Algoritmos de Model checking com Autómatos

Problema: Dados uma fórmula φ em LTL, um sistema T e um estado s, verificar se $T, s \models \varphi$.

Abordagem:

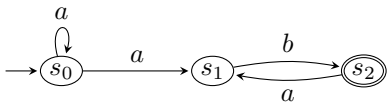
- Construir um autómato $\mathcal{A}_{\neg\varphi}$ que aceita um caminho π se e só se $\pi \models \neg\varphi$, i.e. $\pi \not\models \varphi$.
- Representar (T, s) por um autómato $\mathcal{A}_{T,s}$ (que aceita exactamente os caminhos em T que começam em s).
- Verificar se $\mathcal{L}(\mathcal{A}_{\neg\varphi}) \cap \mathcal{L}(\mathcal{A}_{T,s}) = \emptyset$. Em caso afirmativo, tem-se $T, s \models \varphi$, caso contrário, qualquer caminho pertencente à intersecção das duas linguagens pode ser exibido como contra-exemplo.

Autómatos não determinísticos

Σ alfabeto, $L \subseteq \Sigma^\star$ linguagem
$\mathcal{A} = (\Sigma, S, S^0, \delta, F)$ onde $S^0, F \subseteq S$, $\delta : S \times \Sigma \to 2^S$

$$L(\mathcal{A}) = \{w \in \Sigma^\star \mid \delta(s_0, w) \cap F \neq \emptyset \wedge s_o \in S^0\}$$

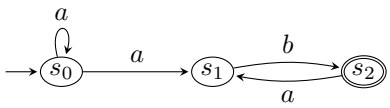
onde $\delta(s, \varepsilon) = s$ e $\delta(s, ax) = \delta(\delta(s, a), x)$.



$$L((a^*ab(ab)^*)$$

As linguagens aceites por NFAs são fechadas para o complementar, reunião e intersecção.

Autómato (não-determinístico) de Büchi



$w = aaababab\ldots$ é aceite

Autómato (não-determinístico) de Büchi

Palavras infinitas

Dado um alfabeto Σ uma palavra infinita de Σ é uma sequência infinita $a_0a_1a_2\ldots a_ma_{m+1}\ldots$ de símbolos $a_i \in \Sigma$.

Σ^ω é o conjunto das palavras infinitas de Σ.

$aaabbbaaabbb\ldots$ representa-se pela r.e $(aaabbb)^\omega$

Autómato de Büchi

$\mathcal{A}_\omega = (\Sigma, S, S^0, \delta, F)$ onde $S^0, F \subseteq S$, $\delta : S \times \Sigma \to 2^S$

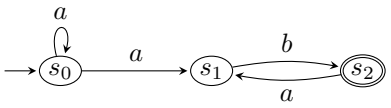
Dado $w = a_0a_1\ldots$, uma computação em w (r_w) é uma sequência de estados $s_0, s_1, \cdots$, com $s_0 \in S^0$ e $s_{i+1} \in \delta(s_i, a_i)$, $i \geq 0$.

$$lim(r_w) = \{s \mid s = s_i \text{ para um número infinito de } i's\}$$

(estados que aparecem em r_w um número infinito de vezes)

$$L(\mathcal{A}_\omega) = \{w \in \Sigma^\omega \mid \exists r_w\ lim(r_w) \cap F \neq \emptyset\}$$

Autómato (não-determinístico) de Büchi



$w = aaababab\ldots$ é aceite

$$L_\omega((a^*(ab)^\omega))$$

Fechos e caracterização

Teorema 17.1. *As linguagens aceites por autómatos de Büchi são fechadas para a reunião, intersecção e complementar.*

- Para a união usa-se a construção do autómato produto
- não funciona para a interseção
- sendo $\mathcal{A}_i = (\Sigma, S_i, S_i^0, \delta_i, F_i)$,

  $$\mathcal{A}_\cap = (\Sigma, S_1 \times S_2 \times \{1,2\}, S_1^0 \times S_2^0 \times \{1\}, \delta_\cap, F_1 \times F_2 \times \{1\})$$

  e $(s', t', j) \in \delta(s, t, i)$ se $s' \in \delta_1(s, a)$, $t' \in \delta_2(t, a)$, e $i = j$ mas se $i = 1$ e $s \in F_1$, então $j = 2$ e se $i = 2$ e $t \in F_2$, então $j = 1$.
- o complementar tem um algoritmo duplamente exponêncial

Fechos e caracterização

Neste caso o determinismo é mais fraco que o não determinismo: $(0+1)^*1^\omega$ não é aceite por um autómato deterministico de Büchi.

