Revisão de crenças no fragmento universal da CTL usando verificação de modelos limitada

Aluno: Bruno Vercelino da Hora
{bruno.hora@usp.br}
Orientador: Marcelo Finger
{mfinger@ime.usp.br}

Instituto de Matemática e Estatística - USP

Novembro de 2009

Sumário

Introdução

Motivação

- Levantamento e análise de requisitos
- Erros ou modificações na especificação geram atrasos no projeto
- Conforme a complexidade e o tamanho do software aumentam, maior a dificuldade em revisar a especificação
- Um processo automático diminui os custos e o tempo desta operação
- Podemos utilizar o processo de Revisão de Crenças

Introdução

Proposta

- Há várias abordagens para o processo de revisão de crenças
- Iremos utilizar o processo de revisão de crenças com verificação de modelos limitada
- Especificações são normalmente escritas em alguma linguagem formal
- Usaremos como linguagem formal a lógica CTL (Computation Tree Logic)
- Procurar inconsistências e revisá-las sugerindo sugestões na especificação

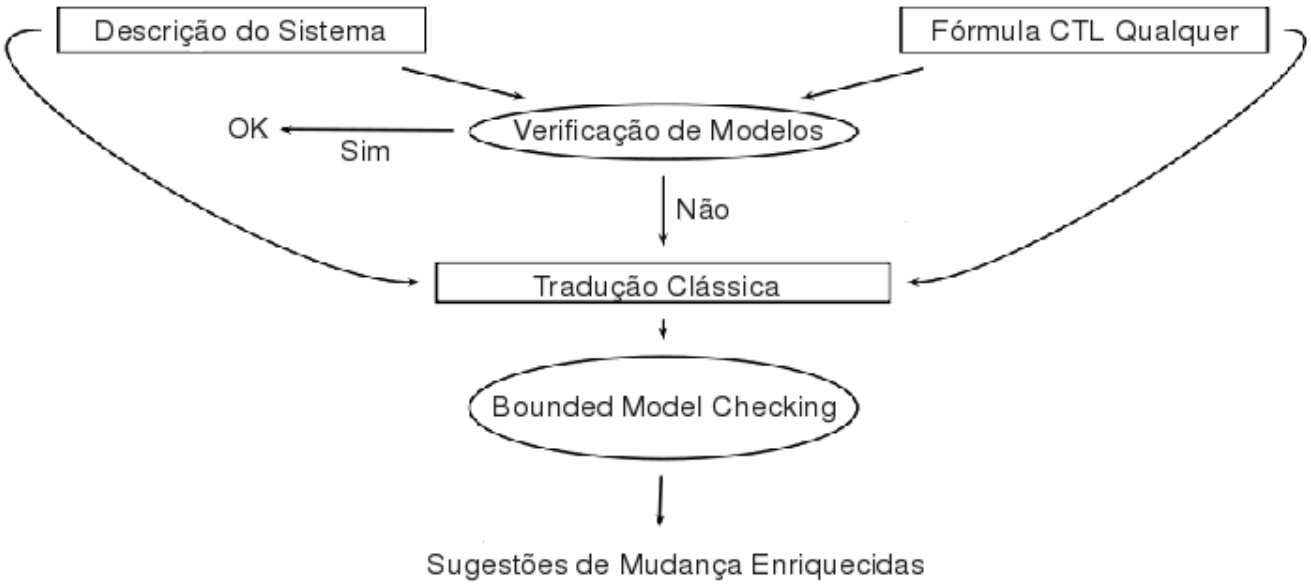


Figura: Revisão de Crenças com Bounded Model Checking

Revisão de Crenças [1]

- Modelar o comportamento de bases de conhecimento que ao receberem novas informações se tornam inconsistentes, a fim de mantê-los consistentes.
- Fundamentada por Alchourrón, Gärdenfors e Makinson, no início da década de 80.
- **Princípio da Mudança Mínima:** reter o máximo de informações possível.

Revisão de Crenças

Operações

- Existem três tipos de operações:
  1. **Expansão ($K + \alpha$):** Uma informação (crença) consistente α é adicionada ao conjunto.

Revisão de Crenças

Operações

- Existem três tipos de operações:
  1. **Expansão ($K + \alpha$):** Uma informação (crença) consistente α é adicionada ao conjunto.
  2. **Contração ($K - \alpha$):** A informação (crença) α é abandonada.

Revisão de Crenças

Operações

- Existem três tipos de operações:
  1. **Expansão ($K + \alpha$):** Uma informação (crença) consistente α é adicionada ao conjunto.
  2. **Contração ($K - \alpha$):** A informação (crença) α é abandonada.
  3. **Revisão ($K * \alpha$):** Uma informação (crença) α é acrescentada ao conjunto, mantendo sua consistência.

Revisão de Crenças

Operações

- Existem três tipos de operações:
  1. **Expansão ($K + \alpha$):** Uma informação (crença) consistente α é adicionada ao conjunto.
  2. **Contração ($K - \alpha$):** A informação (crença) α é abandonada.
  3. **Revisão ($K * \alpha$):** Uma informação (crença) α é acrescentada ao conjunto, mantendo sua consistência.
- E duas relações importante:

Revisão de Crenças

Operações

- Existem três tipos de operações:
  1. **Expansão ($K + \alpha$):** Uma informação (crença) consistente α é adicionada ao conjunto.
  2. **Contração ($K - \alpha$):** A informação (crença) α é abandonada.
  3. **Revisão ($K * \alpha$):** Uma informação (crença) α é acrescentada ao conjunto, mantendo sua consistência.
- E duas relações importante:
  - **Identidade de Harper:** $K - \alpha = K \cap (K * \neg\alpha)$

Revisão de Crenças

Operações

- Existem três tipos de operações:
  1. **Expansão ($K + \alpha$):** Uma informação (crença) consistente α é adicionada ao conjunto.
  2. **Contração ($K - \alpha$):** A informação (crença) α é abandonada.
  3. **Revisão ($K * \alpha$):** Uma informação (crença) α é acrescentada ao conjunto, mantendo sua consistência.
- E duas relações importante:
  - **Identidade de Harper:** $K - \alpha = K \cap (K * \neg\alpha)$
  - **Identidade de Levi:** $K * \alpha = (K - \neg\alpha) + \alpha$

Revisão de Crenças

Operações

- Existem três tipos de operações:
  1. **Expansão ($K + \alpha$):** Uma informação (crença) consistente α é adicionada ao conjunto.
  2. **Contração ($K - \alpha$):** A informação (crença) α é abandonada.
  3. **Revisão ($K * \alpha$):** Uma informação (crença) α é acrescentada ao conjunto, mantendo sua consistência.
- E duas relações importante:
  - **Identidade de Harper:** $K - \alpha = K \cap (K * \neg\alpha)$
  - **Identidade de Levi:** $K * \alpha = (K - \neg\alpha) + \alpha$

Computation Tree Logic [3]

- Duas principais lógicas temporais: LTL e CTL.
- A primeira itera nos caminhos e a segunda utiliza ramificações.
- Em sua maioria uma fórmula em LTL pode ser representado em CTL, portanto iremos utilizar a CTL como nossa linguagem de representação.

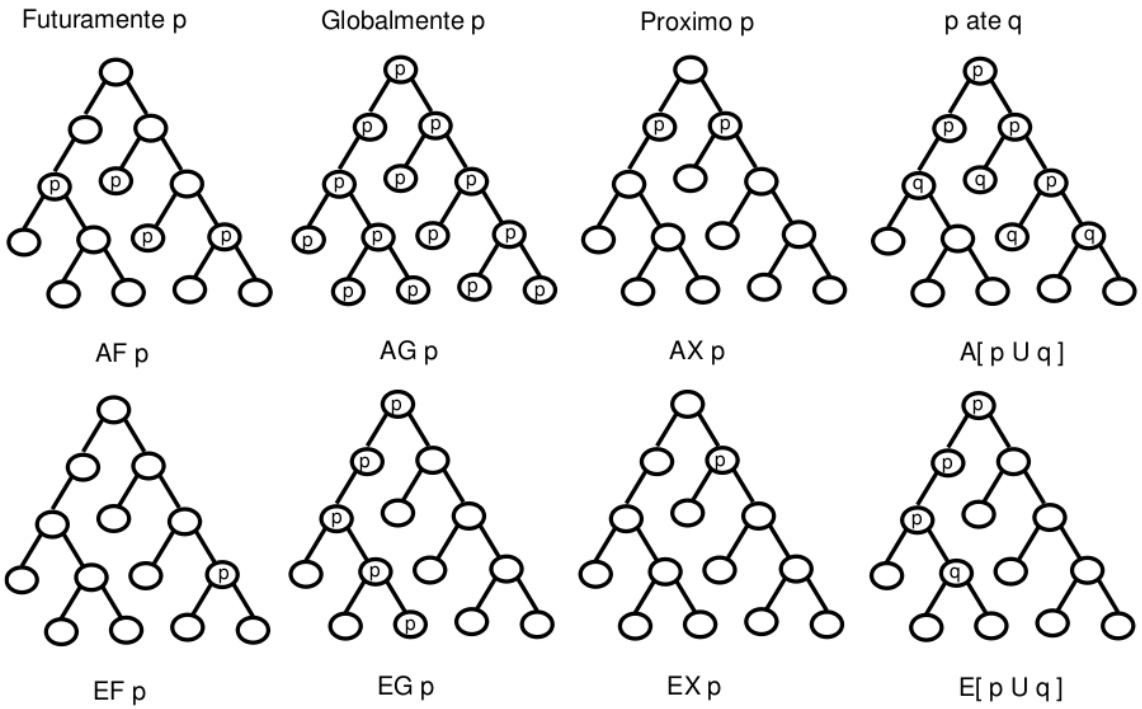


Figura: Operadores Temporais da CTL

Computation Tree Logic

Sintaxe da CTL

Sintaxe

$$\phi ::= p|\neg\phi|\phi \wedge \phi|(AX\phi)|(AG\phi)|(AF\phi)|$$
$$(EX\phi)|(EG\phi)|(EF\phi)|E(\phi \cup \phi)|A(\phi \cup \phi)$$

onde p é um átomo proposicional, $\neg$ e $\wedge$ são os operadores lógicos usuais e os demais são operadores de tempo.

Relações

$$AX\phi = \neg EX\neg\phi$$
$$AG\phi = \neg EF\neg\phi$$
$$AF\phi = \neg EG\neg\phi$$
$$EF\phi = E[true \cup \phi]$$
$$A[\phi \cup \beta] = \neg E[\neg\beta \cup \neg\phi \wedge \neg\beta] \wedge \neg EG\neg\beta$$

Verificação de Modelos

Objetivo: Verificar se um modelo de um sistema satisfaz uma determinada fórmula.

Modelagem: Utilizaremos uma estrutura de Kripke

Problema da Explosão de Estados: Número de estados cresce exponencialmente.

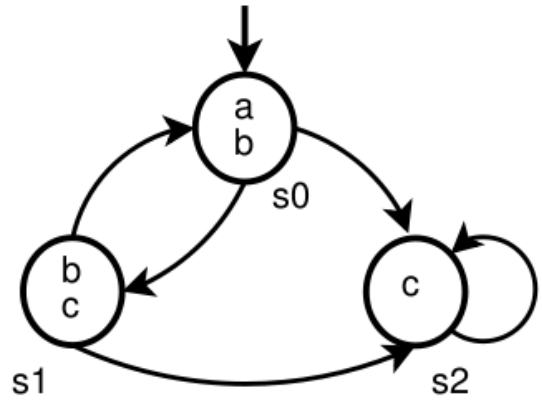


Figura: Estrutura de Kripke

Verificação de Modelos

Objetivo: Verificar se um modelo de um sistema satisfaz uma determinada fórmula.

Modelagem: Utilizaremos uma estrutura de Kripke

Problema da Explosão de Estados: Número de estados cresce exponencialmente.

Solução: Iremos utilizar a abordagem de verificação de modelos limitada.

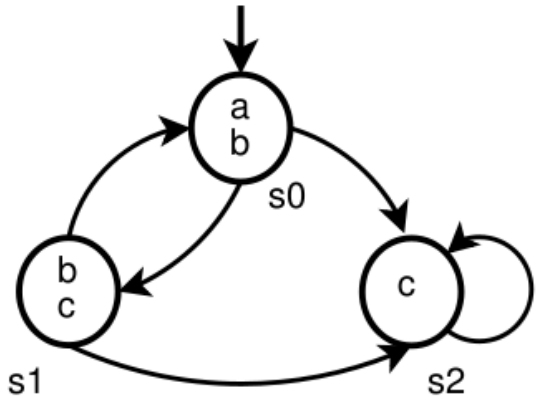


Figura: Estrutura de Kripke

Verificação de Modelos

Objetivo: Verificar se um modelo de um sistema satisfaz uma determinada fórmula.

