

# Automatic System Verification

## Exercices

Cominato Enrico 137396  
Department of Computer Science  
Univeristy of Udine

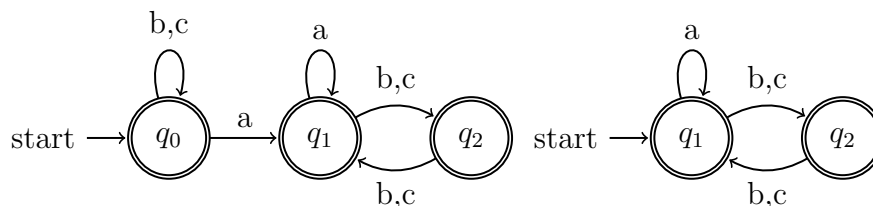
November 20, 2020

### 1 Exercices on the automata's notes

#### Esercizio 2.3

Sia  $\mathcal{A}$  l'automa dell'Esempio 2.2. Si consideri l'automa  $\mathcal{A}'$  ottenuto da  $\mathcal{A}$  rimuovendo lo stato  $q_0$ , e le transizioni in esso entranti e da esso uscenti, e facendo diventare  $q_1$  il nuovo stato iniziale. Si stabilisca se  $\mathcal{A}$  e  $\mathcal{A}'$  riconoscono o meno lo stesso linguaggio

Riporto di seguito i due grafi.



I due linguaggi non sono uguali. Per esempio la  $\omega$  – parola *babcabca...* appartiene al primo dei due automi, ma non al secondo (da  $q_1$  andiamo in  $q_2$  ma da lì possiamo leggere solo una *b* oppure una *c*)

#### Esercizio 2.4

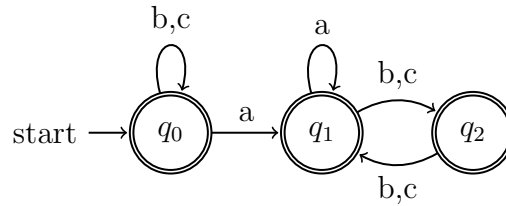
Si costruisca l'automa  $\mathcal{A}'$  che riconosce la variante finita (linguaggio di parole finite) dell'Esempio 2.2

Il linguaggio richiesto è il seguente:

*L'insieme delle parole finite su  $A = \{a, b, c\}$  tali che tra ogni coppia di occorrenze consecutive di  $a$  esiste un numero pari di occorrenze di simboli diversi da  $a$ .*

Osservazione: una parola con una sola occorrenza di  $a$  deve essere sempre accettata.

L'automa risultante quindi è lo stesso dell'esempio 2.2, solo che le run su questo automa sono finite.



## Esercizio 2.5

Sia  $W$  il linguaggio riconosciuto dall'automa  $\mathcal{A}'$  dell'Esercizio 2.4. Si caratterizzi il linguaggio  $\vec{W}$ .

Riprendo la definizione di  $\vec{W}$ :

$$\vec{W} = \{\alpha \in A^\omega \text{ t.c. } \exists^\omega n \alpha(0, n) \in W\}$$

Sono quindi tutte quelle  $\omega$ -parole di cui ogni prefisso finito appartiene a  $W$ . Analizziamo per casi:

- la parola non ha neppure una  $a$ : in questo caso ogni suo prefisso appartiene a  $W$  perchè le parole  $(b|c), (b|c)^2, (b|c)^3, \dots, (b|c)^n, \dots \in W$
- la parola ha una sola occorrenza di  $a$ : anche in questo caso ogni suo prefisso appartiene a  $W$  perchè:
  - come visto nel punto precedente, fino a che non si incontra la lettera  $a$  le parole appartengono a  $W$
  - dopo l'occorrenza di  $a$ , ancora tutti i prefissi appartengono a  $W$ , dato che tutte le parole finite con una sola occorrenza di  $a$  appartengono a  $W$

- infine tutti gli altri casi sono parole con un più di una occorrenza di  $a$ : dove abbiamo che ogni prefisso, o ricade in uno dei precedenti casi, oppure è una parola che tra ogni coppia di occorrenze consecutive di  $a$  esiste un numero pari di occorrenze di simboli diversi da  $a$  e che quindi appartiene a  $W$ .