Teorema 17.2. *Uma ω-linguagem L é reconhecida por um autómato de Büchi sse L é a união finita de linguagens VW^ω, onde $V, W \subseteq \Sigma^\star$ são linguagens regulares e $\varepsilon \notin W$.*

$L_\omega(\mathcal{A}_\omega) = \emptyset$ **?** **e** $L_\omega(\mathcal{A}_\omega) = \Sigma^\omega$

Teorema 17.3. *O problema de determinar se a linguagem aceite por um $\mathcal{A}_\omega$ é vazia pode ser decidido em tempo linear.*

Demonstração. Dado $\mathcal{A}_\omega = (\Sigma, S, S^0, \delta, F)$, $L_\omega(\mathcal{A}_\omega)$ é não vazia sse existe $s_0 \in S^0$ e $t \in F$ tal que s_0 é conexo com t e t é conexo com ele próprio. □

Teorema 17.4. *O problema de determinar se a linguagem aceite por um $\mathcal{A}_\omega$ é Σ^ω pode ser decidido em tempo exponencial.*

Demonstração. Obter o complementar de um autómato de Büchi é exponencial e $L_\omega(A) \neq \Sigma^\omega \leftrightarrow L_\omega(\overline{A}) \neq \emptyset$. □

LTL e Autómatos de Büchi

- A cada fórmula φ do LTL podemos associar um autómato de Büchi φ, $\mathcal{A}_\varphi$,
- A linguagem $L_\omega(\mathcal{A}_\varphi)$ é precisamente o conjunto de caminhos que satisfazem a fórmula φ.
- Considerando o alfabeto $\Sigma = 2^{AP}$ não é difícil associar a fórmulas do LTL autómatos de Büchi
- ... mas é muito pouco eficiente computacionalmente.
- Para φ fórmula proposicional seja

$$\Sigma_\varphi = \{a \in \Sigma \mid a \models \varphi\}$$

.

-

LTL e Autómatos de Büchi

Por exemplo, se $p \in AP$,

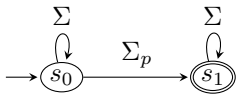
$$\begin{aligned} \Sigma_p &= \{a \in \Sigma \mid p \in a\} \\ \Sigma_{\neg p} &= \Sigma \setminus \Sigma_p \\ \Sigma_{p \wedge q} &= \Sigma_p \cap \Sigma_q \\ \Sigma_{p \vee q} &= \Sigma_p \cup \Sigma_q \end{aligned}$$

Temos que dados estados s, s'

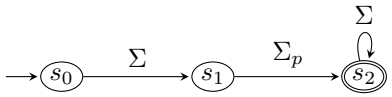
$$s \xrightarrow{\Sigma_\varphi} s' = \{s \xrightarrow{a} s' \mid a \in \Sigma_\varphi\}$$

LTL e Autómatos de Büchi

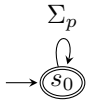
Fp



Xp

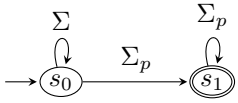


Gp

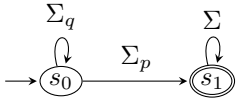


LTL e Autómatos de Büchi

FGp



pUq



Como estes autómatos são fechados para a reunião, interseção e complementar, podem ser construídos para qualquer fórmula. Mas os autómatos para a negação levam a uma explosão combinatória...

LTL e Autómatos

- Para simplificar a exposição e por serem exponencialmente mais sucintos do que os NFAs vamos introduzir os Autómatos alternados.
- Num NFA se há uma escolha entre dois estados isso pode ser representado por uma disjunção.
- Os autómatos alternados generalizam esse conceito a outras fórmulas Booleanas.

Autómato Finito Alternado (AFA)

- Seja X um conjunto de variáveis e $\mathcal{B}^+(X)$ o conjunto das fórmulas com $\wedge$ e $\vee$ sobre X (mais true e false).
- Dado $Y \subseteq X$ dizemos que Y *satisfaz uma fórmula* θ se a atribuição de valores às variáveis que atribuí 1 a elementos de Y e 0 aos restantes, satisfaz θ.
- $\{s_1, s_3\}$ satisfaz $(s_1 \vee s_2) \wedge (s_3 \vee s_4)$
- $\delta(s_0, a) = (s_1 \wedge s_2) \vee (s_3 \wedge s_4)$ significa que o autómato aceita aw de s_0, se aceita w de s_1 e de s_2 ou aceita w de s_3 e de s_4.
- Neste caso há uma escolha existencial (disjunção) e uma universal (conjunção).
- Dado um NFA podemos representar δ usando $\mathcal{B}^+(S)$. Por exemplo $\delta(s_0, a) = \{s_1, s_0\} = s_1 \vee s_0$. Num AFA, o valor de δ pode ser uma fórmula qualquer de $\mathcal{B}^+(S)$.