Modelagem: Utilizaremos uma estrutura de Kripke

Problema da Explosão de Estados: Número de estados cresce exponencialmente.

Solução: Iremos utilizar a abordagem de verificação de modelos limitada.

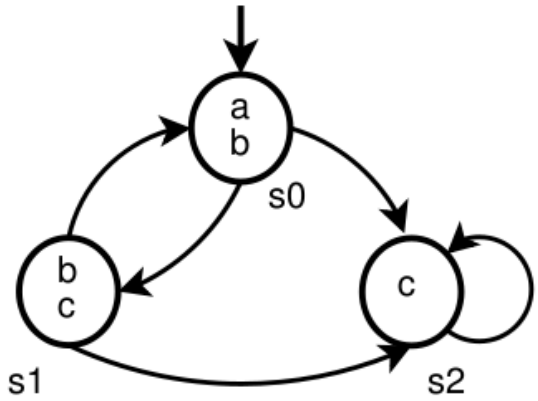


Figura: Estrutura de Kripke

Verificação de Modelos

Bounded Model Checking [2]

- A proposta é de colocar um limite k no tamanho máximo dos caminhos.
- Traduzir a estrutura de Kripke K e a fórmula em CTL ϕ em fórmulas proposicionais K^k e A^k_ϕ.

Verificação de Modelos

Bounded Model Checking [2]

- A proposta é de colocar um limite k no tamanho máximo dos caminhos.
- Traduzir a estrutura de Kripke K e a fórmula em CTL ϕ em fórmulas proposicionais K^k e A^k_ϕ.
- Utilizar um resolvedor SAT para verificar a satisfabilidade de $K^k \wedge A^k_\phi$.

Verificação de Modelos

Bounded Model Checking [2]

- A proposta é de colocar um limite k no tamanho máximo dos caminhos.
- Traduzir a estrutura de Kripke K e a fórmula em CTL ϕ em fórmulas proposicionais K^k e A^k_ϕ.
- Utilizar um resolvedor SAT para verificar a satisfabilidade de $K^k \wedge A^k_\phi$.
- Se for insatisfazível significa que $\neg\phi$ vale em K até o limite estabelecido.

Verificação de Modelos

Bounded Model Checking [2]

- A proposta é de colocar um limite k no tamanho máximo dos caminhos.
- Traduzir a estrutura de Kripke K e a fórmula em CTL ϕ em fórmulas proposicionais K^k e A^k_ϕ.
- Utilizar um resolvedor SAT para verificar a satisfabilidade de $K^k \wedge A^k_\phi$.
- Se for insatisfazível significa que $\neg\phi$ vale em K até o limite estabelecido.
- Caso contrário uma valoração v é retornada.

Verificação de Modelos

Bounded Model Checking [2]

- A proposta é de colocar um limite k no tamanho máximo dos caminhos.
- Traduzir a estrutura de Kripke K e a fórmula em CTL ϕ em fórmulas proposicionais K^k e A^k_ϕ.
- Utilizar um resolvedor SAT para verificar a satisfabilidade de $K^k \wedge A^k_\phi$.
- Se for insatisfazível significa que $\neg\phi$ vale em K até o limite estabelecido.
- Caso contrário uma valoração v é retornada.

Descrição do Algoritmo

- Nosso trabalho segue a linha proposta por Finger e Wassermann [4].
- Podemos descrever o algoritmo da seguinte forma:
  1. Entrada

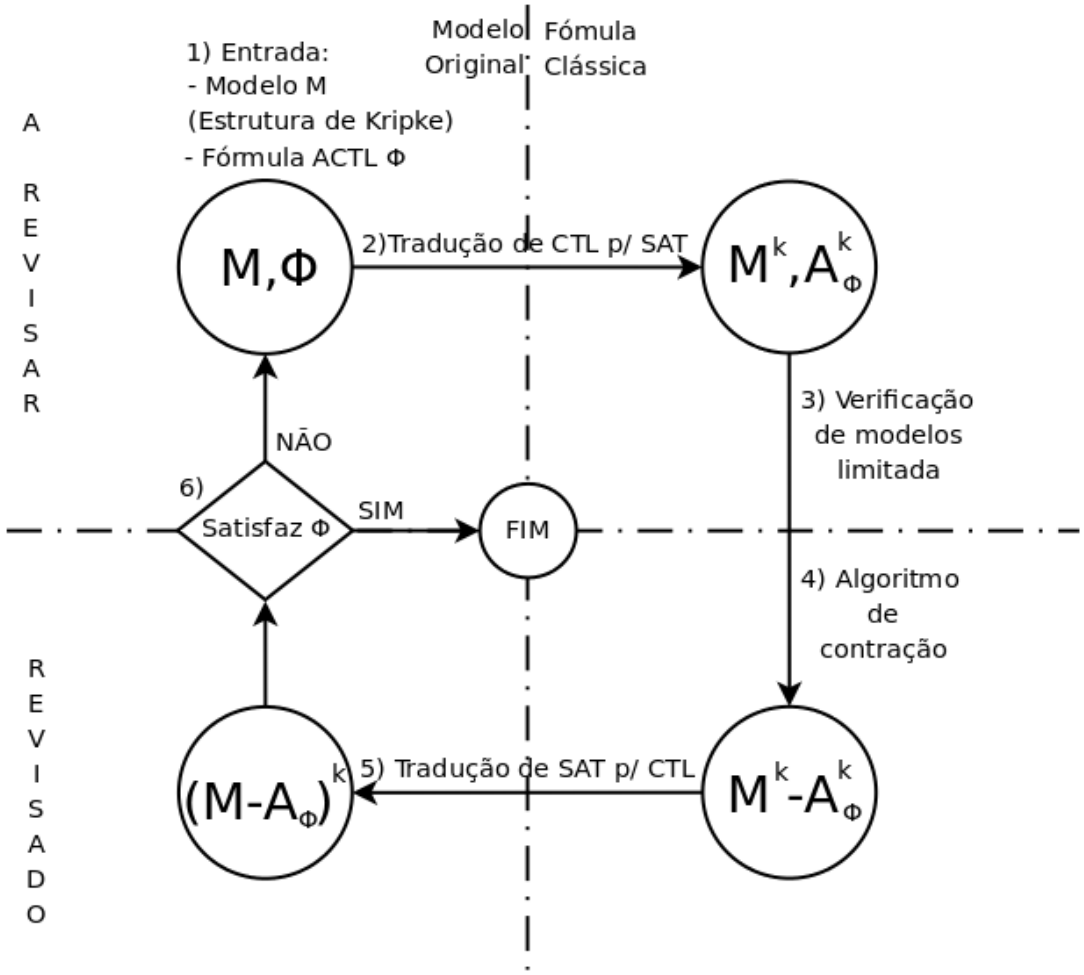


Figura: Algoritmo proposto

Descrição do Algoritmo

- Nosso trabalho segue a linha proposta por Finger e Wassermann [4].
- Podemos descrever o algoritmo da seguinte forma:
  1. Entrada
  2. Tradução inicial

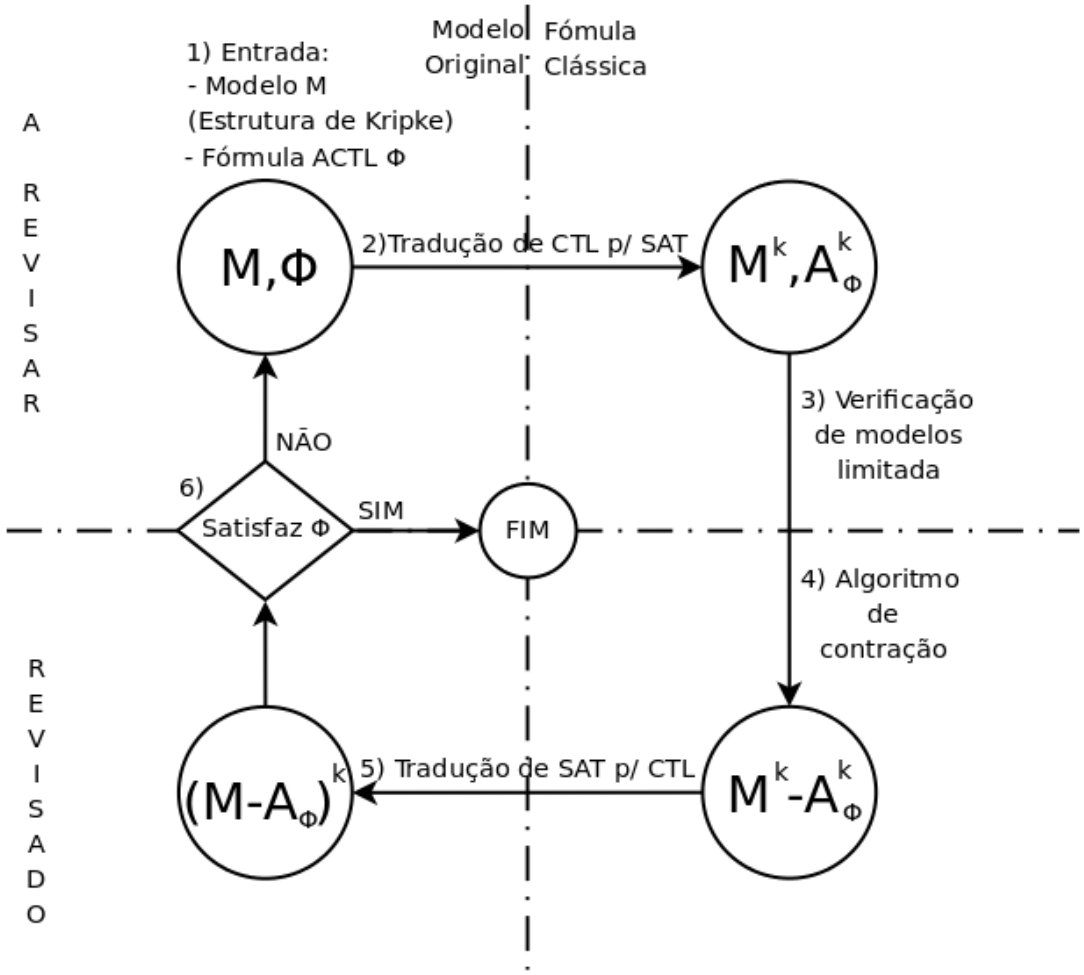


Figura: Algoritmo proposto

Descrição do Algoritmo

- Nosso trabalho segue a linha proposta por Finger e Wassermann [4].
- Podemos descrever o algoritmo da seguinte forma:
  1. Entrada
  2. Tradução inicial
  3. Verificação inicial

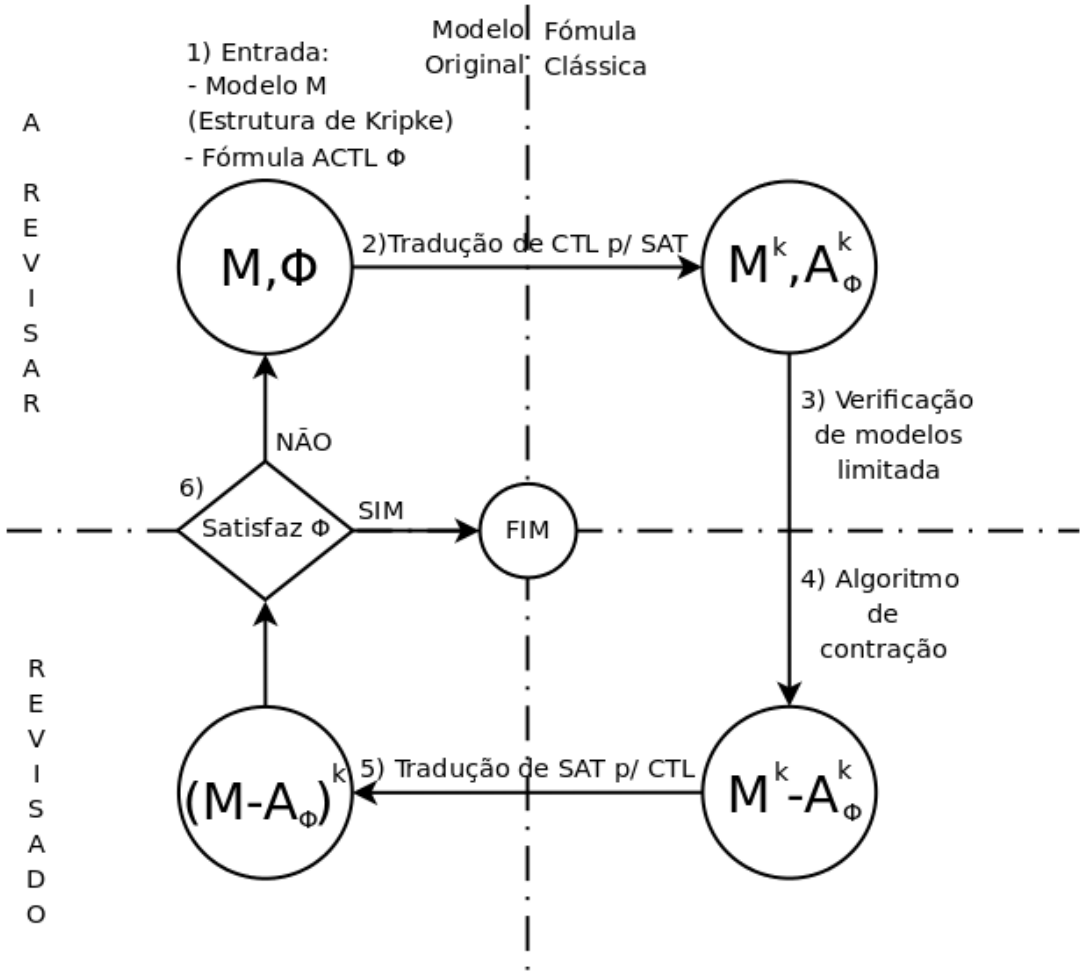


Figura: Algoritmo proposto

Descrição do Algoritmo

- Nosso trabalho segue a linha proposta por Finger e Wassermann [4].
- Podemos descrever o algoritmo da seguinte forma:
  1. Entrada
  2. Tradução inicial
  3. Verificação inicial
  4. Contração

