MAC 5701 - Tópicos em Ciência da Computação
Relatório de Estudo Dirigido

Implementações de Tableaux para Raciocínio Aproximado

Wagner Dias
dias@ime.usp.br

27 de novembro de 2001

Sumário

1 Introdução

Atualmente, uma das maiores questões no campo da Teoria da Computação é o problema $\mathcal{P} = \mathcal{NP}$.

$\mathcal{P}$ é uma *classe de complexidade*, definida como o conjunto de *problemas de decisão* (problemas que admitem resposta do tipo *sim* ou *não*) que podem ser resolvidos em tempo polinomial — $O(n^k)$ para alguma constante k.

$\mathcal{NP}$ é a classe de complexidade dos problemas de decisão cuja resposta pode ser verificada em tempo polinomial através de um *certificado*.

Existe uma classe de problemas — a classe dos problemas $\mathcal{NP}$*-completos* — com a propriedade de que, se qualquer um dos problemas puder ser resolvido em tempo polinomial, então todo problema de $\mathcal{NP}$ é polinomial, ou seja, $\mathcal{P} = \mathcal{NP}$.

Mas até hoje nenhum problema $\mathcal{NP}$-completo tem solução polinomial conhecida, o que leva a maioria dos pesquisadores de Teoria da Computação a crer que $\mathcal{P} \neq \mathcal{NP}$, ou seja, que há problemas em $\mathcal{NP}$ de natureza inerentemente superpolinomial.

Um resultado conhecido é a prova de que o problema SAT é $\mathcal{NP}$-completo [CLR90], ou seja, o problema de determinar se uma fórmula da lógica proposicional clássica é satisfatível (ou válida) é difícil.

De fato, não se conhece nenhum método de prova para a lógica proposicional clássica que tenha tempo polinomial.

O método de *tabela da verdade* de Wittgenstein é imediatamente implementável em uma máquina, mas é exponencial: para uma fórmula com n átomos distintos, a tabela da verdade tem 2^n linhas.

O método de tableaux analíticos de Smullyan é uma alternativa também facilmente implementável. Mas há fórmulas (as *fórmulas "gordas"* — fórmulas que têm tamanho total grande comparado com o número de símbolos proposicionais distintos) cuja prova gera um número grande de ramos no tableau analítico comparado ao número de linhas da tabela da verdade correspondente. Em outras palavras, *tableaux analíticos não p-simulam*[1] *tabelas da verdade*.

Logicamente, as tabelas da verdade não p-simulam os tableaux analíticos,

[1] [D'A90] e [D'A92] descrevem a relação (de ordem parcial) de *p-simulação* como um modo de se medir a *complexidade relativa* de sistemas de prova. Sistemas que se p-simulam mutuamente podem ser considerados essencialmente de mesma complexidade. Um sistema $\mathbf{S}$ que p-simula um sistema $\mathbf{S}'$ mas o contrário não é verdade, pode ser considerado essencialmente mais eficiente que $\mathbf{S}'$.

e portanto os dois métodos são incomparáveis — um não é mais eficiente que o outro.

D'Agostino [D'A90] aponta uma razão para esse fenômeno como sendo uma inadequação das regras de ramificação do tableau analítico que afeta a complexidade das provas no caso geral. Ele propõe o sistema de tableaux **KE**, e mostra que **KE** p-simula tableau analítico, mas o inverso não é verdade, ou seja, o sistema **KE** é essencialmente mais eficiente que o sistema de tableaux analíticos.

Uma outra abordagem para resolver o problema da satisfatibilidade de fórmulas é o *raciocínio aproximado*.

O princípio básico do raciocínio aproximado é a tolerância a inconsistências. Quando se permite a presença de inconsistências em partes que são irrelevantes ao problema que se quer resolver pode-se ganhar eficiência, dado que a manutenção do estado consistente é computacionalmente cara.

O raciocínio aproximado apresenta vantagens particularmente quando se está num cenário de recursos limitados: se o tempo é limitado, pode-se usar a informação da resposta até então obtida como uma aproximação da resposta exata do problema original.

Cadoli e Schaerf [SC95] propuseram uma técnica de raciocínio aproximado. Em seu método, eles definem duas lógicas que servem como um limitante superior (incorreto e completo) e um limitante inferior (correto e incompleto) para a lógica clássica. O método busca respostas aproximadas nessas duas lógicas que satisfazem as seguintes propriedades: (a) dão informações semanticamente claras sobre o problema a cada passo de aproximação; (b) são mais fáceis de computar que a resposta do problema original; e (c) podem ser melhoradas, e eventualmente convergem para a resposta correta.