Autómato Finito Alternado (AFA)

Exemplo 17.1. $A = (\Sigma, S, s^0, \delta, F)$ *onde* $s^0 \in S$, *estado inicial* ,$F \subseteq S$ *estados finais,* $\delta : S \times \Sigma \to \mathcal{B}^+(S)$

$\mathcal{A} = (\{a, b\}, \{s_0, s_1\}, \delta, s_0, \{s_1\})$

δ	a	b
s_0	s_1	$s_0 \vee s_1$
s_1	$s_0 \wedge s_1$	s_0

Computação (run)

- Uma computação agora é uma árvore r (S-etiquetada) com raiz, ε (e finita).
- O nível de um nó x ($|x|$) é a distância do nó à raíz e $r(x)$ a sua etiqueta (um estado)

- Um ramo $\beta = x_0 x_1 \cdots$ é uma sequência maximal de nós onde $x_0 = \varepsilon$ e cada x_i é pai de x_{i+1}, $i \geq 0$.
- E $r(\beta) = r(x_0)r(x_1)\cdots$ é a sequência de etiquetas ao longo de β.
- Uma computação numa palavra $w = a_0 \cdots a_n$ é uma árvore finita S-etiquetada, r_w tal que $r(\varepsilon) = s^0$ e:
  - se $|x| = i < n$, $r(x) = s$ e $\delta(s, a_i) = \theta$ então x tem k filhos $x_1, \ldots, x_k$ para algum $k \leq |S|$, e $\{r(x_1), \ldots r(x_k)\}$ satisfaz θ.
- Notar que se $\delta(r(x), a_i) = \mathsf{true}$ então x não necessita de ter filhos (e se $\delta(s, a_i) = \mathsf{false}$ não há nenhum nó x com $r(x) = s$).

Autómato Finito Alternado (AFA)

Computação de aceitação

Uma computação é de aceitação se todos os nós à profundidade (nível máximo) n são etiquetados com elementos de F . Ou um ramo termina com true ou atinge um estado final.

Linguagem aceite por um AFA

$$L(A) = \{w \mid \exists r_w \text{ de aceitação } \}$$

Exemplo 17.2. *$A = (\Sigma, S, s^0, \delta, F)$ onde $s^0 \in S$, estado inicial ,$F \subseteq S$ estados finais, $\delta : S \times \Sigma \to \mathcal{B}^+(S)$*

$\mathcal{A} = (\{a, b\}, \{s_0, s_1\}, \delta, s_0, \{s_1\})$

δ	a	b
s_0	s_1	$s_0 \vee s_1$
s_1	$s_0 \wedge s_1$	s_0

AFA vs. NFA

Teorema 17.5 (Equivalência AFA e NFA)**.** *Uma linguagem é aceite por um NFA se e só se é aceite por um AFA.*

$\Rightarrow$

Dado NFA $\mathcal{A} = (\Sigma, S, S^0, \delta, F)$, constrói-se $\mathcal{A}_a = (\Sigma, S \cup \{s^0\}, s^0, \delta_a, F)$ e para $b \in \Sigma$

$$\delta_a(s^0, b) = \bigvee_{t \in S^0, t' \in \delta(t,b)} t'$$

$$\delta_a(s, b) = \bigvee_{t \in \delta(s,b)} t$$

A disjunção vazia é false.

AFA vs. NFA

Teorema 17.6 (Equivalência AFA e NFA)**.** *Uma linguagem é aceite por um NFA se e só se é aceite por um AFA.*

$\Leftarrow$

Dado AFA $\mathcal{A} = (\Sigma, S, s^0, \delta, F)$. Então $\mathcal{A}_n = (\Sigma, 2^S, \{\{s^0\}\}, \delta_n, 2^F)$ e para $a \in \Sigma$

$$\delta_n(T, a) = \{T' \mid T' \text{ satisfaz } \ \wedge_{\ t \in T} \delta(t, a)\}$$

Se a conjunção for vazia é true.