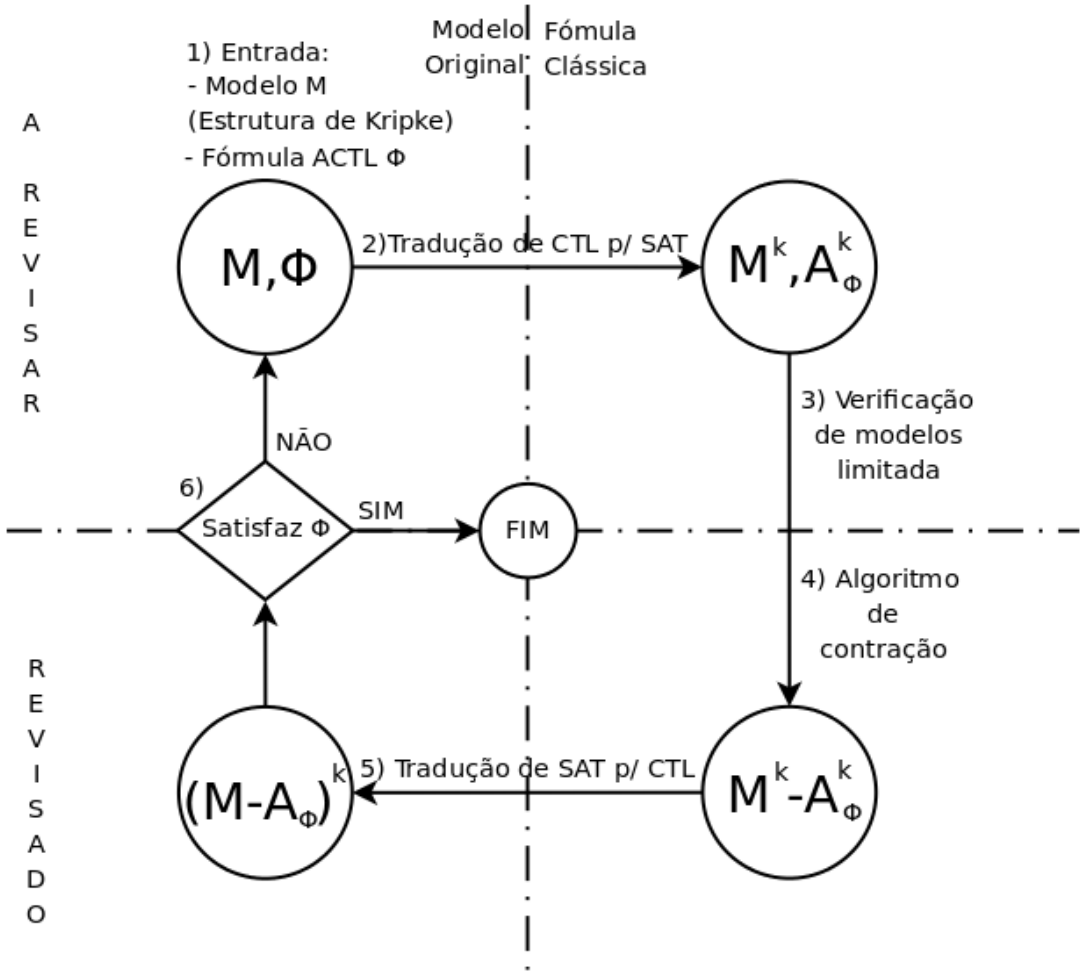


Figura: Algoritmo proposto

Descrição do Algoritmo

- Nosso trabalho segue a linha proposta por Finger e Wassermann [4].
- Podemos descrever o algoritmo da seguinte forma:
  1. Entrada
  2. Tradução inicial
  3. Verificação inicial
  4. Contração
  5. Tradução reversa

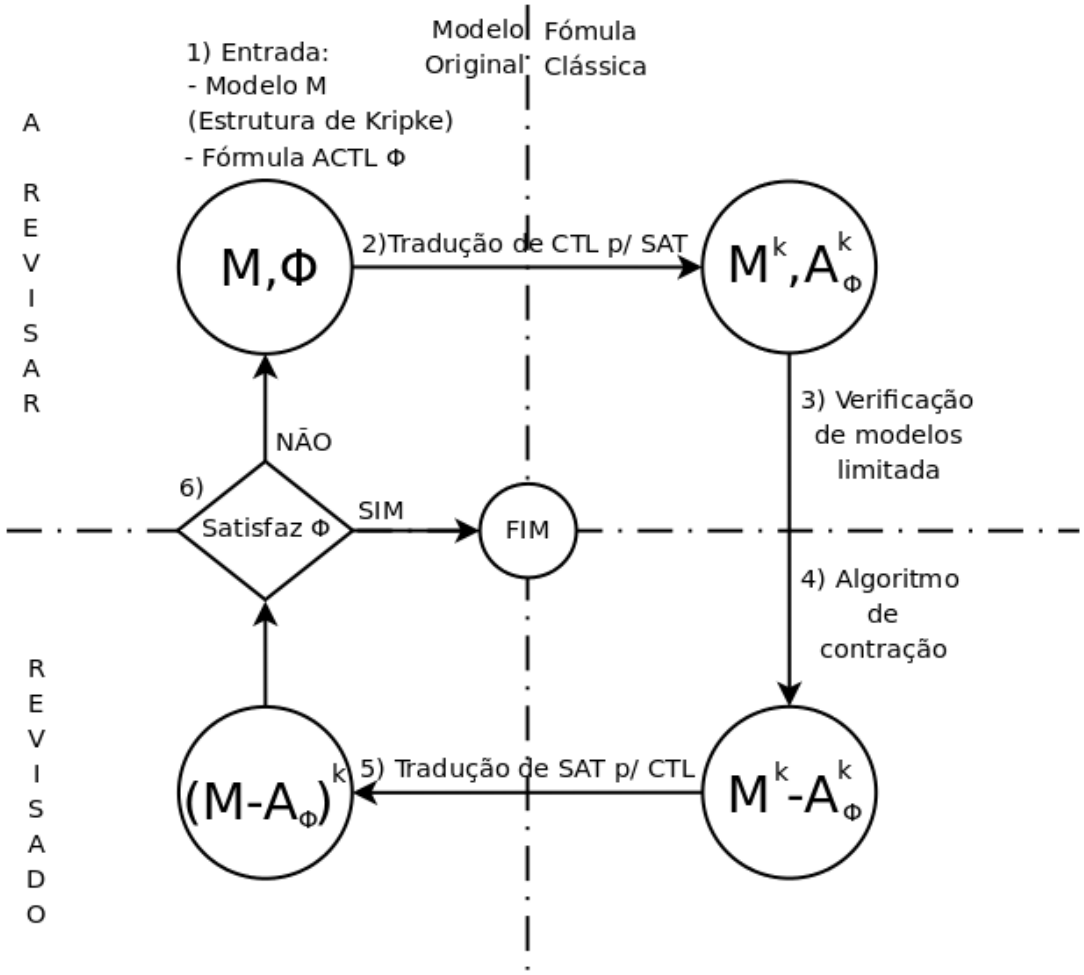


Figura: Algoritmo proposto

Descrição do Algoritmo

- Nosso trabalho segue a linha proposta por Finger e Wassermann [4].
- Podemos descrever o algoritmo da seguinte forma:
  1. Entrada
  2. Tradução inicial
  3. Verificação inicial
  4. Contração
  5. Tradução reversa
  6. Verificação final

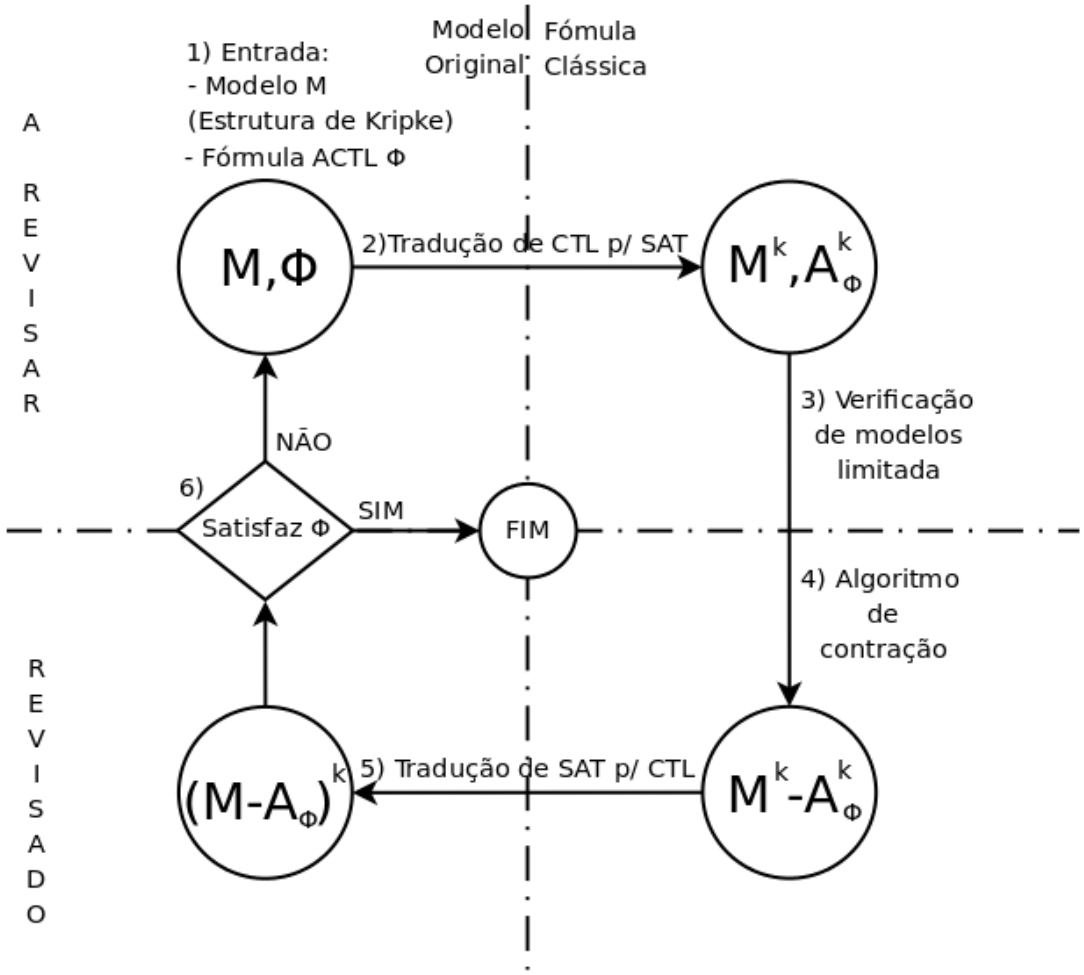


Figura: Algoritmo proposto

Descrição do Algoritmo

- Nosso trabalho segue a linha proposta por Finger e Wassermann [4].
- Podemos descrever o algoritmo da seguinte forma:
  1. Entrada
  2. Tradução inicial
  3. Verificação inicial
  4. Contração
  5. Tradução reversa
  6. Verificação final