O método faz uma restrição à linguagem — fórmulas na forma clausal —, de modo a tornar viável a computação das respostas aproximadas.

Finger e Wassermann [FW01] propuseram um método que generaliza o raciocínio aproximado de Cadoli e Schaerf, eliminando a restrição a cláusulas. Eles estendem a semântica da lógica S_3 usada no método de Cadoli e Schaerf para toda a lógica proposicional, e propõem o método de tableaux KE-S_3 para essa lógica — baseado nos tableaux KE de D'Agostino.

O objetivo deste trabalho é implementar o método de Tableaux Analíticos de Smullyan, o método de Tableaux KE de D'Agostino e o Método de Tableaux KE-S_3 e fazer um teste comparativo dos métodos utilizando benchmarks como DIMACS [JT96] e SATLIB [HS00].

Na Seção 2, serão descritos os fundamentos básicos da lógica clássica proposicional em seus aspectos sintáticos e semânticos.

Na Seção 3, serão descritos o método de Tableaux Analíticos de Smullyan e o método de Tableaux KE de D'Agostino, ambos para a lógica proposicional clássica.

Na Seção 4, serão descritos o método de raciocínio aproximado de Cadoli e Schaerf e o método de Tableaux KE-S_3 de Finger e Wassermann.

Finalmente, a Seção 5 apresenta as considerações finais.

2 Lógica Proposicional Clássica

No decorrer de todo o texto, a não ser que seja explicitamente dito o contrário, usaremos as convenções a seguir.

2.1 Sintaxe

Os *conectivos lógicos* são os quatro símbolos:

1. $\neg$: *negação*;
2. $\wedge$: *conjunção*;
3. $\vee$: *disjunção*;
4. $\rightarrow$: *implicação*.

O conectivo da negação é chamado de *conectivo unário*. Os três últimos são chamados de *conectivos binários*.

As *variáveis proposicionais* (ou *variáveis* ou *átomos*) são os símbolos pertencentes ao conjunto enumerável $\mathcal{P} = \{p_1, p_2, \ldots\}$. Usaremos letras minúsculas $(p, q, r, s, \ldots)$ para denotar elementos de $\mathcal{P}$.

Os símbolos (e) são chamados de *parênteses* e são usados para pontuação.

Uma *fórmula* é um elemento que pode ser definido recursivamente a partir do seguinte conjunto de regras:

F_0: Todo elemento p de $\mathcal{P}$ é uma fórmula;

F_1: Se A é uma fórmula, então $\neg A$ é uma fórmula;

F_2: Se A, B são fórmulas, então $(A \wedge B)$ é uma fórmula;

F_3: Se A, B são fórmulas, então $(A \vee B)$ é uma fórmula;

F_4: Se A, B são fórmulas, então $(A \rightarrow B)$ é uma fórmula.

Usaremos letras maiúsculas $(A, B, C, \ldots)$ para denotar fórmulas e letras maiúsculas gregas $(\Gamma, \Delta, \ldots)$ para denotar conjuntos de fórmulas. O conjunto de todas as fórmulas será denotado por $\mathcal{L}$.

Os parênteses poderão ser omitidos quando permitido pelo contexto.

As *subfórmulas imediatas* de uma fórmula A são dadas pelas regras (B e C são fórmulas):

I_0: Se $A \in \mathcal{P}$, então A não tem subfórmulas imediatas;

I_1: Se A é da forma $\neg B$, então B é a única subfórmula imediata de A;

I_2: Se A é da forma $(B \wedge C)$, então B e C são as únicas subfórmulas imediatas de A;

I_3: Se A é da forma $(B \vee C)$, então B e C são as únicas subfórmulas imediatas de A;

I_4: Se A é da forma $(B \rightarrow C)$, então B e C são as únicas subfórmulas imediatas de A.

As *subfórmulas* de uma fórmula A são dadas pelas regras:

S_1: Se X é uma subfórmula imediata de A ou $X = A$, então X é uma subfórmula de A;

S_2: Se X é uma subfórmula de Y e Y é uma subfórmula de A, então X é uma subfórmula de A.