In questo caso abbiamo che  $W^\omega = \overrightarrow{W}$ .

## Esercizio 2.7

### Teorema 2.6

1. Se  $V \subseteq A^*$  è regolare, allora  $V^\omega$  è  $\omega$ -regolare
2. Se  $V \subseteq A^*$  è regolare e  $L \subseteq A^\omega$  è  $\omega$ -regolare, allora  $V \cdot L$  è  $\omega$ -regolare
3. Se  $L_1, L_2 \subseteq A^*$  sono  $\omega$ -regolari, allora  $L_1 \cup L_2$  e  $L_1 \cap L_2$  sono  $\omega$ -regolari

*Dimostrare le proprietà (2) e (3) del teorema*

Dimostro (2). Siano  $\mathcal{A}, \mathcal{B}$  automi tali che  $\mathcal{A}$  accetta  $V$  e  $\mathcal{B}$  accetta  $L$ , allora possiamo costruire  $\mathcal{C}$  unendo tutti gli stati finali di  $\mathcal{A}$  con lo stato iniziale di  $\mathcal{B}$ . Otteniamo così un'automa che legge il linguaggio  $V \cdot L$ , quindi  $V \cdot L$  è  $\omega$ -regolare.

Dimostro (3). Per quanto riguarda l'unione, prendiamo  $\mathcal{A}, \mathcal{B}$  automi tali che  $\mathcal{A}$  accetta  $L_1$  e  $\mathcal{B}$  accetta  $L_2$ . Costruiamo  $\mathcal{C}$  unendo gli stati iniziali di  $\mathcal{A}$  e  $\mathcal{B}$ . Abbiamo quindi che  $\mathcal{C}$  accetta tutte le parole di  $L_1 \cup L_2$ . Sappiamo inoltre che i linguaggi  $\omega$ -regolari sono chiusi per complementazione, quindi possiamo riscrivere  $L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$  ed ottenere la chiusura per intersezione.

## Esercizio 2.13

*Fornire un esempio di parola non definitivamente periodica*

Un esempio è:

$$ababbab^3ab^4ab^5\ldots$$

## Esercizio 2.16

*Dimostrare che una congruenza è una relazione di equivalenza invariante destra*

Invariante a destra significa che  $\forall x, y, z \in A$ , se  $x \sim y$  allora  $xz \sim yz$ . Questo è sempre vero perchè, concatenando la stessa parola ad  $x, y$  finiremo in un'unica classe di equivalenza.

## Esercizio 2.19

*Dato un automa di Büchi  $\mathcal{A} = (\mathcal{Q}, A, \Delta, q_0, F)$ , dimostrare che, per ogni  $s, s' \in \mathcal{Q}$ ,  $W_{ss'}^F$  è regolare*

Un linguaggio è regolare se esiste un'automa in grado di accettarlo. Per poterlo accettare deve avere almeno uno stato finale. Quindi se eliminiamo dall'automa  $\mathcal{A}$ , tutti gli stati e le relazioni non interessate dai possibili cammini tra  $s$  e  $s'$  otteniamo un automa in grado di leggere solo  $W_{ss'}^F$ , e questo fa di lui un linguaggio regolare.

## Esercizio 2.23

*Dimostrare che la relazione  $\approx_A$  è una congruenza di indice finito*

Perchè  $\approx_A$  sia una congruenza deve valere che  $\forall u, u', v, v' \in A^*$  se  $u \approx_A v$  e  $u' \approx_A v'$  allora  $uu' \approx_A vv'$ . Questo è vero perchè avendo  $u \approx_A v$  e  $u' \approx_A v'$ , allora  $\exists t$  tale che:

- $s \rightarrow_u t \Leftrightarrow s \rightarrow_v t$
- $s \xrightarrow{F}_u t \Leftrightarrow s \xrightarrow{F}_v t$
- $t \rightarrow_{u'} s' \Leftrightarrow t \rightarrow_{v'} s'$
- $t \xrightarrow{F}_{u'} s' \Leftrightarrow t \xrightarrow{F}_{v'} s'$

Quindi abbiamo che  $\forall s, s' \in Q$ :

- $s \rightarrow_{uu'} s' \Leftrightarrow s \rightarrow_{vv'} s'$
- $s \xrightarrow{F}_{uu'} s' \Leftrightarrow s \xrightarrow{F}_{vv'} s'$

e quindi  $uu' \approx_A vv'$ . Il resto della dimostrazione è già stata svolta negli appunti.