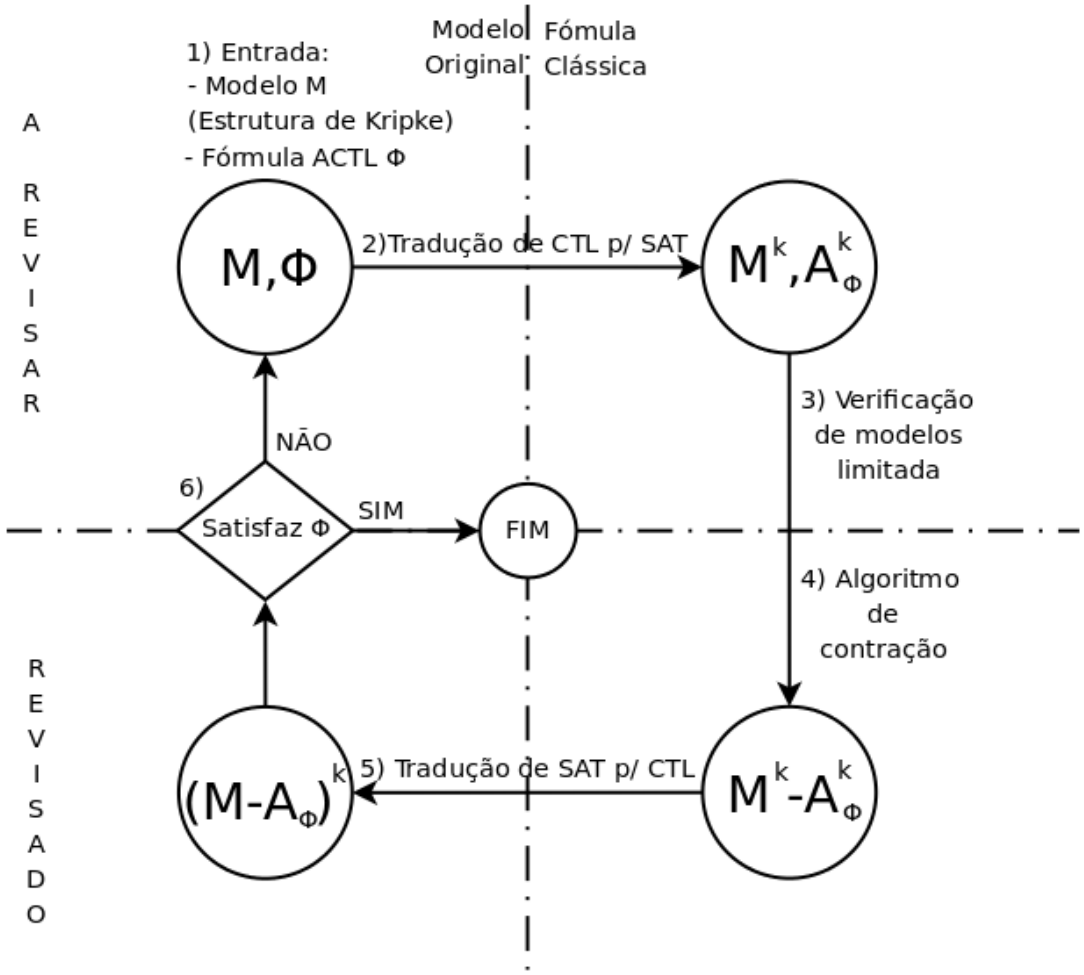


Figura: Algoritmo proposto

Tradução para lógica proposicional

- Implementação do algoritmo proposto por Penczek et. al [5] para estruturas de Kripke.
- **Entrada:** uma estrutura de Kripke e uma fórmula em ACTL (fragmento universal da CTL).
- $ACTL = \{\neg\varphi \mid \varphi \in ECTL\}$.
- **ECTL:** somente operador existencial e negação em proposições.
- Fórmulas na *forma normal clausal (CNF)* .
- Para cada estado, o representamos como um vetor de variáveis proposicionais.

Tradução para lógica proposicional

Funções auxiliares

Função $f_k(\phi)$

Definição: determina o número de caminhos suficientes para checar uma fórmula em ECTL, definida em [5].

Função $D(a, \alpha)$

Definição: define uma variável proposicional a para uma fórmula em CNF α qualquer.

Tradução do modelo

Definimos então as seguintes fórmulas propocisionais:

- $I_s(w)$ sse w é o estado inicial;

Tradução do modelo

Definimos então as seguintes fórmulas propocisionais:

- $I_s(w)$ sse w é o estado inicial;
- $T(w, v)$ *sse* $(w, v) \in T$;

Tradução do modelo

Definimos então as seguintes fórmulas propocisionais:

- $I_s(w)$ sse w é o estado inicial;
- $T(w, v)$ *sse* $(w, v) \in T$;
- $p(w)$ *sse* $p \in \nu(w)$;

Tradução do modelo

Definimos então as seguintes fórmulas propocisionais:

- $I_s(w)$ sse w é o estado inicial;
- $T(w, v)$ *sse* $(w, v) \in T$;
- $p(w)$ *sse* $p \in \nu(w)$;
- $H(w, v)$ *sse* $w = v$;

Tradução do modelo

Definimos então as seguintes fórmulas propocisionais:

- $I_s(w)$ sse w é o estado inicial;
- $T(w, v)$ *sse* $(w, v) \in T$;
- $p(w)$ *sse* $p \in \nu(w)$;
- $H(w, v)$ *sse* $w = v$;
- $L_{k,j}(l)$ que codifica um loop no $k - caminho\ j$.

Tradução do modelo

Definimos então as seguintes fórmulas propocisionais:

- $l_s(w)$ sse w é o estado inicial;
- $T(w, v)$ *sse* $(w, v) \in T$;
- $p(w)$ *sse* $p \in \nu(w)$;
- $H(w, v)$ *sse* $w = v$;
- $L_{k,j}(l)$ que codifica um loop no $k - caminho\ j$.

Fórmula proposicional do modelo

$$[M^{\varphi,s}]_k := l_s(w_{0,0}) \wedge \bigwedge_{j \in LL\varphi} \bigwedge_{i=0}^{k} T(w_{i,j}, w_{i+1,j})$$

onde $w_{0,0}$ e $w_{i,j}$ para $i = 0, ..., k$ e $j \in LL\varphi$ representam os estados através dos $k - caminhos$, e $|LL\varphi| = f_k(\varphi)$.

Tradução do modelo

Definimos então as seguintes fórmulas propocisionais:

- $l_s(w)$ sse w é o estado inicial;
- $T(w, v)$ *sse* $(w, v) \in T$;
- $p(w)$ *sse* $p \in \nu(w)$;
- $H(w, v)$ *sse* $w = v$;
- $L_{k,j}(l)$ que codifica um loop no $k - caminho\ j$.

Fórmula proposicional do modelo

$$[M^{\varphi,s}]_k := l_s(w_{0,0}) \wedge \bigwedge_{j \in LL\varphi} \bigwedge_{i=0}^{k} T(w_{i,j}, w_{i+1,j})$$

onde $w_{0,0}$ e $w_{i,j}$ para $i = 0, ..., k$ e $j \in LL\varphi$ representam os estados através dos $k - caminhos$, e $|LL\varphi| = f_k(\varphi)$.

Tradução do modelo

Fórmulas básicas

- $lit(0, p) = \neg p$ e $lit(1, p) = p$;
- $I_s(w) := \bigwedge_{l=1}^{n} lit(s[l], w[j])$;
- $T(w, v) :=$
$$\begin{cases} \bigvee_{m=1}^{|T|} t_m \\ \bigwedge_{\substack{m=1 \\ (x,y)\in T}}^{|T|} \begin{bmatrix} \bigwedge_{l=1}^{n}(\neg t_m \vee lit(x[l], w[l])) \\ \bigwedge_{l=1}^{n}(\neg t_m \vee lit(y[l], v[l])) \\ \wedge(t_m \vee \bigvee_{l=1}^{n} lit(x[l], w[l]) \vee \bigvee_{l=1}^{n} lit(x[l], w[l])) \end{bmatrix} \end{cases};$$
- $p(w) := V(w, p) := \begin{cases} (\bigvee_{x\,:\,p\,\in\,\nu(x)} s_i)\wedge \\ \bigvee_{x\,:\,p\,\in\,\nu(x)} D(s_i, (\bigwedge_{l=1}^{n} lit(x[l], w[l]) \wedge p)) \end{cases};$
- $\neg p(w) := V(w, \neg p) :=$
$$\begin{cases} (\bigvee_{x\,:\,p\,\notin\,\nu(x)} s_i)\wedge \\ \bigvee_{x\,:\,p\,\notin\,\nu(x)} D(s_i, (\bigwedge_{l=1}^{n} lit(x[l], w[l]) \wedge \neg p)) \end{cases};$$
- $H(w, v) := \bigwedge_{l=1}^{n}(\neg w[l] \vee v[l]) \wedge (w[l] \vee \neg v[l])$;
- $L_{k,j}(l) := T(w_{k,j}, w_{l,j})$ que codifica um loop no $k - caminho\ j$.

Tradução da fórmula ECTL

- $[\varphi]_k^{[m,n]}$ denota a tradução de uma fórmula ECTL φ em $w_{m,n}$ para uma fórmula proposicional.
- Definida para cada fórmula da sintaxe da ECTL.
- Denotaremos $[\varphi]_k^{[0,0]}$ como $[\varphi]_{M_k}$.

Tradução da fórmula ECTL

- $[p]_k^{[m,n]} := p(w_{m,n})$;
- $[\neg p]_k^{[m,n]} := \neg p(w_{m,n})$;
- $[\alpha \wedge \beta]_k^{[m,n]} := [\alpha]_k^{[m,n]} \wedge [\beta]_k^{[m,n]}$;
- $[\alpha \vee \beta]_k^{[m,n]} := \alpha_k^{m,n} \vee \beta_k^{m,n} \wedge D(\alpha_k^{m,n}, [\alpha]_k^{[m,n]}) \wedge D(\beta_k^{m,n}, [\beta]_k^{[m,n]})$;
- $[EX\ \alpha]_k^{[m,n]} := \begin{cases} (\bigvee_{i \in LL\varphi} ex_i^{m,n}) \wedge \\ \bigwedge_{i \in LL\varphi} D(ex_i^{m,n}, H(w_{m,n}, w_{0,i}) \wedge [\alpha]_k^{[1,j]}) \end{cases}$;
- $[EG\ \alpha]_k^{[m,n]} :=$
  $\begin{cases} (\bigvee_{i \in LL\varphi} eg_i^{m,n}) \wedge \\ \bigwedge_{i \in LL\varphi} D(eg_i^{m,n}, H(w_{m,n}, w_{0,i}) \wedge (\bigvee_{l=0}^{k} L_{k,i}(l)) \wedge \bigwedge_{j=0}^{k} [\alpha]_k^{[j,ij]}) \end{cases}$;
- $[E(\alpha U \beta)]_k^{[m,n]} :=$
  $(\bigvee_{i \in LL\varphi} eu_i^{m,n}) \wedge \bigwedge_{i \in LL\varphi} \begin{bmatrix} \bigwedge_{j=0}^{k} D(\alpha\beta_k^{j,i}, [\beta]_k^{[j,i]} \wedge \bigwedge_{t=0}^{j-1} [\alpha]_k^{[t,i]}) \wedge \\ D(eu_i^{m,n}, H(w_{m,n}, w_{0,i}) \wedge \bigvee_{j=0}^{k} \alpha\beta_k^{j,i}) \end{bmatrix}$;

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.
- Caso contrário, o resolvedor SAT irá gerar uma valoração v, a qual representa um caminho através do modelo no qual φ vale.

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.
- Caso contrário, o resolvedor SAT irá gerar uma valoração v, a qual representa um caminho através do modelo no qual φ vale.

Figura: BMC requisitando Revisão de Crenças

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.
- Caso contrário, o resolvedor SAT irá gerar uma valoração v, a qual representa um caminho através do modelo no qual φ vale.

Figura: BMC requisitando Revisão de Crenças

- Sugestões de mudanças a partir da valoração, gerando uma fórmula $A_{\neg v}$.

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.
- Caso contrário, o resolvedor SAT irá gerar uma valoração v, a qual representa um caminho através do modelo no qual φ vale.

Figura: BMC requisitando Revisão de Crenças

- Sugestões de mudanças a partir da valoração, gerando uma fórmula $A_{\neg v}$.
- Verificamos então $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k} \wedge A_{\neg v}$ novamente no resolvedor SAT.

