

Automati di Büchi Etichettati & Model Checking LTL

Giuseppe Oliva

Presentazione per il corso di
Metodi e Strumenti per l'Analisi e la Verifica
Prof.ssa Stefania Gnesi

29 Marzo 2006

Problema del Model Checking

Dato un modello \mathcal{M} , uno stato s ed una proprietà espressa con una formula LTL ϕ , determinare se

$$\mathcal{M}, s \models \phi$$

Soddisfacibilità formule LTL
decidibile



Model Checking per LTL
decidibile

Il Model Checking LTL è basato su una variante degli automi a stati finiti, chiamati **Automi di Büchi**

Problema del Model Checking

Dato un modello \mathcal{M} , uno stato s ed una proprietà espressa con una formula LTL ϕ , determinare se

$$\mathcal{M}, s \models \phi$$

Soddisfacibilità formule LTL
decidibile



Model Checking per LTL
decidibile

Il Model Checking LTL è basato su una variante degli automi a stati finiti, chiamati **Automi di Büchi**

Problema del Model Checking

Dato un modello \mathcal{M} , uno stato s ed una proprietà espressa con una formula LTL ϕ , determinare se

$$\mathcal{M}, s \models \phi$$

Soddisfacibilità formule LTL
decidibile



Model Checking per LTL
decidibile

Il Model Checking LTL è basato su una variante degli automi a stati finiti, chiamati **Automi di Büchi**

Definizione di Automa a stati finiti

Un *Automa a stati finiti etichettato* (LFSA) è una tupla $(\Sigma, S, S^0, \rho, F, L)$

- Σ è un alfabeto di simboli
- S è un insieme finito di stati
- $S^0 \subseteq S$ è l'insieme di stati iniziali
- $\rho : S \rightarrow 2^S$ è la funzione di transizione di stato
- $F \subseteq S$ è l'insieme di stati finali
- $L : S \rightarrow \Sigma$ è la funzione di **etichettatura degli stati**

Un LFSA è deterministico sse

- 1 $|\{s \in S^0 \mid L(s) = a\}| \leq 1 \quad \forall a \in \Sigma$
- 2 $|\{s' \in \rho(s) \mid L(s') = a\}| \leq 1 \quad \forall a \in \Sigma, \forall s \in S$

Definizione di Automa a stati finiti

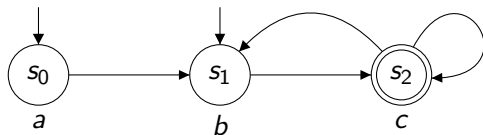
Un *Automa a stati finiti etichettato* (LFSA) è una tupla $(\Sigma, S, S^0, \rho, F, L)$

- Σ è un alfabeto di simboli
- S è un insieme finito di stati
- $S^0 \subseteq S$ è l'insieme di stati iniziali
- $\rho : S \rightarrow 2^S$ è la funzione di transizione di stato
- $F \subseteq S$ è l'insieme di stati finali
- $L : S \rightarrow \Sigma$ è la funzione di **etichettatura degli stati**

Un LFSA è deterministico sse

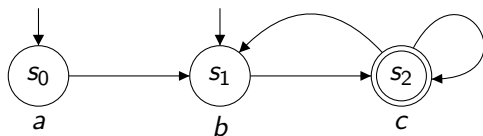
- 1 $|\{s \in S^0 \mid L(s) = a\}| \leq 1 \quad \forall a \in \Sigma$
- 2 $|\{s' \in \rho(s) \mid L(s') = a\}| \leq 1 \quad \forall a \in \Sigma, \forall s \in S$