2.2 Semântica

Um *valor-verdade* é um elemento do conjunto $V = \{0, 1\}$. Os elementos 0 e 1 são chamados, respectivamente, de *falso* e *verdadeiro*.

Uma *valoração* é uma função $v : \Gamma \rightarrow V$. Deste modo, dizemos que $A \in \Gamma$ é *verdadeira* sob v se $v(A) = 1$, e *falsa* sob v caso contrário.

Uma valoração $v : \mathcal{L} \rightarrow V$ é uma *valoração booleana* se, para toda fórmula $A, B \in \mathcal{L}$:

B_1: $v(\neg A) = 1$ sse $v(A) = 0$;

B_2: $v(A \wedge B) = 1$ sse $v(A) = 1$ e $v(B) = 1$;

B_3: $v(A \vee B) = 0$ sse $v(A) = 0$ e $v(B) = 0$;

B_4: $v(A \rightarrow B) = 0$ sse $v(A) = 1$ e $v(B) = 0$.

Uma fórmula A é uma *tautologia* (denotamos $\models A$) se A é verdadeira sob todas as valorações booleanas.

Uma valoração v satisfaz um conjunto de fórmulas Γ se toda fórmula de Γ é verdadeira sob v.

Um conjunto de fórmulas Γ *implica logicamente* uma fórmula A (denotamos $\Gamma \models A$) se A é verdadeira sob toda valoração booleana que satisfaça Γ.

3 Métodos de Tableaux

Nesta seção apresentaremos dois métodos de resolução baseados em tableaux: o método de tableaux analíticos de Smullyan, e o método de tableaux KE de D'Agostino.

3.1 Tableaux Analíticos

O *tableau analítico* é um método de prova proposto por Smullyan em [Smu68] baseado em árvores de *fórmulas rotuladas*. A idéia básica do método é tentar provar $A_1, A_2, \ldots, A_n \vdash B$, através da expansão da árvore associada.

Uma *fórmula rotulada* é uma expressão do tipo TA ou FA, onde A é uma fórmula.

Dada uma valoração, TA é *verdadeira* se A é verdadeira, e $false$, caso contrário. FA é verdadeira se A é falsa, e $false$, caso contrário.

O *conjugado* de uma fórmula rotulada é a fórmula com o rótulo oposto. Deste modo, TA é o conjugado de FA, e vice-versa.

O método consiste em, dado um *tableau inicial* para $A_1, A_2, \ldots, A_n \vdash B$, estendê-lo através das *regras de expansão*, até que se chegue num estado onde se possa decidir se $A_1, A_2, \ldots, A_n \vdash B$ ou se $A_1, A_2, \ldots, A_n \nvdash B$.

O tableau inicial para $A_1, A_2, \ldots, A_n \vdash B$ é a árvore de um ramo onde cada nó é uma das fórmulas rotuladas $TA_1, TA_2, \ldots, TA_n, FB$.

As regras de expansão do tableau estendem o tableau inicial. As regras de expansão de tableaux analíticos são apresentadas na Tabela 1.

Para cada regra de expansão, as fórmulas que aparecem acima da linha horizontal são as *premissas* da regra. As fórmulas que aparecem abaixo da linha horizontal são as *conclusões* da regra.

Sejam X, Y, Z fórmulas rotuladas. As regras da forma

$$\frac{X}{Y} \qquad \frac{X}{\begin{matrix} Y \\ Z \end{matrix}}$$

são denominadas *regras do tipo* α.

As regras da forma

$$\frac{X}{Y|Z}$$

são denominadas *regras do tipo* β.

$$\frac{T\neg A}{FA}\ (T\neg) \qquad \frac{F\neg A}{TA}\ (F\neg)$$

$$\frac{T(A \wedge B)}{\begin{array}{c}TA\\TB\end{array}}\ (T\wedge) \qquad \frac{F(A \wedge B)}{FA|FB}\ (F\wedge)$$

$$\frac{T(A \vee B)}{TA|TB}\ (T\vee) \qquad \frac{F(A \vee B)}{\begin{array}{c}FA\\FB\end{array}}\ (F\vee)$$

$$\frac{T(A \to B)}{FA|TB}\ (T \to) \qquad \frac{F(A \to B)}{\begin{array}{c}TA\\FB\end{array}}\ (F \to)$$

Tabela 1: Regras de expansão de tableaux analíticos

Uma regra pode ser aplicada a um ramo do tableau se este ramo contiver a premissa da regra (um ramo do tableau é um caminho da raiz até uma folha).