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.
- Caso contrário, o resolvedor SAT irá gerar uma valoração v, a qual representa um caminho através do modelo no qual φ vale.

Figura: BMC requisitando Revisão de Crenças

- Sugestões de mudanças a partir da valoração, gerando uma fórmula $A_{\neg v}$.
- Verificamos então $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k} \wedge A_{\neg v}$ novamente no resolvedor SAT.
- Pode ser o caso em que essa fórmula seja satisfeita por uma valoração v_2.

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.
- Caso contrário, o resolvedor SAT irá gerar uma valoração v, a qual representa um caminho através do modelo no qual φ vale.

Figura: BMC requisitando Revisão de Crenças

- Sugestões de mudanças a partir da valoração, gerando uma fórmula $A_{\neg v}$.
- Verificamos então $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k} \wedge A_{\neg v}$ novamente no resolvedor SAT.
- Pode ser o caso em que essa fórmula seja satisfeita por uma valoração v_2.
- Iteramos m vezes, até que obtenhamos uma fórmula inconsistente.

Verificação e Contração

- Verificar $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k}$ num resolvedor SAT.
- Se a fórmula for inconsistentes, nada precisa ser feito em relação à revisão.
- Caso contrário, o resolvedor SAT irá gerar uma valoração v, a qual representa um caminho através do modelo no qual φ vale.

Figura: BMC requisitando Revisão de Crenças

- Sugestões de mudanças a partir da valoração, gerando uma fórmula $A_{\neg v}$.
- Verificamos então $[M^{\varphi,s}]_k \wedge [\varphi]_{M_k} \wedge A_{\neg v}$ novamente no resolvedor SAT.
- Pode ser o caso em que essa fórmula seja satisfeita por uma valoração v_2.
- Iteramos m vezes, até que obtenhamos uma fórmula inconsistente.

Sugestões de mudança

Variáveis a serem observadas

- Uma **variável de estado** $w_{i,j}$;
- Uma **variável de transição** $t_m^{i,i+1,j}$;
- Uma **propriedade** $p_{i,j}$.

Possíveis mudanças

- Remover um estado;
- Remover uma transição;
- Modificar uma propriedade.

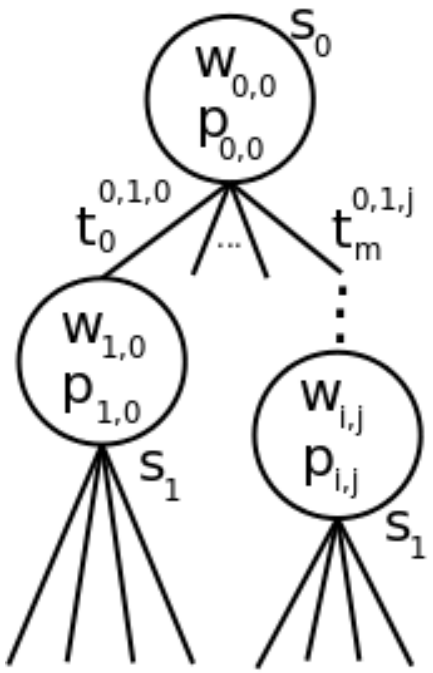


Figura: Variáveis utilizadas para gerar as sugestões

Sugestões de mudança

Variáveis a serem observadas

- Uma **variável de estado** $w_{i,j}$;
- Uma **variável de transição** $t_m^{i,i+1,j}$;
- Uma **propriedade** $p_{i,j}$.

Possíveis mudanças

- Remover um estado;
- Remover uma transição;
- Modificar uma propriedade.

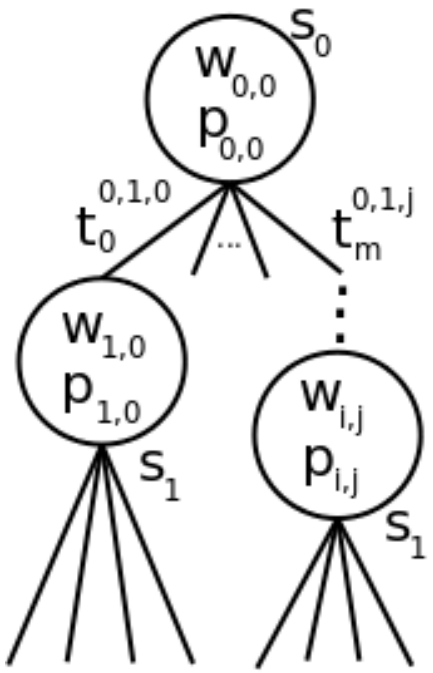


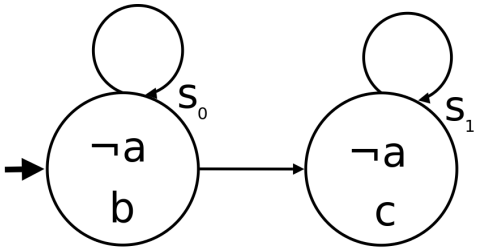
Figura: Variáveis utilizadas para gerar as sugestões

Modificações na fórmula

- **Remover um estado:**
  $R_s(v) = \bigwedge_{j \in LL\varphi} \bigwedge_{i=0}^{k} \bigvee_{l=0}^{n} \neg lit(v[l], w_{i,j}[l])$,
  onde v é a valoração do vetor do estado a ser excluído;
- **Remover uma transição:**
  $$R_t(v) = \bigwedge_{j \in LL\varphi} \bigwedge_{i=0}^{k-1} \begin{bmatrix} \bigvee_{l=0}^{n} \neg lit(x[l], w_{i,j}[l]) \vee \\ \bigvee_{l=0}^{n} \neg lit(y[l], w_{i+1,j}[l]) \vee \\ \neg t_m^{i,i+1,j} \end{bmatrix},$$
  onde (x, y) formam a transição $t_m^{i,i+1,j}$;
- **Modificar uma propriedade:**
  $R_p(v) = \bigwedge_{j \in LL\varphi} \bigwedge_{i=0}^{k} [\bigvee_{l=0}^{n} \neg lit(v[l], w_{i,j}[l])) \vee \neg p_{i,j}]$,
  onde v é a valoração do estado onde p deve ser modificada.

Um simples exemplo

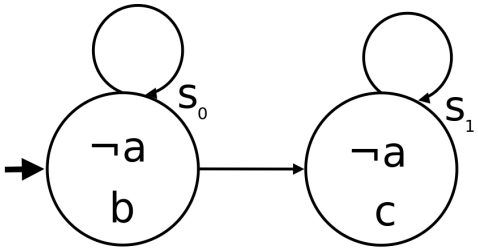
- Testaremos $M \models AG\ a$.



$I = \{s_0\}$

Um simples exemplo

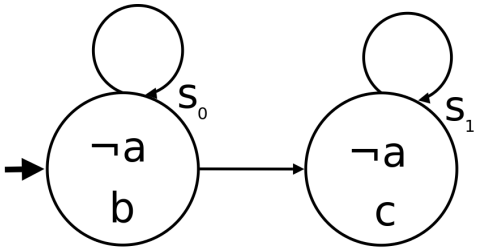
- Testaremos $M \models AG\ a$.



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0, 1\}$$

Um simples exemplo

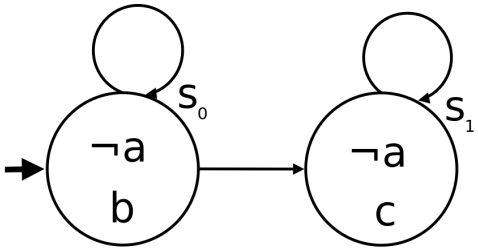
- Testaremos $M \models AG\ a$.



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \left\{ \begin{matrix} (s0, s0), \\ (s0, s1), \\ (s1, s1) \end{matrix} \right\}$$

Um simples exemplo

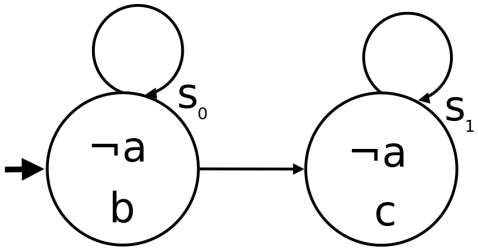
- Testaremos $M \models AG\ a$.



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \left\{\begin{matrix}(s0, s0),\\ (s0, s1),\\ (s1, s1)\end{matrix}\right\}$$
$$\nu(s_0) = \{b\}$$

Um simples exemplo

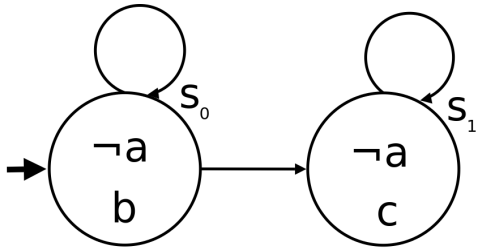
- Testaremos $M \models AG\ a$.



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \left\{\begin{matrix}(s0, s0),\\ (s0, s1),\\ (s1, s1)\end{matrix}\right\}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$

Um simples exemplo

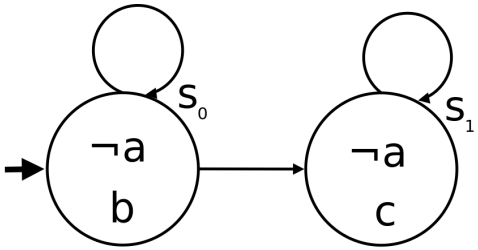
- Testaremos $M \models AG\ a$.



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \left\{\begin{matrix}(s0, s0),\\ (s0, s1),\\ (s1, s1)\end{matrix}\right\}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$
$$\varphi = \neg\psi =$$
$$E(T\ U \neg\ a)$$

Um simples exemplo

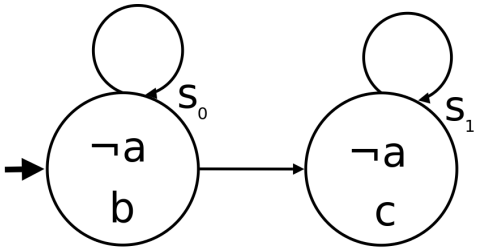
- Testaremos $M \models AG\ a$.
- Adotaremos um $k = 1$.



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \begin{Bmatrix} (s0, s0), \\ (s0, s1), \\ (s1, s1) \end{Bmatrix}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$
$$\varphi = \neg\psi =$$
$$E(T\ U \neg\ a)$$

Um simples exemplo

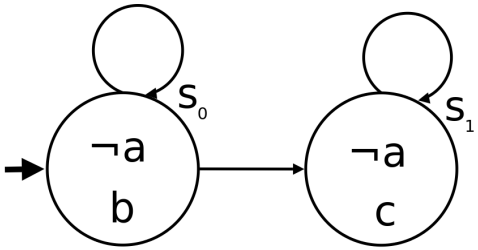
- Testaremos $M \models AG\ a$.
- Adotaremos um $k = 1$.
- $f_1(E(T\ U \neg a)) = 1 * f_1(T) + f_1(\neg a) + 1 = 1$



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \begin{Bmatrix} (s0, s0), \\ (s0, s1), \\ (s1, s1) \end{Bmatrix}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$
$$\varphi = \neg\psi =$$
$$E(T\ U \neg a)$$

Um simples exemplo

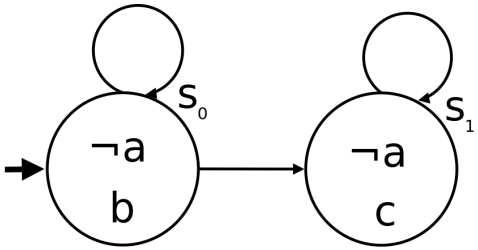
- Testaremos $M \models AG\ a$.
- Adotaremos um $k = 1$.
- $f_1(E(T\ U \neg a)) = 1 * f_1(T) + f_1(\neg a) + 1 = 1$
- Descrevemos as fórmulas para a tradução do modelo geradas abaixo:



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \begin{Bmatrix} (s0, s0), \\ (s0, s1), \\ (s1, s1) \end{Bmatrix}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$
$$\varphi = \neg\psi =$$
$$E(T\ U \neg a)$$

Um simples exemplo

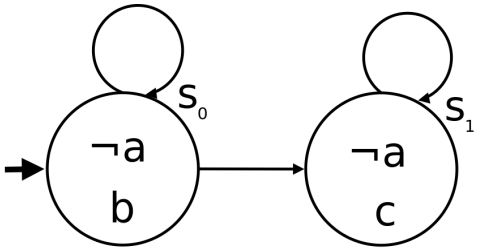
- Testaremos $M \models AG\ a$.
- Adotaremos um $k = 1$.
- $f_1(E(T\ U \neg a)) = 1 * f_1(T) + f_1(\neg a) + 1 = 1$
- Descrevemos as fórmulas para a tradução do modelo geradas abaixo:
  - $l_s(w_{0,0}) = \neg w_{0,0}$



$$I = \{s_0\}$$
$$W = \{s_0, s_1\}$$
$$= \{0,\ 1\}$$
$$T = \begin{Bmatrix} (s0, s0), \\ (s0, s1), \\ (s1, s1) \end{Bmatrix}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$
$$\varphi = \neg\psi =$$
$$E(T\ U \neg a)$$