## Esercizio 2.25

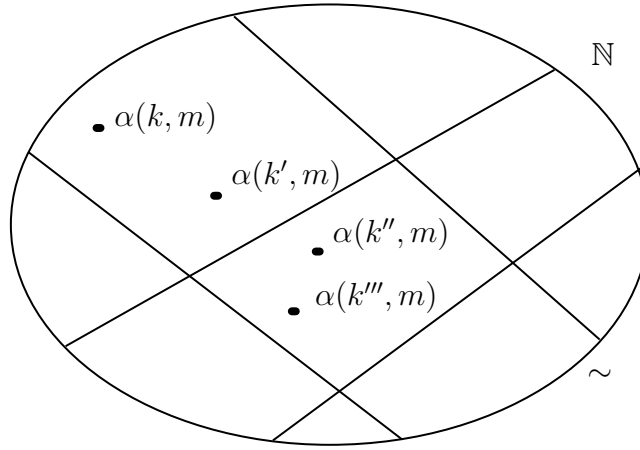
*Si dimostri che la relazione  $\cong_\alpha$  è una relazione di equivalenza sui naturali di indice finito*

Riprendo la definizione di  $\cong_\alpha$ . Sia  $\sim$  una congruenza su  $A^*$  di indice finito. Sia  $\alpha \in A^\omega$  e siano  $k, k'$  posizioni. Diciamo che  $k \cong_\alpha^m k'$  ( $k, k'$  si riuniscono in  $m > k, k'$ ) se  $\alpha(k, m) \sim \alpha(k', m)$ . Diciamo che  $k \cong_\alpha k'$  se esiste  $m$  per cui  $k \cong_\alpha^m k'$ .

Dimostro che è una relazione di equivalenza:

- **Riflessività:**  $\forall k \in \mathbb{N}$  è sempre vero che  $k \cong_\alpha k$  perchè  $k \cong_\alpha k \Leftrightarrow \exists m \text{ t.c. } \alpha(k, m) \sim \alpha(k, m)$  e questo è vero  $\forall k$  perchè  $\sim$  è una relazione di equivalenza
- **Simmetria:**  $\forall k, k' \in \mathbb{N}$  è sempre vero che  $k \cong_\alpha k' \Rightarrow k' \cong_\alpha k$  perchè  $k \cong_\alpha k' \Leftrightarrow \exists m \text{ t.c. } \alpha(k, m) \sim \alpha(k', m)$ . Dato che  $\sim$  è una relazione di equivalenza allora vale che  $\exists m \text{ t.c. } \alpha(k', m) \sim \alpha(k, m)$ , il che significa che  $k' \cong_\alpha k$ .
- **Transitività:**  $\forall i, j, k \in \mathbb{N}$  è sempre vero che  $i \cong_\alpha j$  e  $j \cong_\alpha k \Rightarrow i \cong_\alpha k$ , perchè  $i \cong_\alpha j \Leftrightarrow \exists m \text{ t.c. } \alpha(i, m) \sim \alpha(j, m)$  e  $j \cong_\alpha k \Leftrightarrow \exists n \text{ t.c. } \alpha(j, n) \sim \alpha(k, n)$ . Senza perdere di generalità pongo  $n > m$ , quindi abbiamo che  $\exists m \text{ t.c. } \alpha(i, m) \sim \alpha(j, m)$  e  $\alpha(j, m) \sim \alpha(k, m)$ . Dato che  $\sim$  è una relazione di equivalenza allora vale che  $\exists m$  tale che  $\alpha(i, m) \sim \alpha(j, m)$  e  $\alpha(j, m) \sim \alpha(k, m) \Rightarrow \alpha(i, m) \sim \alpha(k, m)$ , il che significa che  $i \cong_\alpha k$ .