Esempi determinismo/nondeterminismo

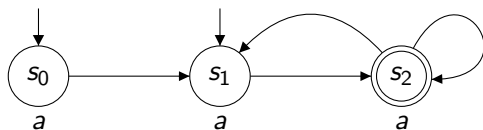
Deterministico

Esempi determinismo/nondeterminismo

Deterministico

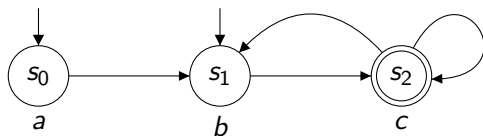


Nondeterministico

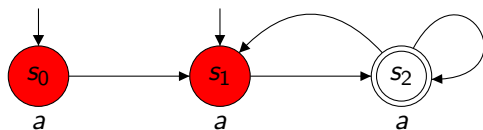


Esempi determinismo/nondeterminismo

Deterministico

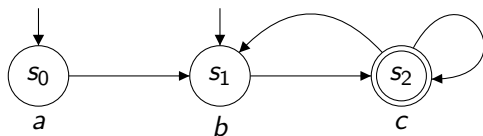


Nondeterministico

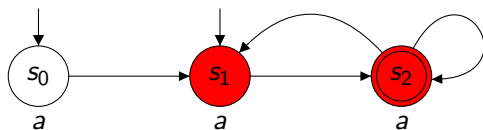


Esempi determinismo/nondeterminismo

Deterministico



Nondeterministico



Linguaggio accettato

- Un **run** per un LFSA è una sequenza finita $\sigma = s_0 s_1 \dots s_n$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall 0 \leq i < n$$

- Un run è detto **accepting** se $s_n \in F$

- Una parola $w = a_0 a_1 \dots a_n$ è **accettata** sse esiste un accepting run $\sigma = s_0 s_1 \dots s_n$ tale che $L(s_i) = a_i \quad \forall 0 \leq i \leq n \quad (a_i \in \Sigma)$

- Se si indica con Σ^* l'insieme di tutte le possibili **parole finite** su Σ , il **linguaggio accettato** dall'LFSA A è

$$\mathcal{L}(A) = \{ w \in \Sigma^* \mid w \text{ è accettata da } A \}$$

Linguaggio accettato

- Un **run** per un LFSA è una sequenza finita $\sigma = s_0 s_1 \dots s_n$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall 0 \leq i < n$$

- Un run è detto **accepting** se $s_n \in F$

- Una parola $w = a_0 a_1 \dots a_n$ è **accettata** sse esiste un accepting run $\sigma = s_0 s_1 \dots s_n$ tale che $L(s_i) = a_i \quad \forall 0 \leq i \leq n \quad (a_i \in \Sigma)$

- Se si indica con Σ^* l'insieme di tutte le possibili **parole finite** su Σ , il **linguaggio accettato** dall'LFSA A è

$$\mathcal{L}(A) = \{ w \in \Sigma^* \mid w \text{ è accettata da } A \}$$

Linguaggio accettato

- Un **run** per un LFSA è una sequenza finita $\sigma = s_0 s_1 \dots s_n$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall 0 \leq i < n$$

- Un run è detto **accepting** se $s_n \in F$

- Una parola $w = a_0 a_1 \dots a_n$ è **accettata** sse esiste un accepting run

$$\sigma = s_0 s_1 \dots s_n \quad \text{tale che} \quad L(s_i) = a_i \quad \forall 0 \leq i \leq n \quad (a_i \in \Sigma)$$

- Se si indica con Σ^* l'insieme di tutte le possibili **parole finite** su Σ , il **linguaggio accettato** dall'LFSA A è

$$\mathcal{L}(A) = \{ w \in \Sigma^* \mid w \text{ è accettata da } A \}$$

Linguaggio accettato

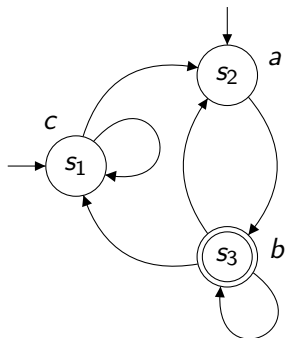
- Un **run** per un LFSA è una sequenza finita $\sigma = s_0 s_1 \dots s_n$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall 0 \leq i < n$$