Um simples exemplo

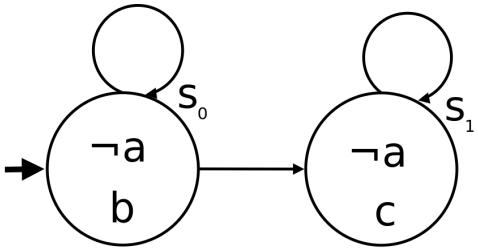
- Testaremos $M \models AG\ a$.
- Adotaremos um $k = 1$.
- $f_1(E(T\ U \neg a)) = 1 * f_1(T) + f_1(\neg a) + 1 = 1$
- Descrevemos as fórmulas para a tradução do modelo geradas abaixo:
  - $I_s(w_{0,0}) = \neg w_{0,0}$
  - $T(w_{0,0}, w_{1,0}) = \begin{cases} t_0 \vee t_1 \vee t_2 \wedge \\ \neg t_0 \vee \neg w_{0,0} \wedge \\ \neg t_0 \vee \neg w_{1,0} \wedge \\ t_0 \vee w_{0,0} \vee w_{1,0} \wedge \\ \neg t_1 \vee \neg w_{0,0} \wedge \\ \neg t_1 \vee w_{1,0} \wedge \\ t_1 \vee w_{0,0} \vee \neg w_{1,0} \wedge \\ \neg t_2 \vee w_{0,0} \wedge \\ \neg t_2 \vee w_{1,0} \wedge \\ t_2 \vee \neg w_{0,0} \vee \neg w_{1,0} \end{cases}$



$$I = \{s_0\}$$
$$W = \{s_0, s_1\} = \{0, 1\}$$
$$T = \begin{cases} (s0, s0), \\ (s0, s1), \\ (s1, s1) \end{cases}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$
$$\varphi = \neg\psi = E(T\ U \neg a)$$

Um simples exemplo

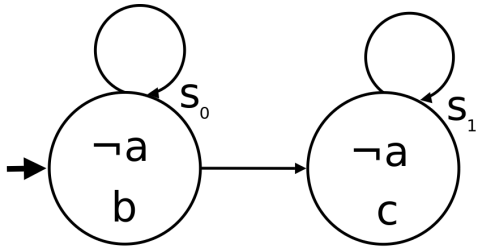
- Testaremos $M \models AG\ a$.
- Adotaremos um $k = 1$.
- $f_1(E(T\ U \neg a)) = 1 * f_1(T) + f_1(\neg a) + 1 = 1$
- Descrevemos as fórmulas para a tradução do modelo geradas abaixo:
  - $I_s(w_{0,0}) = \neg w_{0,0}$
  - $$T(w_{0,0}, w_{1,0}) = \begin{cases} t_0 \vee t_1 \vee t_2 \wedge \\ \neg t_0 \vee \neg w_{0,0} \wedge \\ \neg t_0 \vee \neg w_{1,0} \wedge \\ t_0 \vee w_{0,0} \vee w_{1,0} \wedge \\ \neg t_1 \vee \neg w_{0,0} \wedge \\ \neg t_1 \vee w_{1,0} \wedge \\ t_1 \vee w_{0,0} \vee \neg w_{1,0} \wedge \\ \neg t_2 \vee w_{0,0} \wedge \\ \neg t_2 \vee w_{1,0} \wedge \\ t_2 \vee \neg w_{0,0} \vee \neg w_{1,0} \end{cases}$$



$$I = \{s_0\}$$
$$W = \{s_0, s_1\} = \{0,\ 1\}$$
$$T = \begin{cases} (s0, s0), \\ (s0, s1), \\ (s1, s1) \end{cases}$$
$$\nu(s_0) = \{b\}$$
$$\nu(s_1) = \{c\}$$
$$\varphi = \neg\psi = E(T\ U \neg a)$$

Descrevemos a tradução para a fórmula abaixo:

- $H(w_{0,0},\ w_{0,0}) = \begin{cases} \neg w_{0,0} \ \vee \ w_{0,0} \ \wedge \\ w_{0,0} \ \vee \ \neg w_{0,0} \end{cases}$

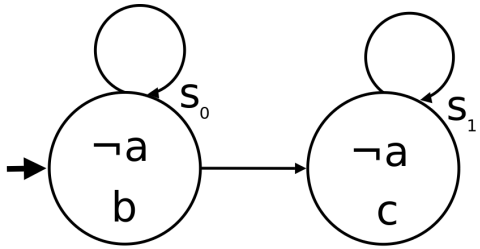


$$[E(T\ U \neg a)]_1^{[0,0]} = H(w_{0,0},\ w_{0,0}) \wedge (\alpha_{0,0} \ \vee \ \alpha_{1,0}) \wedge D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \wedge D(\alpha_{1,0}, [\neg a]_1^{1,0})$$

$$\alpha_{0,0} = [\neg a]_1^{[0,0]}$$
$$\alpha_{1,0} = [\neg a]_1^{[1,0]}$$

Descrevemos a tradução para a fórmula abaixo:

- $H(w_{0,0},\ w_{0,0}) = \begin{cases} \neg w_{0,0} \ \vee \ w_{0,0} \ \wedge \\ w_{0,0} \ \vee \ \neg w_{0,0} \end{cases}$
- $D(\alpha_{0,0},\ [\neg a]_1^{0,0}) = \begin{cases} \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg a_{0,0} \\ \alpha_{0,0} \vee w_{0,0} \vee a_{0,0} \\ \alpha_{0,0} \vee \neg w_{0,0} \vee a_{0,0} \end{cases}$

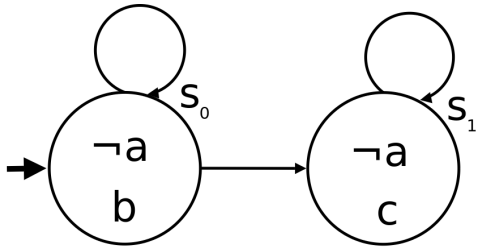


$$[E(T\ U \neg a)]_1^{[0,0]} = H(w_{0,0},\ w_{0,0}) \wedge (\alpha_{0,0} \ \vee \ \alpha_{1,0}) \wedge D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \wedge D(\alpha_{1,0}, [\neg a]_1^{1,0})$$

$$\alpha_{0,0} = [\neg a]_1^{[0,0]}$$
$$\alpha_{1,0} = [\neg a]_1^{[1,0]}$$

Descrevemos a tradução para a fórmula abaixo:

- $H(w_{0,0},\ w_{0,0}) = \begin{cases} \neg w_{0,0} \ \vee \ w_{0,0} \ \wedge \\ w_{0,0} \ \vee \ \neg w_{0,0} \end{cases}$
- $D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \ = \ \begin{cases} \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg a_{0,0} \\ \alpha_{0,0} \vee w_{0,0} \vee a_{0,0} \\ \alpha_{0,0} \vee \neg w_{0,0} \vee a_{0,0} \end{cases}$
- $D(\alpha_{1,0},\ [\neg a]_1^{1,0}) \ = \ \begin{cases} \neg\alpha_{1,0} \vee \neg w_{1,0} \vee \neg a_{1,0} \\ \neg\alpha_{1,0} \vee \neg w_{1,0} \vee \neg a_{1,0} \\ \neg\alpha_{1,0} \vee \neg a_{1,0} \\ \alpha_{1,0} \vee w_{1,0} \vee a_{1,0} \\ \alpha_{1,0} \vee \neg w_{1,0} \vee a_{1,0} \end{cases}$

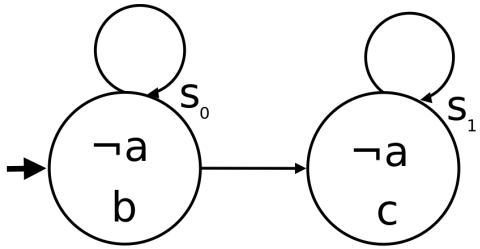


$$[E(T\ U \neg a)]_1^{[0,0]} = H(w_{0,0},\ w_{0,0}) \wedge (\alpha_{0,0} \ \vee \ \alpha_{1,0}) \wedge D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \wedge D(\alpha_{1,0}, [\neg a]_1^{1,0})$$

$$\alpha_{0,0} \ = \ [\neg a]_1^{[0,0]}$$
$$\alpha_{1,0} \ = \ [\neg a]_1^{[1,0]}$$

Descrevemos a tradução para a fórmula abaixo:

- $H(w_{0,0},\ w_{0,0}) = \begin{cases} \neg w_{0,0} \ \vee \ w_{0,0} \ \wedge \\ w_{0,0} \ \vee \ \neg w_{0,0} \end{cases}$
- $D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \ = \ \begin{cases} \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg a_{0,0} \\ \alpha_{0,0} \vee w_{0,0} \vee a_{0,0} \\ \alpha_{0,0} \vee \neg w_{0,0} \vee a_{0,0} \end{cases}$
- $D(\alpha_{1,0},\ [\neg a]_1^{1,0}) \ = \ \begin{cases} \neg\alpha_{1,0} \vee \neg w_{1,0} \vee \neg a_{1,0} \\ \neg\alpha_{1,0} \vee \neg w_{1,0} \vee \neg a_{1,0} \\ \neg\alpha_{1,0} \vee \neg a_{1,0} \\ \alpha_{1,0} \vee w_{1,0} \vee a_{1,0} \\ \alpha_{1,0} \vee \neg w_{1,0} \vee a_{1,0} \end{cases}$



$$[E(T\ U \neg a)]_1^{[0,0]} = H(w_{0,0},\ w_{0,0}) \wedge (\alpha_{0,0} \ \vee \ \alpha_{1,0}) \wedge D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \wedge D(\alpha_{1,0}, [\neg a]_1^{1,0})$$

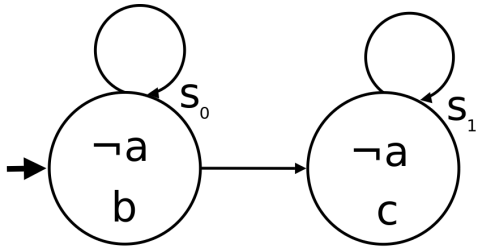
$$\alpha_{0,0} \ = \ [\neg a]_1^{[0,0]}$$
$$\alpha_{1,0} \ = \ [\neg a]_1^{[1,0]}$$

Renomeação

$$w_{0,0} \ = \ 1,\ w_{1,0} \ = \ 2,\ a_{0,0} \ = \ 3, a_{1,0} \ = \ 4,$$
$$t_0 \ = \ 5,\ t_1 \ = \ 6,\ t_2 \ = \ 7,\ \alpha_{0,0} \ = \ 8,\ \alpha_{1,0} \ = \ 9$$

Descrevemos a tradução para a fórmula abaixo:

- $H(w_{0,0},\ w_{0,0}) = \begin{cases} \neg w_{0,0} \ \vee \ w_{0,0} \ \wedge \\ w_{0,0} \ \vee \ \neg w_{0,0} \end{cases}$
- $D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \ = \ \begin{cases} \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg w_{0,0} \vee \neg a_{0,0} \\ \neg\alpha_{0,0} \vee \neg a_{0,0} \\ \alpha_{0,0} \vee w_{0,0} \vee a_{0,0} \\ \alpha_{0,0} \vee \neg w_{0,0} \vee a_{0,0} \end{cases}$
- $D(\alpha_{1,0},\ [\neg a]_1^{1,0}) \ = \ \begin{cases} \neg\alpha_{1,0} \vee \neg w_{1,0} \vee \neg a_{1,0} \\ \neg\alpha_{1,0} \vee \neg w_{1,0} \vee \neg a_{1,0} \\ \neg\alpha_{1,0} \vee \neg a_{1,0} \\ \alpha_{1,0} \vee w_{1,0} \vee a_{1,0} \\ \alpha_{1,0} \vee \neg w_{1,0} \vee a_{1,0} \end{cases}$



$$[E(T\ U \neg a)]_1^{[0,0]} = H(w_{0,0},\ w_{0,0}) \wedge (\alpha_{0,0} \ \vee \ \alpha_{1,0}) \wedge D(\alpha_{0,0},\ [\neg a]_1^{0,0}) \wedge D(\alpha_{1,0}, [\neg a]_1^{1,0})$$