Dimostro che  $\cong_\alpha$  ha indice finito. Dato che  $\sim$  ha indice finito, per un  $m$  fisso, ci troviamo in una situazione del genere:



Dove il numero di classi di equivalenza è limitato dal numero di classi di equivalenza di  $\sim$ , e sappiamo che  $\sim$  ha un numero finito di classi di equivalenza, quindi anche  $\cong_\alpha$  avrà un numero finito di classi di equivalenza.

### Esercizio 2.44

*Dimostrare la chiusura della classe dei linguaggi riconosciuti dagli automi di Büchi deterministici rispetto alle operazioni di unione e intersezione*

Siano  $\mathcal{A} = (\mathcal{Q}_A, A, \Delta_A, q_{0A}, F_A)$  e  $\mathcal{B} = (\mathcal{Q}_B, A, \Delta_B, q_{0B}, F_B)$

**Unione:** Se assumiamo che  $\mathcal{Q}_A \cap \mathcal{Q}_B = \emptyset$  allora possiamo costruire l'automa unione  $\mathcal{C}$  come segue:

- $\mathcal{Q}_C = \mathcal{Q}_A \cup \mathcal{Q}_B \cup \{q_{0C}\}$
- $A$  rimane invariato
- $\Delta_C = \Delta_A \cup \Delta_B$
- $q_{0C}$  come nuovo stato iniziale, con le stesse relazioni di  $q_{0A}$  e  $q_{0B}$ , finale nel caso che almeno uno tra  $q_{0A}$  e  $q_{0B}$  sia uno stato finale
- $F_C = F_A \cup F_B$

**Intersezione:** Costruiamo l'automa intersezione  $\mathcal{C}$ , partendo dal prodotto cartesiano degli stati:

- $\mathcal{Q}_C = \mathcal{Q}_A \times \mathcal{Q}_B \times \{1, 2\}$

- $A$  rimane invariato
- $\Delta_C = \Delta_1 \cup \Delta_2$  dove
  - $\Delta_1 = \{((q_A, q_B, 1), a, (q'_A, q'_B, i)) \mid (q_A, a, q'_A) \in \Delta_A \text{ e } (q_B, a, q'_B) \in \Delta_B \text{ e se } q_A \in F_A \text{ allora } i = 2 \text{ altrimenti } i = 1\}$
  - $\Delta_2 = \{((q_A, q_B, 2), a, (q'_A, q'_B, i)) \mid (q_A, a, q'_A) \in \Delta_A \text{ e } (q_B, a, q'_B) \in \Delta_B \text{ e se } q_B \in F_B \text{ allora } i = 1 \text{ altrimenti } i = 2\}$
- $q_{0C} = (q_{0A}, q_{0B}, 1)$
- $F_C = \{(q_a, q_b, 2) \mid q_B \in F_B\}$

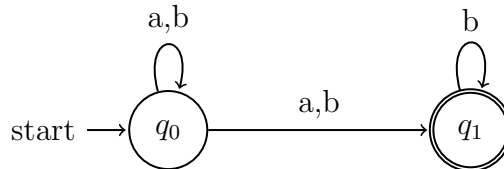
Per costruzione,  $r_C = (q_A^0, q_B^0, i^0), (q_A^1, q_B^1, i^1), \dots$  è un'esecuzione su  $\mathcal{C}$  per la parola  $w$  se:

- $r_A = q_A^0, q_A^1, \dots$  è un'esecuzione su  $\mathcal{A}$  per  $w$
- $r_B = q_B^0, q_B^1, \dots$  è un'esecuzione su  $\mathcal{B}$  per  $w$

$r_A$  e  $r_B$  sono accettate se  $r_C$  è la concatenazione di una serie infinita di segmenti finiti di stati 1 (stati con terza componente 1) e stati 2 (stati con terza componente 2) alternativamente. Questa sequenza esiste se  $r_C$  è accettato da  $\mathcal{A}$