- Un run è detto **accepting** se $s_n \in F$
- Una parola $w = a_0 a_1 \dots a_n$ è **accettata** sse esiste un accepting run $\sigma = s_0 s_1 \dots s_n$ tale che $L(s_i) = a_i \quad \forall 0 \leq i \leq n \quad (a_i \in \Sigma)$
- Se si indica con Σ^* l'insieme di tutte le possibili **parole finite** su Σ , il **linguaggio accettato** dall'LFSA A è

$$\mathcal{L}(A) = \{ w \in \Sigma^* \mid w \text{ è accettata da } A \}$$

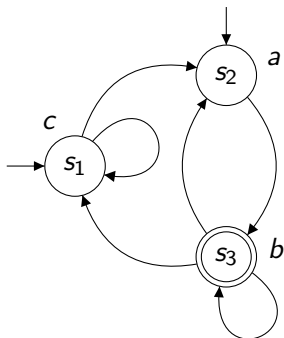
Esempio linguaggio accettato



Linguaggio accettato

$$(c^* a b^+)^+$$

Esempio linguaggio accettato



Linguaggio accettato

$$(c^* a b^+)^+$$

Definizione di Automa di Büchi

Un *Automa di Büchi etichettato* (LBA) è un LFSA che accetta parole di lunghezza **infinita** invece che di lunghezza finita

- Un **run** per un LBA è una sequenza infinita $\sigma = s_0s_1 \dots$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall i \geq 0$$
- Sia $\text{lim}(\sigma)$ l'insieme di stati che occorre in σ infinitamente spesso. Allora σ è un **accepting** run sse $\text{lim}(\sigma) \cap F \neq \emptyset$
- Poiché il numero di stati dell'automa è finito, mentre la sequenza σ ha lunghezza infinita, sicuramente $\text{lim}(\sigma) \neq \emptyset$

Definizione di Automa di Büchi

Un *Automa di Büchi etichettato* (LBA) è un LFSA che accetta parole di lunghezza **infinita** invece che di lunghezza finita

- Un **run** per un LBA è una sequenza infinita $\sigma = s_0 s_1 \dots$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall i \geq 0$$

- Sia $\text{lim}(\sigma)$ l'insieme di stati che occorre in σ infinitamente spesso. Allora σ è un **accepting** run sse $\text{lim}(\sigma) \cap F \neq \emptyset$
- Poiché il numero di stati dell'automa è finito, mentre la sequenza σ ha lunghezza infinita, sicuramente $\text{lim}(\sigma) \neq \emptyset$

Definizione di Automa di Büchi

Un *Automa di Büchi etichettato* (LBA) è un LFSA che accetta parole di lunghezza **infinita** invece che di lunghezza finita

- Un **run** per un LBA è una sequenza infinita $\sigma = s_0 s_1 \dots$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall i \geq 0$$

- Sia $\text{lim}(\sigma)$ l'insieme di stati che occorre in σ infinitamente spesso. Allora σ è un **accepting** run sse $\text{lim}(\sigma) \cap F \neq \emptyset$
- Poiché il numero di stati dell'automa è finito, mentre la sequenza σ ha lunghezza infinita, sicuramente $\text{lim}(\sigma) \neq \emptyset$

Definizione di Automa di Büchi

Un *Automa di Büchi etichettato* (LBA) è un LFSA che accetta parole di lunghezza **infinita** invece che di lunghezza finita

- Un **run** per un LBA è una sequenza infinita $\sigma = s_0 s_1 \dots$ tale che

$$s_0 \in S^0 \quad \text{e} \quad s_i \rightarrow s_{i+1} \quad \forall i \geq 0$$

- Sia $\text{lim}(\sigma)$ l'insieme di stati che occorre in σ infinitamente spesso. Allora σ è un **accepting** run sse $\text{lim}(\sigma) \cap F \neq \emptyset$
- Poiché il numero di stati dell'automa è finito, mentre la sequenza σ ha lunghezza infinita, sicuramente $\text{lim}(\sigma) \neq \emptyset$

Linguaggio accettato

- Una parola $w = a_0a_1 \dots$ è **accettata** da un LBA sse esiste un accepting run

$$\sigma = s_0s_1 \dots \quad \text{tale che} \quad L(s_i) = a_i \quad \forall i \geq 0 \quad (a_i \in \Sigma)$$

- Se si indica con Σ^ω l'insieme di tutte le possibili **parole infinite** su Σ , il **linguaggio accettato** dall'LBA A è

$$\mathcal{L}_\omega(A) = \{ w \in \Sigma^\omega \mid w \text{ è accettata da } A \}$$