Exercício:

Determina um autómato finito alternado que aceita a mesma linguagem como o autómato finito não determinístico seguinte:

$$\mathcal{A}_\omega = (\{a, b\}, \{s_0, s_1, s_2\}, \{s_0\}, \delta, \{s_1\}),$$

onde $\delta(s_0, a) = \{s_0, s_2\}$, $\delta(s_1, a) = \{s_2\}$, $\delta(s_2, a) = \{s_0, s_2\}$, $\delta(s_0, b) = \{s_1, s_2\}$, $\delta(s_1, b) = \{s_0\}$ e $\delta(s_2, b) = \{s_0, s_2\}$.

Exercício:

Determina um autómato finito não determinístico que aceita a mesma linguagem como o autómato finito alternado seguinte: $\mathcal{A} = (\{a, b\}, \{s_0, s_1\}, \delta, s_0, \{s_1\})$

δ	a	b
s_0	s_1	$s_0 \vee s_1$
s_1	$s_0 \wedge s_1$	s_0

AFA Complementar

Complementar

Se $\theta \in \mathcal{B}^+(S)$ a seu dual $\overline{\theta}$ é obtido trocando true por false e $\vee$ por $\wedge$. Dado um AFA $A = (\Sigma, S, s^0, \delta, F)$, seja $\overline{A} = (\Sigma, S, s^0, \overline{\delta}, S - F)$ e $\overline{\delta}(s, a) = \overline{\delta(s, a)}$. Então $L(\overline{A}) = \Sigma^\star \setminus L(A)$.

onde:

- $\overline{x} = x$
- $\overline{\mathsf{true}} = \mathsf{false}$ e $\overline{\mathsf{false}} = \mathsf{true}$
- $\overline{\alpha \ \wedge \ \beta} = \overline{\alpha} \ \vee \ \overline{\beta}$
- $\overline{\alpha \ \vee \ \beta} = \overline{\alpha} \ \wedge \ \overline{\beta}$

Autómato Alternado de Büchi

Seja $A = (\Sigma, S, s^0, \delta, F)$ um autómato alternado em palavras infinitas, onde $s^0 \in S$ é o estado inicial , $F \subseteq S$ estados finais, $\delta : S \times \Sigma \to \mathcal{B}^+(S)$.

- Uma computação de A numa palavra infinita $w = a_0 a_1 \ldots$ é uma árvore S-etiquetada r (possivelmente infinita) tal que $r(\varepsilon) = s^0$ e:
- se $|x| = i$, $r(x) = s$ e $\delta(s, a_i) = \theta$, então x tem k filhos $x_1, \ldots, x_k$ com $k \leq |S|$ e $\{r(x_1), \ldots, r(x_k)\}$ satisfaz θ.
- Uma computação é de aceitação se todo o ramo infinito de r inclui um número infinito de etiquetas de F. Se $\delta(s, a_i) = \mathsf{true}$, x não tem filhos. Uma computação finita é de aceitação.

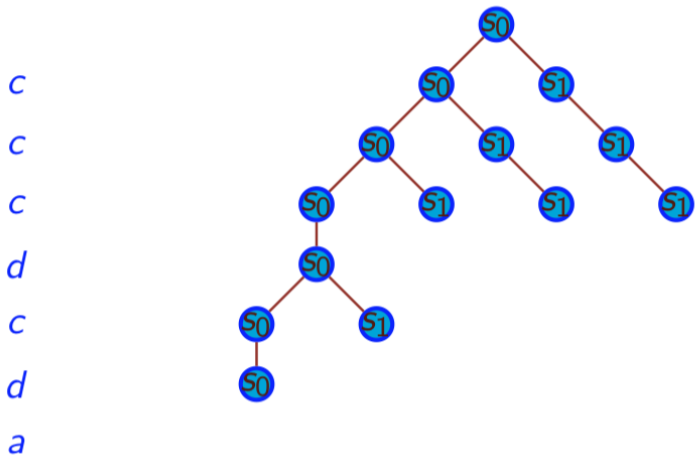
$$L_\omega(A) = \{w \in \Sigma^\omega \mid \text{existe uma computação de aceitação } r \text{ para } w\}$$

Exemplo

$\mathcal{A} = (\{a, b, c, d\}, \{s_0, s_1, s_2\}, \delta, s_0, \{s_1\})$

δ	a	b	c	d
s_0	true	true	$s_0 \wedge (s_1 \vee s_2)$	s_0
s_1	false	true	s_1	true
s_2	true	s_1	true	s_2