A aplicação de uma regra do tipo α num ramo do tableau consiste em adicionar as conclusões da regra neste ramo.

A aplicação de uma regra do tipo β num ramo do tableau consiste em *bifurcar* o ramo e, em cada ramo resultante, adicionar a conclusão correspondente.

Se um ramo do tableau contiver duas fórmulas conjugadas (ou seja, se contiver TA e FA para alguma fórmula A), dizemos que o ramo está *fechado*.

Se todos os ramos do tableau estiverem fechados, dizemos que o tableau está fechado, e neste caso, $A_1, A_2, \ldots, A_n \vdash B$.

Se, por outro lado, o tableau não estiver fechado (estiver *aberto*), e não houver nenhuma regra aplicável, então $A_1, A_2, \ldots, A_n \nvdash B$.

Exemplo 3.1 O tableau da Figura 1 foi obtido a partir do tableau inicial (composto apenas da fórmula 1) através de aplicações sucessivas das regras de expansão. As fórmulas 2 e 3 foram obtidas de 1 através da regra $(F \to)$. As fórmulas 4 e 5 foram obtidas de 2 através da regra $(T\wedge)$. As fórmulas

6 e 7 foram obtidas de 3 através da regra $(F \rightarrow)$. As fórmulas 8 e 9 foram obtidas de 4 através da regra $(T \rightarrow)$. As fórmulas 10 e 11 foram obtidas de 5 através da regra $(T \rightarrow)$. O símbolo $\times$ indica que o ramo correspondente do tableau está fechado. Deste modo, o ramo que termina com a fórmula 8 está fechado, porque contém as fórmulas conjugadas 6 e 8. O ramo que termina com a fórmula 10 está fechado, porque contém as fórmulas conjugadas 9 e 10. O ramo que termina com a fórmula 11 está fechado, porque contém as fórmulas conjugadas 7 e 11. Logo, o tableau está fechado, e portanto,

$$\vdash ((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

□

$$\begin{array}{c}
F(((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r))\ (1) \\
T((p \rightarrow q) \wedge (q \rightarrow r))\ (2) \\
F(p \rightarrow r)\ (3) \\
T(p \rightarrow q)\ (4) \\
T(q \rightarrow r)\ (5) \\
Tp\ (6) \\
Fr\ (7) \\
\begin{array}{c|c}
\begin{array}{c} Fp\ (8) \\ \times \\ \\ \end{array} &
\begin{array}{c} Tq\ (9) \\ \begin{array}{c|c} \begin{array}{c} Fq\ (10) \\ \times \end{array} & \begin{array}{c} Tr\ (11) \\ \times \end{array} \end{array} \end{array}
\end{array}
\end{array}$$

Figura 1: Tableau analítico para $\vdash ((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$.

Em [Smu68], Smullyan provou a correção e a completude do método de tableaux analíticos, ou seja, se $\vdash$ representa a relação de conseqüência estabelecida pelo método de tableaux analíticos, e $\models$ a relação de conseqüência lógica (semântica), então

$$A_1, A_2, \ldots, A_n \vdash B \text{ se e somente se } A_1, A_2, \ldots, A_n \models B$$

3.2 Tableaux KE

D'Agostino propôs em [D'A90] o método de Tableaux KE como uma melhoria do método de tableaux analíticos de Smullyan, no sentido computacional.

Tableaux KE *p-simulam* tableaux analíticos, mas tableaux analíticos não p-simulam tableaux KE [2].

Os tableaux KE são semelhantes aos tableaux analíticos, no sentido em que também utilizam fórmulas rotuladas, apresentam regras de expansão com premissas e conclusões, e têm as mesmas regras para a construção do tableau inicial e para o fechamento do tableau.

As regras de expansão do tableau KE são apresentadas na Tabela 2.