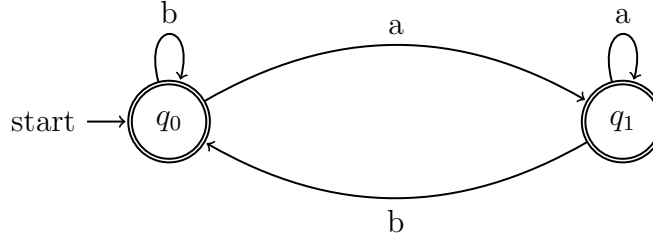
## Esercizio 2.46

Sia  $A = \{a, b\}$  e  $L = \{\alpha \in A^\omega. \exists^{<\omega} n \alpha(n) = a\}$ . Si costruisca un automa di Büchi non deterministico che riconosca il linguaggio  $L$



## Esercizio 2.48

Sia  $A = \{a, b\}$  e  $L = \overrightarrow{\{b^*a^*\}}$ . Si costruisca un automa di Büchi deterministico che riconosca il linguaggio  $L$



### Esercizio 2.50

*Dimostrare che la classe degli  $\omega$ -linguaggi  $\omega$ -regolari coincide con la classe degli  $\omega$ -linguaggi riconosciuti dagli automi di Muller non deterministici*

Dato che un  $\omega$ -linguaggio per essere  $\omega$ -regolare deve essere accettato da un automa di Büchi, mi basta dimostrare l'equivalenza tra gli automi di Büchi non deterministici e quelli di Muller non deterministici.

Sia  $\mathcal{A} = (\mathcal{Q}, A, \Delta, q_0, F)$  un automa di Büchi, possiamo costruire un automa di Muller:  $\mathcal{M} = (\mathcal{Q}, A, \Delta, q_0, \mathcal{F})$  dove  $\mathcal{F} = \{X \mid X \in 2^{\mathcal{Q}} \wedge X \cap F \neq \emptyset\}$ . Si può osservare che una  $\omega$ -parola viene accettata da  $\mathcal{A}$  se e solo se passa infinite volte per uno stato finale. Quindi viene accettata anche da  $\mathcal{M}$  perchè, presa una sua computazione  $\sigma$ , abbiamo che  $In(\sigma) \cap F \neq \emptyset$  e  $In(\sigma) \in 2^{\mathcal{Q}} \Rightarrow In(\sigma) \in \mathcal{F}$  quindi la stessa  $\omega$ -parola viene accettata da  $\mathcal{F}$ . Lo stesso ragionamento lo possiamo fare al contrario. Se una  $\omega$ -parola viene accettata da  $\mathcal{M}$  allora  $In(\sigma) \cap F \neq \emptyset$  quindi esiste una computazione che passa infinite volte per uno stato finale, quindi la stessa  $\omega$ -parola viene accettata da  $\mathcal{A}$ .

### Esercizio 2.57

*Dimostrare che l'insieme  $W_V \subseteq A^*$  dei  $V$ -testimoni, con  $V$  classe di congruenza  $\approx_{\mathcal{A}}$  è regolare*



## 2 Exercices of chapter 0 of Temporal Verification of Reactive Systems

### Problem 0.1

```

out x: integer where x = 0

l0 : [ [l1 : while x ≥ 0 do l2 : x := x + 1]
      or
      [l3 : await x > 0] ]
l4 :

Program S8 (strange behavior).

```

**a)** Identify the locations of this program as equivalence classes of labels. List the post-location of each of the statements.

There are three classes:

$$l_0 = \{l_0, l_1, l_3\}$$

$$l_2 = \{l_2\}$$

$$l_4 = \{l_4\}$$

While the post-locations are:

$$post(l_0) = post(l_1) = post(l_3) = [l_4]$$

$$post(l_2) = [l_0]$$

**b)** Show that this program has a terminating computation.

The program can terminate because the post-location of the body of the while is the selection statement. So the await condition can be satisfied and this terminate the program.