- Determina uma computação sobre a palavra $cccdcdad\ldots$
- Determina uma computação de aceitação e outra de não aceitação para a palavra $caccccc\ldots$



Autómatos Alternados de Büchi e Autómatos de Büchi

Teorema 17.7 (Equivalência ABA e NBA)**.** *Uma linguagem é aceite por um NBA se e só se é aceite por um ABA.*

$\Leftarrow$

Dado ABA $\mathcal{A} = (\Sigma, S, s^0, \delta, F)$. Então $\mathcal{A}_n = (\Sigma, 2^S \times 2^S, \{(\{s^0\}, \emptyset)\}, \delta_n, \{\emptyset\} \times 2^F)$ e para $a \in \Sigma$. Para $U \neq \emptyset$

$$\begin{aligned}
\delta_n((U,V),a) &= \{(U',V') \mid \exists X, Y \subseteq S,\ X \text{ satisfaz } \ \wedge\ _{t\in U}\delta(t,a) \\
&\quad\ Y \text{ satisfaz } \ \wedge\ _{t\in V}\delta(t,a), \\
&\quad\ U' = X \setminus F \wedge V' = Y \cup (X \cap F)\} \\
\delta_n((\emptyset,V),a) &= \{(U',V') \mid \exists Y \subseteq S, \\
&\quad\ Y \text{ satisfaz } \ \wedge\ _{t\in V}\delta(t,a), \\
&\quad\ U' = Y \setminus F \wedge V' = Y \cap F\}
\end{aligned}$$

Exercícios

Exercício 17.1. *Considera o autómato alternado de Büchi* $\mathcal{A} = (\{a,b,c,d\}, \{s_0,s_1,s_2,s_3,s_4\}, \delta, s_0, \{s_4\})$*, onde*

δ	a	b	c	d
s_0	$s_2 \wedge s_3$	$s_1 \wedge s_3$	$s_1 \vee s_2$	s_0
s_1	s_4	s_1	s_1	s_0
s_2	s_2	s_4	s_2	s_0
s_3	s_3	s_3	s_3	s_0
s_4	s_4	s_4	s_4	s_0

Indica palavras ω_1 *e* ω_2 *tal que* $\omega_1 \in L(\mathcal{A})$ *e* $\omega_2 \notin L(\mathcal{A})$ *juntamente com uma computação de aceitação para* ω_1 *e uma de não aceitação para* ω_2*. Descreve informalmente* $L(\mathcal{A})$*.*

$\diamond$

Satisfazibilidade do LTL

Satisfazibilidade

Dado um modelo $T = (S, \longrightarrow, AP, L)$ e um caminho $\pi = s_0 \longrightarrow \ldots$, define-se a relação de satisfazibilidade $\models$ indutivamente por:

1. $\pi \models p$ sse $p \in L(s_1)$
2. $\pi \models \neg\varphi$ sse $\pi \not\models \varphi$
3. $\pi \models \varphi \ \wedge\ \psi$ sse $\pi \models \varphi$ e $\pi \models \psi$
4. $\pi \models \mathrm{X}\varphi$ sse $\pi[1...] \models \varphi$

5. $\pi \models \varphi \mathrm{U} \psi$ sse $\exists i \geq 0, \pi[i...] \models \psi$ e $\forall\, 0 \leq j < i,\ \pi[j...] \models \varphi$

Um caminho $\pi = s_0 s_1 s_2 \cdots$ tem associado um traço σ que é uma palavra infinita sobre $\Sigma = 2^{AP}$:

$$L(s_0)L(s_1)L(s_2)\cdots$$

Vamos ver que o conjunto de caminhos que satisfazem uma fórmula são os aceites por um autómato sobre palavras infinitas.