$\dfrac{T\neg A}{FA}$	$(T\neg)$	$\dfrac{F\neg A}{TA}$	$(F\neg)$		
$\dfrac{T(A \wedge B)}{\begin{array}{c}TA\\TB\end{array}}$	$(T\wedge)$	$\dfrac{\begin{array}{c}F(A \wedge B)\\TA\end{array}}{FB}$	$(F\wedge_1)$	$\dfrac{\begin{array}{c}F(A \wedge B)\\TB\end{array}}{FA}$	$(F\wedge_2)$
$\dfrac{F(A \vee B)}{\begin{array}{c}FA\\FB\end{array}}$	$(F\vee)$	$\dfrac{\begin{array}{c}T(A \vee B)\\FA\end{array}}{TB}$	$(T\vee_1)$	$\dfrac{\begin{array}{c}T(A \vee B)\\FB\end{array}}{TA}$	$(T\vee_2)$
$\dfrac{F(A \rightarrow B)}{\begin{array}{c}TA\\FB\end{array}}$	$(F \rightarrow)$	$\dfrac{\begin{array}{c}T(A \rightarrow B)\\TA\end{array}}{TB}$	$(T \rightarrow_1)$	$\dfrac{\begin{array}{c}T(A \rightarrow B)\\FB\end{array}}{FA}$	$(T \rightarrow_2)$
$\dfrac{}{TA \mid FA}$	(PB)				

Tabela 2: Regras de expansão de tableaux KE

Os conectivos lógicos $\wedge, \vee$ e $\rightarrow$ têm três regras de expansão: uma com uma premissa e duas conclusões — regras do tipo α —, e duas com duas

[2] Em outras palavras, para toda prova em tableau analítico de tamanho n, existe uma prova em tableau KE com tamanho polinomial em n, mas existe pelo menos uma prova em tableau KE de tamanho n cuja prova correspondente em tableau analítico tem tamanho superpolinomial em n.

premissas e uma conclusão — regras do tipo β. Nas regras β, a primeira premissa é a *premissa primária* e a segunda é a *premissa secundária*[3]. O conectivo unário $\neg$ tem duas regras do tipo α.

A regra (PB) permite que, a qualquer momento, o tableau seja bifurcado em ramos com as duas fórmulas conjugadas TA e FA. Isto corresponde ao *Princípio da Bivalência*, ou seja, o fato de uma fórmula poder assumir apenas um dos valores verdade: 0 (F) ou 1 (T). Note que esta é a única regra que permite a bifurcação de ramos do tableau.

O uso da regra (PB) no sistema KE deve se restringir a *aplicações analíticas*, ou seja, a fórmula A sobre a qual a regra se aplica deve servir como premissa secundária de alguma fórmula β não analisada. Isto garante a propriedade (também presente no tableau analítico) de que toda regra de expansão adiciona uma subfórmula de alguma fórmula mais complexa ocorrente no tableau.

Exemplo 3.2 Vamos refazer o Exemplo 3.1 usando o método de tableaux KE. O tableau da Figura 2 foi obtido a partir do tableau inicial (composto apenas da fórmula 1) através de aplicações sucessivas das regras de expansão. As fórmulas 2 e 3 foram obtidas de 1 através da regra $(F \rightarrow)$. As fórmulas 4 e 5 foram obtidas de 2 através da regra $(T\wedge)$. As fórmulas 6 e 7 foram obtidas de 3 através da regra $(F \rightarrow)$. A fórmula 8 foi obtida usando-se a regra $(T \rightarrow_1)$ com a fórmula 4 como premissa primária e a fórmula 6 como premissa secundária. A fórmula 9 foi obtida usando-se a regra $(T \rightarrow_1)$ com a fórmula 5 como premissa primária e a fórmula 8 como premissa secundária. Neste ponto, o tableau está fechado, pois seu único ramo contém as fórmulas conjugadas 7 e 9. Portanto,

$$\vdash ((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$$

Note que, no Exemplo 3.1, foram geradas 11 fórmulas e houve duas bifurcações. Neste caso, foram geradas 9 fórmulas e o tableau fechou sem bifurcar nenhuma vez. $\square$

Exemplo 3.3 Neste exemplo, queremos provar a consistência do conjunto $\{\neg p \rightarrow q, \neg p \rightarrow \neg q, p \rightarrow r, p \rightarrow \neg r\}$. Para isso construímos o tableau da

[3]Por simplicidade, chamaremos as fórmulas que podem ser premissas de alguma regra do tipo α de *fórmulas* α, e as fórmulas que podem ser premissas primárias de alguma regra do tipo β de *fórmulas* β.

$$
\begin{array}{c}
F(((p \to q) \land (q \to r)) \to (p \to r))\ (1) \\
T((p \to q) \land (q \to r))\ (2) \\
F(p \to r)\ (3) \\
T(p \to q)\ (4) \\
T(q \to r)\ (5) \\
Tp\ (6) \\
Fr\ (7) \\
Tq\ (8) \\
Tr\ (9) \\
\times
\end{array}
$$

Figura 2: Tableau KE para $\vdash ((p \to q) \land (q \to r)) \to (p \to r)$.