**c)** Define transitions and transition relations for this version of a WHILE statement. Show that the version of program SB in which the while statement has been replaced by this WHILE statement has no terminating computation.

Recall of the new WHILE statement:

$$l_1 : [WHILE c DO [l_2 : S; \hat{l}_2]]; l_3 : \text{ where } \hat{l}_2 \approx l_1$$

We can define its transitions  $\tau_{l_1}, \tau_{\hat{l}_2}$  and transition relations  $\rho_{l_1}, \rho_{\hat{l}_2}$  as follow:

$\rho_{l_1} : \rho_{l_1}^T \vee \rho_{l_1}^F$  where

$$\rho_{l_1}^T : \text{move}(l_1, l_2) \wedge c \wedge \text{pres}(Y)$$

$$\rho_{l_1}^F : \text{move}(l_1, l_3) \wedge \neg c \wedge \text{pres}(Y)$$

$\rho_{\hat{l}_2} : \rho_{\hat{l}_2}^T \vee \rho_{\hat{l}_2}^F$  where

$$\rho_{\hat{l}_2}^T : \text{move}(\hat{l}_2, l_2) \wedge c \wedge \text{pres}(Y)$$

$$\rho_{\hat{l}_2}^F : \text{move}(\hat{l}_2, l_3) \wedge \neg c \wedge \text{pres}(Y)$$

As we can see, now the program cannot terminate anymore, because once in the WHILE loop, the transition  $\rho_{\hat{l}_2}^F$  will be never satisfied.

## Problem 0.2

**out y: integer where y = 0**  
**local x: boolean where x = T**

$$P_1 :: \left[ \begin{array}{l} l_0 : \text{while } x \text{ do} \\ l_1 : y := y + 1 \\ l_2 : \end{array} \right] \quad \parallel \quad P_2 :: \left[ \begin{array}{l} m_0 : x := F \\ m_1 : \end{array} \right]$$

Program ANY-NAT (computing any natural number).

**a)** Consider program ANY-NAT. Argue (informally) that this program always terminates. Also show that, for every natural number  $n \geq 0$ , there exists a computation of ANY-NAT such that  $y = n$  on termination.

The program terminates whenever  $m_0$  is executed. The transition  $\tau_{m_0}$  is always enabled and because of *justice* requirement we know that, soon or later, it will be taken. Once executed  $\rho_{l_0}^F$  is satisfied and control goes to  $\pi = \{l_2, m_1\}$ . Furthermore, the justice requirement allow only a finite (but unbounded) number of transition  $\tau_{l_0}$ . So,  $\forall n \in \mathbb{N}$  there exists a run where  $\tau_{l_0}$  is taken  $n$  times before  $\tau_{m_0}$ , leading to have  $y = n$  on termination.

**b)** Construct a program with a single process that exhibits a similar behavior; that is, all of its computations terminate and, for each natural number  $n$ , there exists a computation producing  $n$ .

<b>out y: integer where y = 0</b> <b>local x: boolean where x = T</b>  $l_0 : \left[ \text{while } x \text{ do } l_1 : \left[ \begin{array}{c} [l_2 : y := y + 1] \\ \text{or} \\ [l_3 : x := F] \end{array} \right] \right]$ $l_4 :$
---

This Program exhibits the same behavior of the previous one:

- All of its computation terminates, because  $\tau_{l_3}$  is always enabled and thank to the justice requirement we know that soon or later it will be taken and change the value of  $x$  ending the cycle.
- As the previous program, the number of iteration before the execution of  $\tau_{l_3}$  is finite but unbounded, so  $y$  could have any value in  $\mathcal{N}$  at the end of the computation.