LTL e Autómatos Alternados de Büchi

Teorema 17.8. *Dada uma fórmula φ do LTL podemos construir um autómato alternado de Büchi $A_\varphi = (2^{AP}, S, \varphi, \delta, F)$ tal que $|S| = \mathcal{O}(|\varphi|)$ e tal que $L_\omega(A_\varphi)$ é exactamente o conjunto de caminhos que satisfazem a fórmula φ.*

- Seja S o conjunto de todas as subfórmulas de φ e das suas negações.
- Seja F o conjunto de todas as fórmulas de S da forma $\neg(\psi \mathrm{U} \varphi)$
- Considere-se o dual duma fórmula, estendido a negações $\overline{\neg\varphi} = \varphi$ e $\overline{\varphi} = \neg\varphi$. Então para $A \in 2^{AP}$
  - $\delta(p, A) = \mathsf{true}$ se $p \in A$
  - $\delta(p, A) = \mathsf{false}$ se $p \notin A$
  - $\delta(\varphi \ \wedge\ \psi, A) = \delta(\varphi, A) \ \wedge\ \delta(\psi, A)$
  - $\delta(\neg\varphi, A) = \overline{\delta(\varphi, A)}$
  - $\delta(\mathrm{X}\varphi, A) = \varphi$
  - $\delta(\varphi \mathrm{U} \psi, A) = \delta(\psi, A) \ \vee\ (\delta(\varphi, A) \ \wedge\ \varphi \mathrm{U} \psi)$

LTL e Autómatos Alternados de Büchi

Seja r uma computação de A_φ. Os ramos infinitos a partir de certa altura ou são etiquetados por $\varphi \mathrm{U} \psi$ ou por $\neg(\varphi \mathrm{U} \psi)$. Temos que

$$\delta(\neg(\varphi \mathrm{U} \psi), A) = \overline{\delta(\psi, A)} \ \wedge\ (\overline{\delta(\varphi, A)} \ \vee\ \neg(\varphi \mathrm{U} \psi))$$

então se um ramo for etiquetado por $\neg(\varphi \mathrm{U} \psi)$ isso garante que $\varphi \mathrm{U} \psi$ não se verifica a partir daí (porque ψ não é satisfeito). Por isso esses são estados finais.

Por outro lado para os ramos etiquetados por $\varphi \mathrm{U} \psi$ não é garantido que $\varphi \mathrm{U} \psi$ se verifique pois não há garantia que ψ vá ser satisfeito.

A demonstração do teorema segue por indução na estrutura da fórmula φ, mostrando que as palavras π aceites por A_φ são os caminhos que satisfazem φ, i.e $\pi \models \varphi$.

Demonstração. Caso base: se $\varphi = \mathsf{true}$ (resp. false) a trasição por qualquer letra é true (resp. false) e portanto todas as palavras são aceites (não aceites). Se $\varphi = p$ (resp. $\varphi = \neg\varphi$), $\pi \models \varphi$ se e só se $p \in \pi[1]$ ($p \notin \pi[1]$). Uma computação em A_φ é uma palavra com uma só letra (φ) e é de aceitação. Por definição de δ, existe uma computação sobre π se e só se $p \in \pi[1]$. Caso $\varphi = \mathrm{X}\psi$. Pela definição de A_φ, o autómato é idêntico ao de A_ψ, apenas com mais um estado (que é o inicial) e é etiquetado por φ e uma transição $\delta(\varphi, A) = \varphi$ para todas as letras A. Temos que $\pi \models \mathrm{X}\varphi$ sse $\pi[1...] \models \varphi$ sse existe uma computação de aceitação r_1 de A_ψ sobre $\pi[1...]$ sse $r = \varphi \cdot r_1$ (árvore com raiz etiquetada por φ) é uma computação de aceitação de A_φ com π. Caso $\varphi = \psi_1 \wedge \psi_2$. Temos que $\pi \models \psi_1 \wedge \psi_2$ sse $\pi \models \psi_1$ e $\pi \models \psi_2$ sse, por hipótese de indução, existe uma computação de aceitação r_i de A_{ψ_i} com π para $i = 1, 2$. Seja $r_1 \oplus r_2$ a árvore cuja raiz é etiquetada por φ, os nós são os de r_1 e os de r_2, sendo as respectivas raizes nós filhos do nó etiquetado por φ. Temos que $r_1 \oplus r_2$ é uma computação de A_φ dado π e qualquer computação de A_φ para π pode-se decompor em $r_1 \oplus r_2$. A computação $r_1 \oplus r_2$ é de aceitação sse todos os ramos infniros incluem um número infinito de etiquetas de estados finais sse todos os ramos de r_1 e de r_2 contêm um número infinito de etiquetas de estados finais sse r_1 e r_2 são de aceitação.

