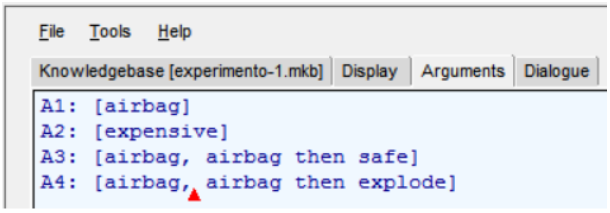


Figura 13 – Argumentos exibidos na aba *Arguments*.

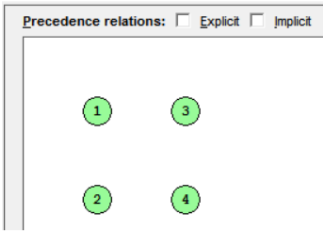


Figura 14 – Grafo de decisão: argumentos coerentes.

Analisando os resultados obtidos, podemos constatar que não há conflitos entre os argumentos gerados a partir da base. Este resultado reforça nossa conjectura de que todos os argumentos construídos a partir de uma base de conhecimentos consistente são sempre coerentes.

4.2. Experimento II: Base Inconsistente

O segundo experimento visa mostrar que, para uma base de conhecimentos inconsistente (Figura 15), se não for usada uma relação de precedência, há pelo menos uma conclusão sobre a qual não se pode decidir a coerência. Note que a base da Figura 15 é inconsistente, pois o par de literais `safe` e `not safe` pode ser derivado dela.

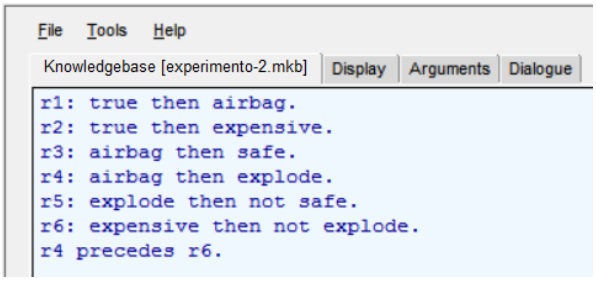


Figura 15 – Base de conhecimentos inconsistente.

Os argumentos gerados a partir dessa base são exibidos na Figura 16. De acordo com as definições na Seção 2.5, temos que os argumentos `A4` e `A5` se atacam e também os argumentos `A3` e `A6`. Por outro lado, o argumento

A5 ataca A6, mas A6 não ataca A5 (pela definição, um argumento *A* só ataca outro *B* se a conclusão de *A* é complementar do consequente de alguma regra em *B*).

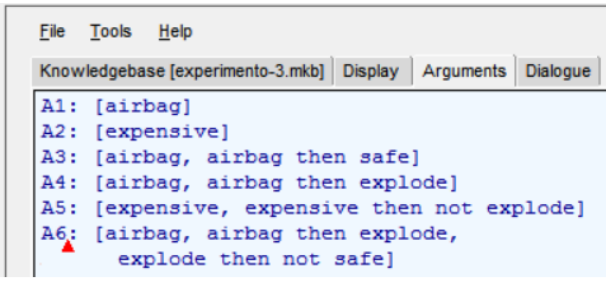


Figura 16 – Argumentos exibidos na aba *Arguments*.

Analisando o grafo de decisão de coerência exibido na Figura 17, vemos que, como esperado, há argumentos conflitantes, cuja coerência não pode ser decidida.

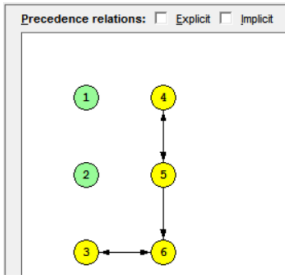


Figura 17 – Grafo de decisão de coerência (sem precedência).

4.3. Experimento III: Uso de Precedências

Este experimento visa mostrar o uso das relações de precedência para a resolução de conflitos. Para isso, usamos a mesma base inconsistente da Figura 15.