Figura 3. Note que é impossível fechar o tableau sem o uso da regra (PB), pois a partir do tableau inicial não há nenhuma regra α ou β que possa ser utilizada.

$$
\begin{array}{c}
T\neg p \to q \\
T\neg p \to \neg q \\
Tp \to r \\
Tp \to \neg r \\
\begin{array}{c|c}
T\neg p & F\neg p \\
Tq & Tp \\
T\neg q & Tr \\
Fq & T\neg r \\
\times & Fr \\
 & \times
\end{array}
\end{array}
$$

Figura 3: Tableau KE para verificar a consistência do conjunto de fórmulas $\{\neg p \to q, \neg p \to \neg q, p \to r, p \to \neg r\}$.

[D'A90] apresenta a prova de correção e completude do método de tableaux KE.

4 Raciocínio Aproximado

Nesta seção, apresentaremos o método de raciocínio aproximado de Cadoli e Schaerf, e o método de tableaux KE para a lógica S_3 (ou tableaux KE-S_3).

4.1 O método de raciocínio aproximado de Cadoli e Schaerf

Cadoli e Schaerf propuseram em [SC95] um método de *raciocínio aproximado*. Esse método aproxima a lógica clássica por duas outras lógicas: a lógica S_1, que serve como um limitante superior, e a lógica S_3, que serve como um limitante inferior da lógica clássica. O método se restringe a fórmulas na forma clausal (ou seja, conjunções de disjunções de literais).

Ambas as lógicas S_1 e S_3 são baseadas em um *conjunto-contexto* S, que é um conjunto de átomos. Os elementos de S são os elementos que se comportam classicamente na lógica. Deste modo, as lógicas S_1 e S_3 se aproximam da lógica clássica conforme o conjunto S aumenta.

Uma valoração v em S_1 é definida como se segue:

- $v(p) \neq v(\neg p)$, se $p \in S$;
- $v(p) = v(\neg p) = 0$, se $p \notin S$.

Para um conjunto de fórmulas Γ e uma fórmula A, dizemos que Γ implica logicamente A em S_1 (e escrevemos $\Gamma \models_S^1 A$) se toda valoração em S_1 que satisfaça Γ também satisfaz A.

Pode-se provar que $\models_S^1$ é completa e incorreta com relação a $\models$. Ou seja, para todo Γ e para todo A,

$$\Gamma \not\models_S^1 A \Rightarrow \Gamma \not\models A$$

Uma valoração v em S_3 é definida como se segue:

- $v(p) \neq v(\neg p)$, se $p \in S$;
- $v(p) \neq v(\neg p)$ ou $v(p) = v(\neg p) = 1$, se $p \notin S$.

Para um conjunto de fórmulas Γ e uma fórmula A, dizemos que Γ implica logicamente A em S_3 (e escrevemos $\Gamma \models_S^3 A$) se toda valoração em S_3 que satisfaça Γ também satisfaz A.

Pode-se provar que $\models^3_S$ é correta e incompleta com relação a $\models$. Ou seja, para todo Γ e para todo A,

$$\Gamma \models^3_S A \Rightarrow \Gamma \models A$$

Note que, se $S = \mathcal{P}$, então $v(p) \neq v(\neg p)$ para todo $p \in \mathcal{P}$, ou seja, tanto S_1 como S_3 coincidem com a lógica clássica.

O raciocínio aproximado de Cadoli e Schaerf baseia-se em simplificações dos conjuntos de fórmulas, que utilizam as propriedades das lógicas S_1 e S_3 descritas acima.