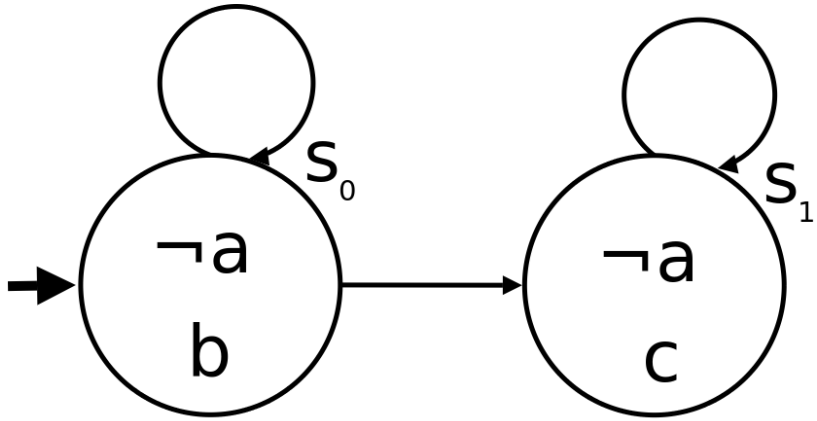
$$\alpha_{0,0} \ = \ [\neg a]_1^{[0,0]}$$
$$\alpha_{1,0} \ = \ [\neg a]_1^{[1,0]}$$

Renomeação

$$w_{0,0} \ = \ 1,\ w_{1,0} \ = \ 2,\ a_{0,0} \ = \ 3, a_{1,0} \ = \ 4,$$
$$t_0 \ = \ 5,\ t_1 \ = \ 6,\ t_2 \ = \ 7,\ \alpha_{0,0} \ = \ 8,\ \alpha_{1,0} \ = \ 9$$

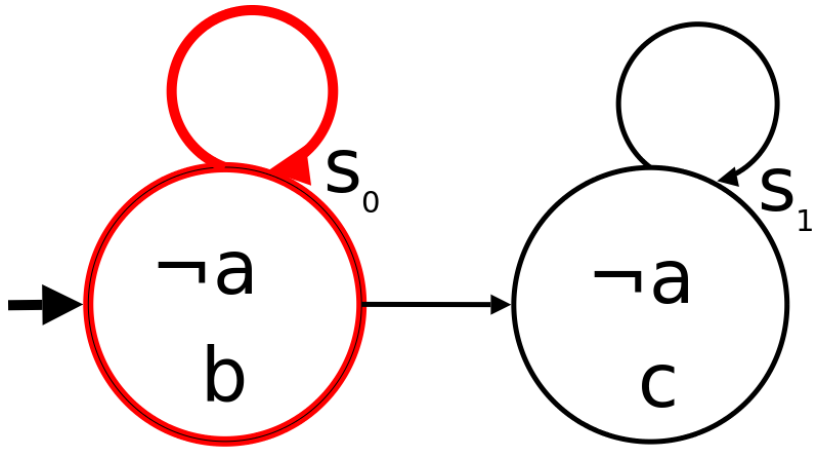
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9



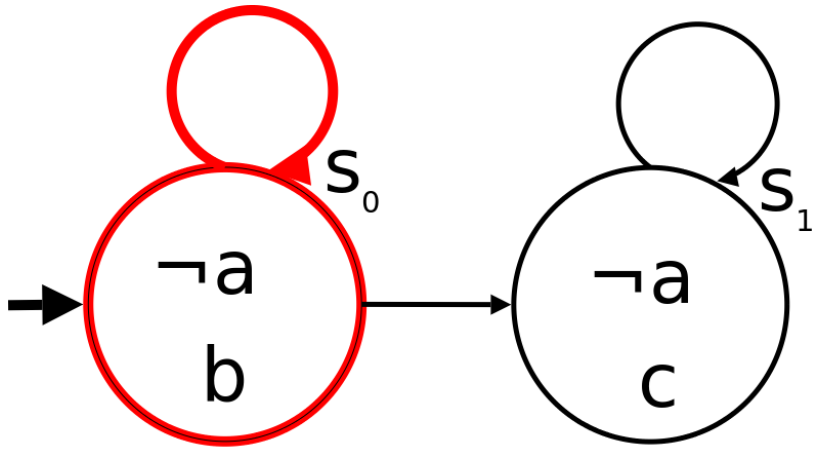
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9
- Possíveis modificações:



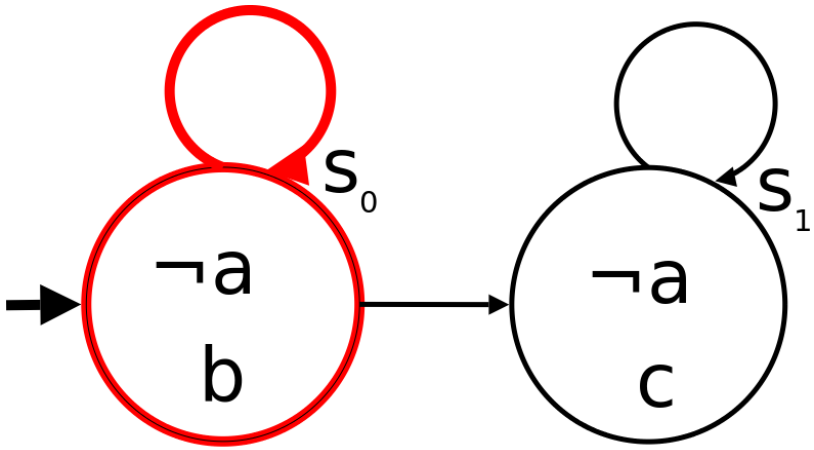
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9
- Possíveis modificações:
  1. Remover um estado



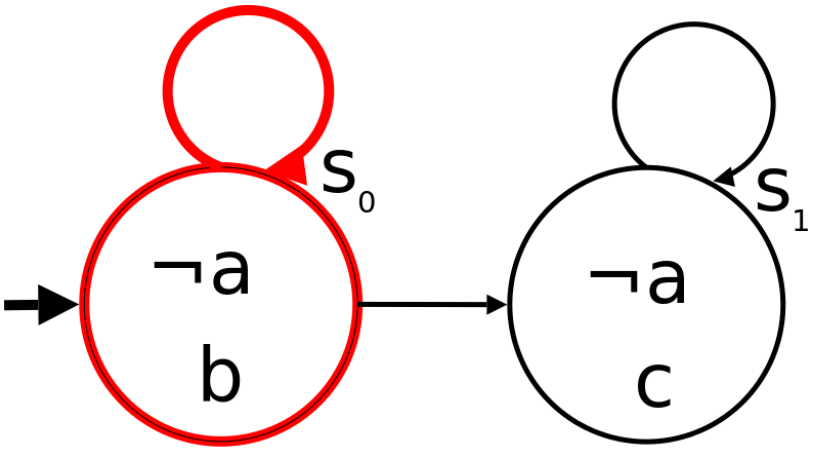
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9
- Possíveis modificações:
  1. Remover um estado
  2. Remover uma transição



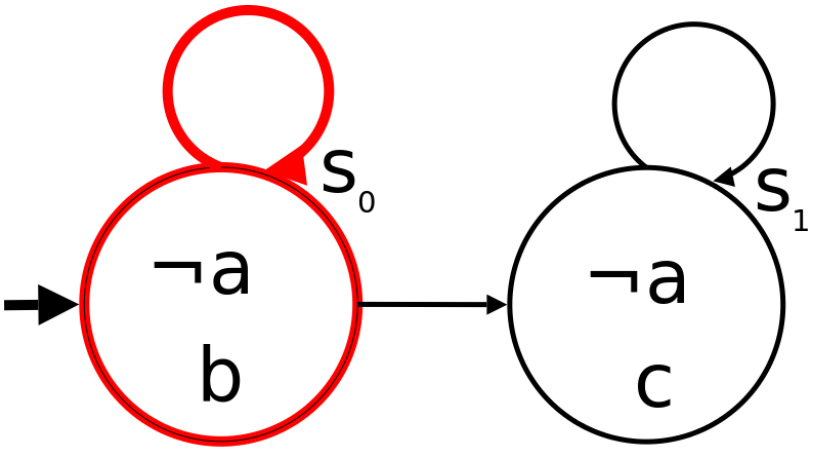
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9
- Possíveis modificações:
  1. Remover um estado
  2. Remover uma transição
  3. Modificar uma propriedade



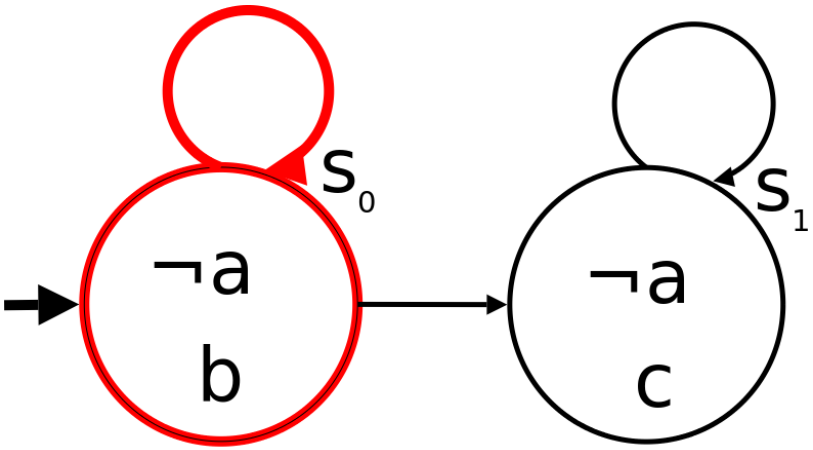
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9
- Possíveis modificações:
  1. Remover um estado
  2. Remover uma transição
  3. Modificar uma propriedade
- Escolhemos **modificar uma propriedade**



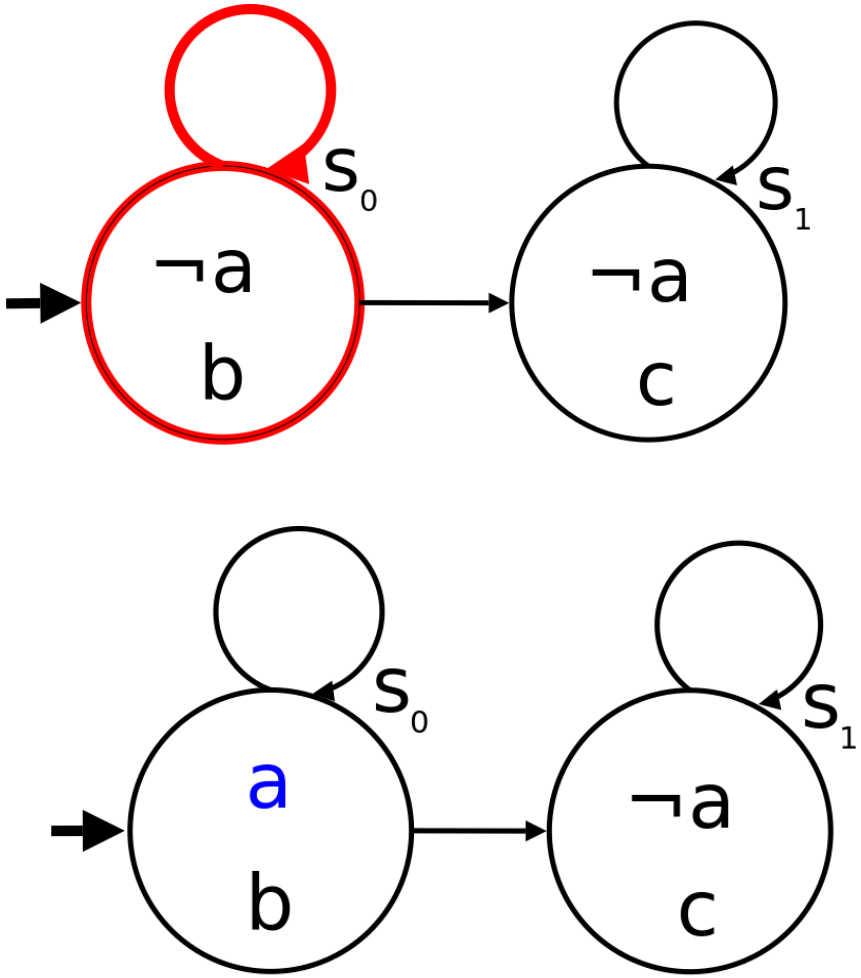
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9
- Possíveis modificações:
  1. Remover um estado
  2. Remover uma transição
  3. Modificar uma propriedade
- Escolhemos **modificar uma propriedade**
- Fórmula a ser acrescentada:
  $(w_{0,0} \vee a_{0,0}) \wedge (w_{1,0} \vee a_{1,0})$