Quando a relação de precedência implícita é ativada, o sistema conclui, pelo critério de especificidade, que a regra r3 é mais *concisa* que a regra r5. Assim, o argumento A3 derrota o argumento A6, como mostra a Figura 18(*a*). Note, contudo, que usando apenas a relação de precedência implícita não é possível resolver todos os conflitos entre os argumentos.

Quando a relação de precedência explícita é ativada, a regra r4 precede a regra r3, como declarado na base. Portanto, o argumento A5 deixa de atacar o argumento A6 e o argumento A4 derrota o argumento A5 (Figura 18(*b*)).

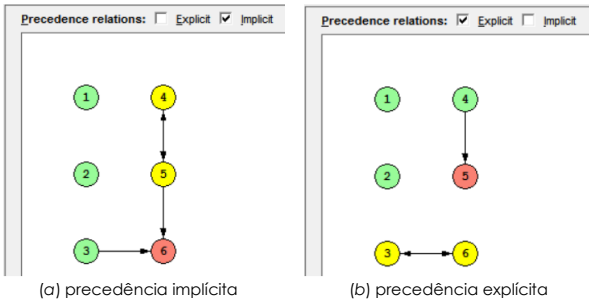


Figura 18 – Grafos de decisão de coerência.

Em ambos os casos, não foi possível solucionar todos os conflitos entre os argumentos. Quando a relação de precedência mista é ativada, obtemos o grafo de decisão de coerência mostrado na Figura 19, em que todos os conflitos entre os argumentos estão resolvidos. Assim, os argumentos A4 e A3 são coerentes e os argumentos A5 e A6 são incoerentes com a base de conhecimentos.

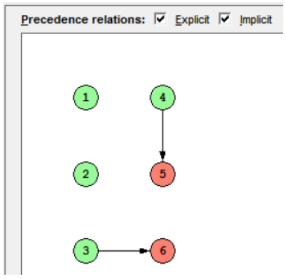


Figura 19 – Grafo de decisão de coerência (precedência *mista*).

5. Conclusões

Este artigo define os fundamentos lógicos a partir dos quais um protótipo do sistema de argumentação computacional monológica, que permite raciocínio coerente na presença de inconsistências, foi desenvolvido em Prolog. O artigo também descreve as principais funcionalidades desse sistema e relata resultados de experimentos.

Os resultados empíricos mostraram que: (*a*) o sistema funciona de acordo com sua especificação lógica e produz resultados que são coerentes com aqueles esperados e (*b*) o uso das relações de precedência propostas neste trabalho (explícita, implícita e mista) é eficaz para resolver conflitos entre argumentos gerados a partir de bases de conhecimentos inconsistentes.

Agradecimentos

Ao CNPq pela bolsa de Iniciação Científica[1](Processo nº 800476/2014-0) e pela bolsa de Produtividade em Pesquisa[3] (Processo nº 102869/2015-4).

Referências Bibliográficas

[1] V. Efstathiou. **Algorithms for Computational Argumentation in Artificial Intelligence**. Ph.D. Thesis. University College London. UK, 2010.

[2] A. Carnielli; J. Marcos. **Ex contradictione non sequitur quodlibet**. In II Annual Conference on Reasoning and Logic, Bucharest, RO, 89–109, 2001.

[3] D. Walton. **Fundamentals of Critical Argumentation**. Cambridge University Press, UK, 2006.

[4] P. Dung. **On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games**, Artificial Intelligence, v. 77, p. 321-357, Elsevier, 1994.

[5] S. Russell; P. Norvig. **Artificial intelligence: a modern approach**, 3rd ed., Prentice-Hall, 2010.

[6] R. Kowalski. **Predicate Logic as a Programming Language**. Information Processing, North Holland Publishing Co., 569–574, 1974.

[7] S. Benferhat. **Computing specificity in default reasoning, Algorithms for Uncertainty and Defeasible Reasoning**, 5:147-177, Kluwer Publishers, 2001.

[8] I. Bratko. **Prolog Programming for Artificial Intelligence**, 4th ed., Pearson, 2011.

[9] S. L. Pereira; L. F. Z. Santos; L. N. Lira. **A Dialogue System for Coherent Reasoning with Inconsistent Knowledge Bases**, Journal of Computer and Communications, vol. 3., p. 11-19, 2015.