Lema 4.1 (Cadoli & Schaerf [SC95]) *Seja simplifica-1*(Γ, S) *o resultado de apagar todos os literais do conjunto de fórmulas* Γ *que mencionam átomos fora do conjunto* S. Γ *é* S_1*-satisfatível sse simplifica-1*(Γ, S) *é classicamente satisfatível.*

Teorema 4.1 (Cadoli & Schaerf [SC95]) *Seja* $A = A_S \vee A_{\overline{S}}$, *onde os átomos de* A_S *estão todos em* S *e nenhum dos átomos de* $A_{\overline{S}}$ *está em* S. $\Gamma \models^1_S A$ *sse* $\Gamma \cup \{\neg A_S\}$ *não é* S_1*-satisfatível.* [4]

Lema 4.2 (Cadoli & Schaerf [SC95]) *Seja simplifica-3*(Γ, S) *o resultado de apagar todas as cláusulas de* Γ *que contêm pelo menos um átomo fora de* S. Γ *é* S_3*-satisfatível sse simplifica-3*(Γ, S) *é classicamente satisfatível.*

Teorema 4.2 (Cadoli & Schaerf [SC95]) *Seja* A *tal que os átomos de* A *estejam todos em* S. $\Gamma \models^3_S A$ *sse* $\Gamma \cup \{\neg A\}$ *não é* S_3*-satisfatível.*

Um algoritmo para decidir se $\Gamma \models A$ baseado no raciocínio aproximado pode ser feito da seguinte maneira: começando com S igual ao conjunto dos átomos de A e passo a passo adicionando átomos a S, e efetuando as simplificações *simplifica-1* e *simplifica-3*, parando com o menor S tal que $\Gamma \not\models^1_S A$ ou $\Gamma \models^3_S A$.

Um tal algoritmo apresenta certas desvantagens:

- O método é restrito a fórmulas na forma clausal, devido às simplificações *simplifica-1* e *simplifica-3*;

[4] Isto só pode ser feito porque A_S se comporta classicamente, e assim podemos calcular sua negação em forma clausal.

- Não se tem uma heurística para determinar qual átomo será adicionado ao conjunto S a cada passo.

Exemplo 4.1 Seja $\Gamma = \{\neg p \vee q, \neg q \vee r\}$ e $A = \neg p \vee r$. Queremos decidir se $\Gamma \models A$.

1. Faça $S \leftarrow \{p, r\}$. Temos:
   - *simplifica-1*$(\Gamma \cup \neg A_S, S) = \{\neg p, r, p, \neg r\}$, e portanto $\Gamma \cup \neg A_S$ não é S_1-satisfatível, ou seja, $\Gamma \models_S^1 A$.
   - *simplifica-3*$(\Gamma \cup \neg A, S) = \{p, \neg r\}$, e portanto $\Gamma \cup \neg A$ é S_3-satisfatível, ou seja, $\Gamma \not\models_S^3 A$.
2. Faça $S \leftarrow S \cup \{q\}$. Temos:
   - *simplifica-1*$(\Gamma \cup \neg A_S, S) = \{\neg p \vee q, \neg q \vee r, p, \neg r\}$, e portanto $\Gamma \cup \neg A_S$ não é S_1-satisfatível, ou seja, $\Gamma \models_S^1 A$.
   - *simplifica-3*$(\Gamma \cup \neg A, S) = \{\neg p \vee q, \neg q \vee r, p, \neg r\}$, e portanto $\Gamma \cup \neg A$ não é S_3-satisfatível, ou seja, $\Gamma \models_S^3 A$, donde concluímos que $\Gamma \models A$. $\square$

4.2 Tableaux KE-S_3

Finger and Wassermann propuseram em [FW01] o método de tableaux KE-S_3 como uma generalização do método de Cadoli e Schaerf para raciocínio aproximado.

As regras de expansão do tableau KE-S_3 são apresentadas na Tabela 3.

O funcionamento deste método é muito semelhante ao do método de tableaux KE. Neste método, deve-se manter um conjunto S, que no tableau inicial é igual ao conjunto vazio. Toda vez que se aplica a regra $(T\neg)$ sobre a fórmula A, deve-se adicionar os átomos de A ao conjunto S.

Exemplo 4.2 Vamos decidir se $\Gamma \vdash A$, onde Γ e A são os mesmos do exemplo 4.1. O tableau correspondente a essa prova é o da Figura 4. O tableau inicial é composto pelas três primeiras fórmulas, e o conjunto S é inicializado com vazio.