Caso $\varphi = \psi_1 \mathrm{U} \psi_2$. Suponhamos que $\pi \models \psi_1 \mathrm{U} \psi_2$. Então, $\exists i \geq 0, \pi[i...] \models \psi_2$ e $\forall\, 0 \leq j < i, \pi[j...] \models \psi_1$. Mostramos por indução em i , que existe uma computação de aceitação para π em A_φ. Se $i = 0$ então $\pi \models \psi_2$ e existe uma computação de aceitação r_2 em A_{ψ_2} para π. Esta computação r_2 é também uma computação de aceitação de A_φ pela definição de δ. Se $i > 0$ então $\pi \models \psi_1$ e $\pi[1...] \models \psi_1 \mathrm{U} \psi_2$. Pelas hipóteses de indução, existem computações de aceitação r_1 de A_{ψ_1} para π e r_2 de A_φ para $\pi[1...]$ então $(1 \cdot r_2) \oplus r_1$ é uma computação de aceitação de A_φ para π. Inversamente, se existe uma computação de aceitação r de A_φ para π, r não contém nenhum ramo etiquetadi exclusivamente por φ. Seja n o nível maximal dum nó etiquetado por φ. Mostramos por indução em n que $\pi \models \varphi$. Se $n = 0$ pela definição de δ se se substituir φ por ψ_2 na raiz de r obtem-se também uma computação de aceitação. Por hipótese de indução, $\pi \models \psi_2$ e portanto $\pi \models \varphi$. Se $n \geq 1$, pela definição de δ, r decompõe-se em $r = 1 \cdot r_1 \oplus r_2$, onde r_1 é uma computação de aceitação de A_φ para $\pi[1 \ldots]$ e r_2 de A_{ψ_1} para π. Pelas hipóteses de indução $\pi \models \psi_1$ e e $\pi[1 \ldots] \models \varphi$, logo $\pi \models \varphi$. Caso $\varphi = \neg(\psi_1 \mathrm{U} \psi_2)$. Se $\pi \models \neg(\psi_1 \mathrm{U} \psi_2)$, ou $\forall i,\ \pi[i \ldots] \models \neg\psi_2$ ou existe um j tal que $\pi[j \ldots] \models \neg\psi_1 \wedge \neg\psi_2$ e para todos $i \leq j$, $\pi[i \ldots] \models \neg\psi_2$. No primeiro caso, por hipótese de indução, e para todo i existe r_i computação de aceitação de $A_{\neg\psi_2}$ para $\pi[i \ldots]$. Podemos construir r da seguinte forma: $1^i \cdot r_i$ e $r(1^i) = \varphi$. E r é de aceitação. No segundo caso, procede-se por indução em j. Se $j = 0$, $\pi \models \neg\psi_1 \wedge \neg\psi_2$ e por hipótese de indução existem computações de aceitação r_i em $A_{\neg\psi_i}$ para π e $i = 1, 2$. Podemos construir uma computação para A_φ como no caso da conjunção. Se $j \geq 1$ constrói-se uma computação de aceitação para $\pi[1 \ldots]$ e duma computação de aceitação de $A_{\neg\psi_2}$. Inversamente se π é aceite por A_φ, seja r uma computação de aceitação para π. Pela definição de δ ou r tem uma subárvore r_1, computação de aceitação de $A_{\neg\psi_1}$ ou só tem subárvores

r_2 (computação de aceitação de $A_{\neg\psi_2}$). Em ambos os casos, $\pi \models \varphi$. □

LTL e Autómatos Alternados de Büchi

Corolário 17.1. *Dada uma fórmula φ do LTL pode-se construir um autómato de Büchi A_φ tal que $L(A_\varphi)$ é exactamente o conjunto de caminhos que satisfazem a fórmula φ.*

Exercício 17.2. *Para $\varphi = Xp$, determina o autómato alternado de Büchi A_φ. Determina a conjunto de caminhos aceites por A_φ*

◇

Exemplo

Seja $\varphi = (\mathrm{X}\neg p)\mathrm{U}q$.

$$A_\varphi = (2^{\{p.q\}}, \{\varphi, \neg\varphi, \mathrm{X}\neg p, \neg\mathrm{X}\neg p, \neg p, p, \neg q, q\}, \varphi, \delta, \{\neg\varphi\})$$

s	$\{p,q\}$	$\{p\}$	$\{q\}$	$\emptyset$
φ	true	$\neg p \wedge \varphi$	true	$\neg p \wedge \varphi$
$\neg\varphi$	false	$p \vee \neg\varphi$	false	$p \vee \neg\varphi$
$\mathrm{X}\neg p$	$\neg p$	$\neg p$	$\neg p$	$\neg p$
$\neg\mathrm{X}\neg p$	p	p	p	p
$\neg p$	false	false	true	true
p	true	true	false	false
q	true	false	true	false
$\neg q$	false	true	false	true

No estado φ se q não se verifica então $\neg p$ e φ tem de se verificar no estado seguinte. Como $\varphi \notin F$, A_φ terá de chegar a um estado tal que q se verifica.