*c) Prove that no single-process program with only just transitions can have the same behavior as ANY-NAT. This kind of program need:*

- an incremental statement
- a termination statement

### Problem 0.3

*Consider a program P of the form*

$$P :: \left[ \text{declaration}; \left[ [l_1 : S_1; \hat{l}_1 :] \parallel \dots \parallel [l_k : S_k; \hat{l}_k :] \right] \right]$$

*a) Consider the two statements*

$$S_1 :: [x := 1; x := 2]$$

$$S_2 :: [x := 1; x := x + 1]$$

and the context

$$P[S] :: [\textbf{out } x : \textbf{integer where } x = 0; S].$$

Show that  $P[S_1]$  and  $P[S_2]$  are termination equivalent

**b)** Consider the preceding two statements and the context

$$Q[S] :: [\textbf{out } x : \textbf{integer where } x = 0; [S \parallel x := 0]].$$

Show that  $Q[S_1]$  and  $Q[S_2]$  are not termination equivalent. We may conclude that  $S_1$  and  $S_2$  are not congruent.

**c)** Show the following congruence:

1.  $[S_1 \textbf{or} S_2] \approx [S_2 \textbf{or} S_1]$
2.  $[S_1 \parallel S_2] \approx [S_2 \parallel S_1]$
3.  $S \approx [S; \textbf{skip}]$
4.  $\textbf{await } c \approx \textbf{while } \neg c \textbf{ do skip}$

**d)** Are the two statements:

$$\textbf{await } x \textbf{ and skip}; m : \textbf{await } x$$

congruent? Prove your answer.

**e)** Let  $y$  be a boolean variable. Which of the three following statements are congruent?

$$y := T$$

$$\textbf{if } y \textbf{ then } y := T \textbf{ else } y := T$$

$$[[\textbf{await } y; y := T] \textbf{ or } [\textbf{await } \neg y; y := T]]$$

### 3 Additional exercises

#### Esercizio1

*Dato un linguaggio  $L \subseteq A^*$ , dimostrare se che  $L$  è un linguaggio star-free, allora  $L$  è definibile nel frammento al prim'ordine di  $S1S_A$ , con la relazione di ordinamento  $<$  e i predicati unari  $Q_a$ , con  $a \in A$ .*

#### Esercizio2

*Dimostrare che l'insieme dei linguaggi riconosciuti da automi di Büchi su alberi infiniti con insieme degli stati finali singoletto è strettamente contenuto nell'insieme dei linguaggi riconosciuti da automi di Büchi su alberi infiniti*

#### Esercizio3

*Sia  $A = \{a, b\}$  e  $T_1 = \{t \in T_A^\omega : \text{tutti i cammini di } t \text{ contengono un numero finito di occorrenze di } a\}$ .  $T_1$  contiene l'insieme di tutti gli alberi  $t_i$ , con  $i \geq 0$ , tali che  $t_i$  ha un'occorrenza di  $a$  nelle posizioni  $\epsilon, 1^{m_1}0, \dots, 1^{m_1}01^{m_2}0 \dots 1^{m_i}0$ , con  $m_1, m_2, \dots, m_i > 0$ . Immaginiamo che esista un automa di Büchi  $\mathcal{A} = (\mathcal{Q}, A, \Delta, q_0, F)$  con  $n + 1$  stati, con  $n \geq 1$ , incluso lo stato iniziale  $q_0$  che occorre solo in posizione  $\epsilon$  tale che  $L(\mathcal{A}) = T_1$  e sia  $r$  un run di successo di  $\mathcal{A}$  su  $t_n$ . Mostrare che deve esistere un cammino in  $t_n$  contenente 3 nodi  $u, v$  e  $w$ , con  $u < v < w$ , tali che  $r(u) = r(w) = s \in F$  e  $t_n(v) = a$ .*

#### Esercizio4

*Siano  $C = \{c_1, \dots, c_m\}$  e  $\bar{c} = (c_1, \dots, c_m)$ . Sia dato  $T \subseteq T_A^\omega$  tale che  $T = T_0 \cdot \bar{c}(T_1, \dots, T_m)^\omega$*

#### Esercizio5

*Dimostrare la (correttezza e completezza della) caratterizzazione di uno degli operatori di CTL (diverso da AF) quale minimo punto fisso di un'opportuna trasformazione di predicato.*

## Esercizio6

*Dimostrare la (correttezza e completezza della) caratterizzazione di uno degli operatori di CTL (diverso da EG) quale massimo punto fisso di un'opportuna trasformazione di predicato.*