As fórmulas (4) e (5) são obtidas de (3) usando a regra $(F\vee)$. A fórmula (6) é obtida de (4) usando a regra $(F\neg)$. A fórmula (7) é obtida de (1) e (4) usando a regra $(T\vee_1)$. A fórmula (8) é obtida de (2) e (5) usando a

$\dfrac{T\neg A}{FA}$ se *átomos(A)* $\subseteq S$	$(T\neg)$			$\dfrac{F\neg A}{TA}$	$(F\neg)$
$\begin{array}{c} T(A \wedge B) \\ \hline TA \\ TB \end{array}$	$(T\wedge)$	$\begin{array}{c} F(A \wedge B) \\ TA \\ \hline FB \end{array}$	$(F\wedge_1)$	$\begin{array}{c} F(A \wedge B) \\ TB \\ \hline FA \end{array}$	$(F\wedge_2)$
$\begin{array}{c} F(A \vee B) \\ \hline FA \\ FB \end{array}$	$(F\vee)$	$\begin{array}{c} T(A \vee B) \\ FA \\ \hline TB \end{array}$	$(T\vee_1)$	$\begin{array}{c} T(A \vee B) \\ FB \\ \hline TA \end{array}$	$(T\vee_2)$
$\begin{array}{c} F(A \to B) \\ \hline TA \\ FB \end{array}$	$(F \to)$	$\begin{array}{c} T(A \to B) \\ TA \\ \hline TB \end{array}$	$(T \to_1)$	$\begin{array}{c} T(A \to B) \\ FB \\ \hline FA \end{array}$	$(T \to_2)$
$\begin{array}{c} \\ \hline TA \mid FA \end{array}$	(PB)				

Tabela 3: Regras de expansão de tableaux KE-S_3

regra ($T\vee_2$). A fórmula (9) é obtida de (8) usando a regra ($T\neg$) e fazendo $S \leftarrow S \cup \{q\}$.

Pela figura, vemos que o tableau fecha com $S = \{q\}$. $\square$

$$
\begin{array}{ccc}
T\neg p \vee q & S = \{\} & (1) \\
T\neg q \vee r & & (2) \\
F\neg p \vee r & & (3) \\
F\neg p & & (4) \\
Fr & & (5) \\
Tp & & (6) \\
Tq & & (7) \\
T\neg q & & (8) \\
Fq & S = \{q\} & (9) \\
\times & &
\end{array}
$$

Figura 4: Tableau KE-S_3 para $\neg p \vee q, \neg q \vee r \vdash \neg p \vee r$.

Em [FW01] são provados os teoremas de correção e completude do método de tableaux KE-S_3.

5 Considerações Finais

Neste projeto, será feita a implementação dos três métodos de tableaux descritos nas seções acima. Para isso, será criado um *arcabouço* (do inglês, *framework*) para métodos de tableau.

A função do arcabouço é implementar o funcionamento *genérico* de um método de tableaux, isto é, o que é comum a todos os métodos. Depois de implementado o arcabouço, este será *instanciado* em cada método específico. Na instanciação, tudo o que é específico de cada método é acoplado ao arcabouço — o conjunto de regras, por exemplo.

Portanto, será implementado um arcabouço para métodos de tableau, que será instanciado três vezes, uma vez para cada um dos métodos de tableaux descritos acima.

As implementações serão testadas utilizando benchmarks conhecidos como o DIMACS [JT96] e o SATLIB [HS00].

Com este projeto, espera-se através dos benchmarks, fazer uma comparação entre os métodos de tableaux para a lógica proposicional clássica e para raciocínio aproximado, explorando as vantagens e desvantagens de cada método em cada situação de uso.

Referências

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, McGraw-Hill Book Company, 1990.

[D'A90] Marcello D'Agostino. Investigations into the complexity of some propositioal calculi. Master's thesis, Oxford University Computing Laboratory, Programming Research Group, 1990.

[D'A92] Marcello D'Agostino. Are tableaux an improvement on truth-tables? — cut-free proofs and bivalence. *Journal of Logic, Language and Information*, 1:235–252, 1992.

[FW01] Marcelo Finger and Renata Wassermann. Tableaux for approximate reasoning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'2001)*, Seattle, Washington, USA, 2001.

[HS00] Holger H. Hoos and Thomas Stützle. Satlib: An online resource for research on sat. In I. P. Gent, H. v. Maaren, and T. Walsh, editors, *SAT 2000*, pages 283–292. IOS Press, 2000. http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/.

[JT96] D. S. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* American Mathematical Society, 1996. ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/.

[SC95] Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.

[Smu68] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.