Podemos estender a função de transição a outras conectivas:

$$\begin{aligned} \delta(\mathrm{G}\varphi, A) &= \delta(\varphi, A) \wedge \mathrm{G}\varphi \\ \delta(\mathrm{F}\varphi, A) &= \delta(\varphi, A) \vee \mathrm{F}\varphi \end{aligned}$$

Modelos e Autómatos de Büchi

Seja um sistema de transições (modelo) $T = (S, \longrightarrow, AP, L)$, para um conjunto de variáveis proposicionais AP . Um caminho $\pi = s_0 s_1 \ldots$ pode ser visto como uma sequência de subconjuntos de AP.

Dado um modelo T e $s_0 \in S$, associamos um autómato de Büchi

$$A_M = (2^{AP}, S, \{s_0\}, \delta, S),$$

tal que $s' \in \delta(s, A)$ sse $s \longrightarrow s'$ e $A = L(s)$.

Como o conjunto de estados finais coincide com S, qualquer caminho π é uma computação de aceitação e então $L_\omega(A_T)$ coincide com o conjunto de caminhos de T.

Exercícios

Exercício 17.3. *a) Para $\varphi = a\ U\ b$ determina o autómato alternado de Büchi $A_{\neg\varphi}$. Determina o conjunto de caminhos aceites por $A_{\neg\varphi}$.*

b) Repite a alínea anterior para a fórmula $a \wedge Fb$ (lembra-te de que $F\varphi \equiv \mathsf{true} U \varphi$).

$\diamond$

Exercícios

Exercício 17.4. *Considera o modelo $T = (S, \to, L)$ com*

- $S = \{q_1, q_2, q_3, q_4\}$,
- $\to = \{q_1 \to q_2, q_1 \to q_4, q_2 \to q_4, q_3 \to q_2, q_3 \to q_3, q_4 \to q_4\}$,
- *e* $L(q_1) = \{\}, L(q_2) = \{b\}, L(q_3) = \{a\}, L(q_4) = \{a, b\}$).

a) Representa (T, q_3) por um autómato não determinístico de Büchi $\mathcal{A}_{(T,q_3)}$.

b) Mostra que existe uma palavra $\omega \in L(\mathcal{A}_{(\mathcal{M},q_3)}) \cap L(A_{\neg(aUb)})$ (ver alínea a) do exercício anterior).

c) O que podes concluir acerca de $T, q_3 \models aUb$?

$\diamond$

Exercício:

Considera o modelo $T = (S, \to, L)$ com

- $S = \{q_1, q_2, q_3, q_4\}$,
- $\to = \{q_1 \to q_2, q_2 \to q_2, q_3 \to q_1, q_3 \to q_2, q_3 \to q_4, q_4 \to q_3\}$,
- e $L(q_1) = \{\}, L(q_2) = \{b\}, L(q_3) = \{a\}, L(q_4) = \{a, b\}$).

Representa (T, q_1) por um autómato não determinístico de Büchi.

Algoritmo de *Model Checking* para o LTL

O problema de verificação de que um modelo satisfaz uma fórmula φ, $M, s_o \models \varphi$ reduz-se a saber se

$$L_\omega(A_M) \subseteq L_\omega(A_\varphi)$$

Ou equivalentemente,

$$L_\omega(A_M) \cap L_\omega(\overline{A_\varphi}) = \emptyset$$

Notar que $L_\omega(\overline{A_\varphi}) = L_\omega(A_{\neg\varphi})$.

O autómato para a intersecção tem $|S|.2^{\mathcal{O}(|\varphi|)}$ estados.

Logo o model checking pode ser feito em tempo $O(|S|.2^{\mathcal{O}(|\varphi|)})$. Como a especificação é em geral pequena, este algoritmo é razoavelmente eficiente...

Referências

[Var94] M. Vardi. An automata-theoretic approach to linear temporal logic. In *Banff'94*, 1994.

[Var06] Moshe Vardi. Automata-theoretic techniques for temporal reasoning. In Patrick Blackburn, Johan van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*. Elsevier, 2006.

[VW07] M. Vardi and T. Wilke. Automata: From logics to algorithms. In *WAL 07*, pages 645–753, 2007.