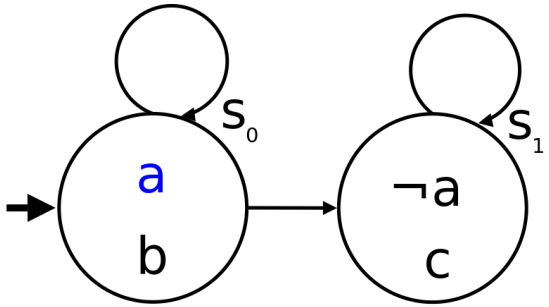
Primeira verificação

- Resultado do SAT:
  -1 -2 -3 4 5 -6 -7 8 -9
- Possíveis modificações:
  1. Remover um estado
  2. Remover uma transição
  3. Modificar uma propriedade
- Escolhemos **modificar uma propriedade**
- Fórmula a ser acrescentada:
  $(w_{0,0} \vee a_{0,0}) \wedge (w_{1,0} \vee a_{1,0})$



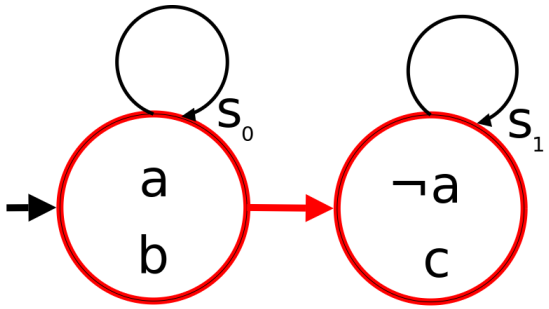
Segunda verificação

- Resultado do SAT:
  -1 2 3 -4 -5 6 -7 -8 9



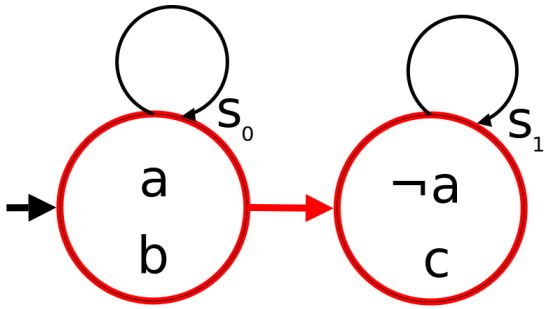
Segunda verificação

- Resultado do SAT:
  -1 2 3 -4 -5 6 -7 -8 9
- Escolhemos **remover uma transição**



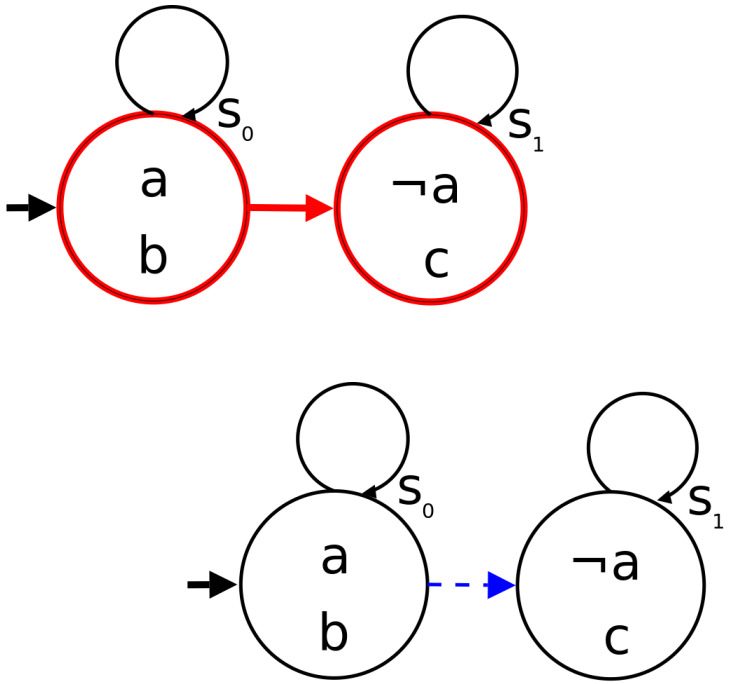
Segunda verificação

- Resultado do SAT:
  -1 2 3 -4 -5 6 -7 -8 9
- Escolhemos **remover uma transição**
- Fórmula a ser acrescentada:
  $(w_{0,0} \vee \neg w_{1,0} \vee \neg t_1)$



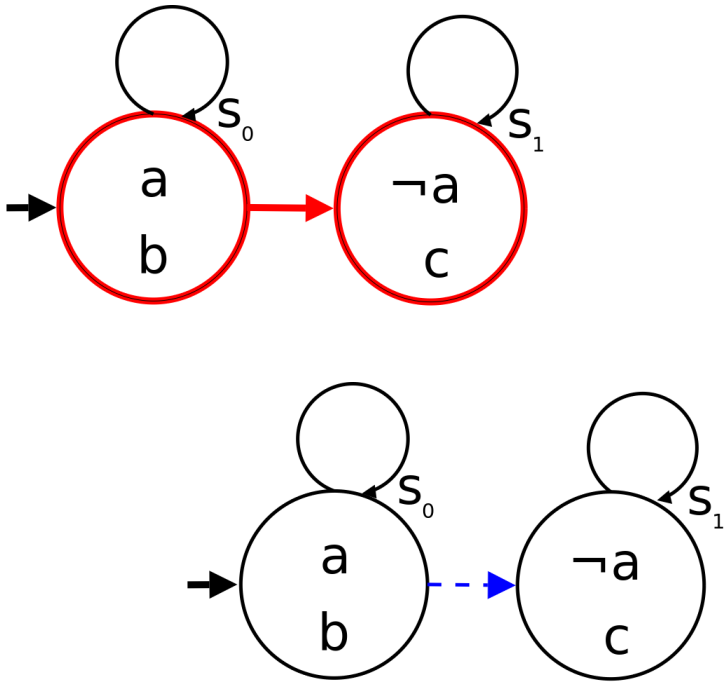
Segunda verificação

- Resultado do SAT:
  -1 2 3 -4 -5 6 -7 -8 9
- Escolhemos **remover uma transição**
- Fórmula a ser acrescentada:
  $(w_{0,0} \vee \neg w_{1,0} \vee \neg t_1)$
- Ou podemos **modificar uma propriedade**



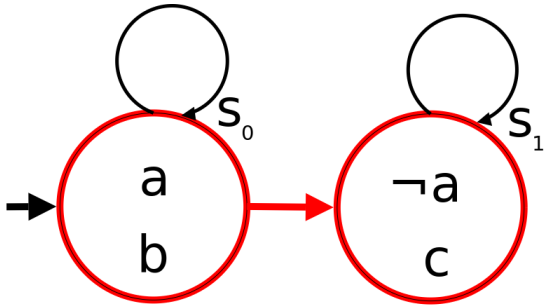
Segunda verificação

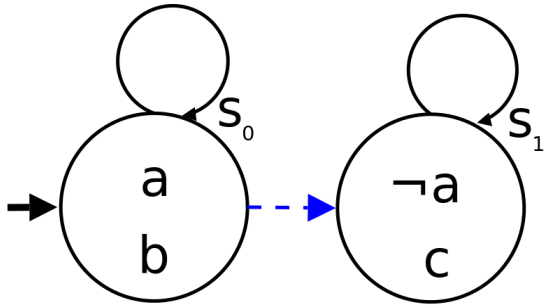
- Resultado do SAT:
  -1 2 3 -4 -5 6 -7 -8 9
- Escolhemos **remover uma transição**
- Fórmula a ser acrescentada:
  $(w_{0,0} \vee \neg w_{1,0} \vee \neg t_1)$
- Ou podemos **modificar uma propriedade**
- Fórmula a ser acrescentada:
  $(\neg w_{0,0} \vee a_{0,0}) \wedge (\neg w_{1,0} \vee a_{1,0})$

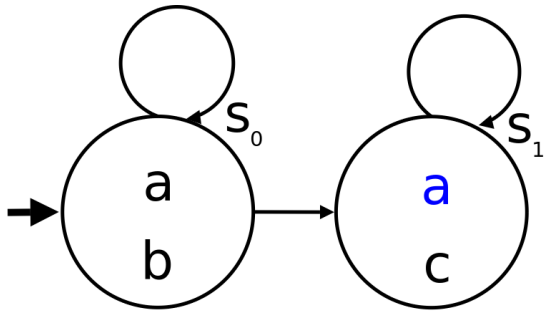


Segunda verificação

- Resultado do SAT:
  -1 2 3 -4 -5 6 -7 -8 9
- Escolhemos **remover uma transição**
- Fórmula a ser acrescentada:
  $(w_{0,0} \vee \neg w_{1,0} \vee \neg t_1)$
- Ou podemos **modificar uma propriedade**
- Fórmula a ser acrescentada:
  $(\neg w_{0,0} \vee a_{0,0}) \wedge (\neg w_{1,0} \vee a_{1,0})$
- Em ambos os casos chegamos em uma fórmula insatisfazível.

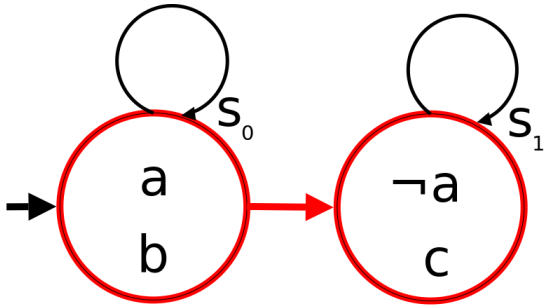


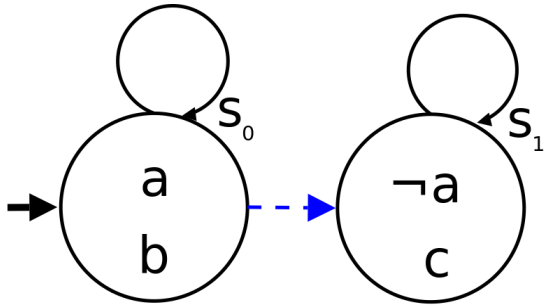


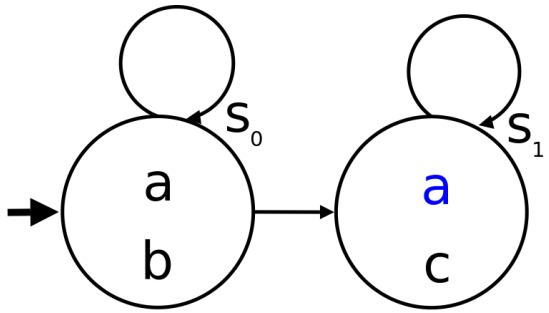


Segunda verificação

- Resultado do SAT:
  -1 2 3 -4 -5 6 -7 -8 9
- Escolhemos **remover uma transição**
- Fórmula a ser acrescentada:
  $(w_{0,0} \vee \neg w_{1,0} \vee \neg t_1)$
- Ou podemos **modificar uma propriedade**
- Fórmula a ser acrescentada:
  $(\neg w_{0,0} \vee a_{0,0}) \wedge (\neg w_{1,0} \vee a_{1,0})$
- Em ambos os casos chegamos em uma fórmula insatisfazível.







In the next chapters...

- Implementar o algoritmo;
- Implementar a entrada para a linguagem SMV;
- Implementar melhoras na tradução;
- Definir a escolha do k, e;
- Definir a geração de sugestões de forma automática.

Principais referências


C. E. Alchourròn, P. Gärdenfors, and D. Makinson.
On the logic of theory change: Partial meet contraction and revision functions.
Journal of Symbolic Logic, 50:510–530, 1985.


Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu.
Symbolic model checking without bdds.
Lecture Notes in Computer Science, 1579:193–207, 1999.


Edmund M. Clarke and E. Allen Emerson.
Design and synthesis of synchronization skeletons using branching-time temporal logic.
In *Logic of Programs, Workshop*, pages 52–71, London, UK, 1982. Springer-Verlag.


Marcelo Finger and Renata Wassermann.
Revising specifications with ctl properties using bounded model checking.
In *SBIA '08: Proceedings of the 19th Brazilian Symposium on Artificial Intelligence*, pages 157–166, Berlin, Heidelberg, 2008. Springer-Verlag.


Wojciech Penczek, Bozena Wozna, and Andrzej Zbrzezny.
Bounded model checking for the universal fragment of ctl.
Fundam. Inform., 51(1-2):135–156, 2002.