Linguaggio accettato

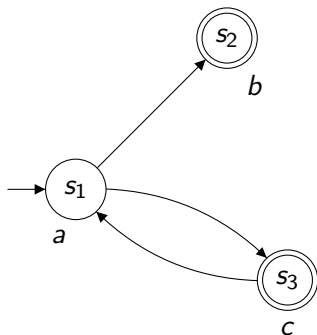
- Una parola $w = a_0a_1 \dots$ è **accettata** da un LBA sse esiste un accepting run

$$\sigma = s_0s_1 \dots \quad \text{tale che} \quad L(s_i) = a_i \quad \forall i \geq 0 \quad (a_i \in \Sigma)$$

- Se si indica con Σ^ω l'insieme di tutte le possibili **parole infinite** su Σ , il **linguaggio accettato** dall'LBA A è

$$\mathcal{L}_\omega(A) = \{ w \in \Sigma^\omega \mid w \text{ è accettata da } A \}$$

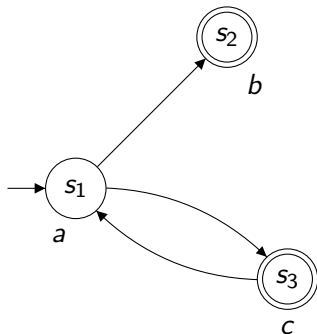
Esempio Automa di Büchi



Linguaggio accettato

$(ac)^\omega$

Esempio Automa di Büchi



Linguaggio accettato

$(ac)^\omega$

Equivalenza ed Espressività

Due automi A_1 e A_2 sono **equivalenti** se accettano lo stesso linguaggio

Valgono in generale le seguenti considerazioni:

- $\mathcal{L}(A_1) = \mathcal{L}(A_2) \not\Rightarrow \mathcal{L}_\omega(A_1) = \mathcal{L}_\omega(A_2)$
- $\mathcal{L}_\omega(A_1) = \mathcal{L}_\omega(A_2) \not\Rightarrow \mathcal{L}(A_1) = \mathcal{L}(A_2)$

... inoltre

- *A LFSA nondeterministico \Rightarrow esiste A' LFSA deterministico tale che*

$$\mathcal{L}(A) = \mathcal{L}(A')$$
- *A LBA nondeterministico $\not\Rightarrow$ esiste A' LBA deterministico tale che*

$$\mathcal{L}_\omega(A) = \mathcal{L}_\omega(A')$$

Equivalenza ed Espressività

Due automi A_1 e A_2 sono **equivalenti** se accettano lo stesso linguaggio

Valgono in generale le seguenti considerazioni:

- $\mathcal{L}(A_1) = \mathcal{L}(A_2) \not\Rightarrow \mathcal{L}_\omega(A_1) = \mathcal{L}_\omega(A_2)$
- $\mathcal{L}_\omega(A_1) = \mathcal{L}_\omega(A_2) \not\Rightarrow \mathcal{L}(A_1) = \mathcal{L}(A_2)$

... inoltre

- A LFSA *nondeterministico* \Rightarrow esiste A' LFSA *deterministico* tale che
$$\mathcal{L}(A) = \mathcal{L}(A')$$
- A LBA *nondeterministico* $\not\Rightarrow$ esiste A' LBA *deterministico* tale che
$$\mathcal{L}_\omega(A) = \mathcal{L}_\omega(A')$$

Idee alla base del Model Checking LTL (1)

- Modifichiamo la funzione di etichettatura degli stati in modo da considerare **insiemi di simboli**

$$L : S \rightarrow 2^\Sigma$$

- Una parola $w = a_0 a_1 \dots$ è **accettata** da un LBA sse esiste un accepting run

$$\sigma = s_0 s_1 \dots \quad \text{tale che} \quad a_i \in L(s_i) \quad \forall i \geq 0$$

- Consideriamo $\Sigma = 2^{AP}$, così gli stati saranno etichettati con **insiemi di insiemi** di proposizioni atomiche

Idee alla base del Model Checking LTL (1)

- Modifichiamo la funzione di etichettatura degli stati in modo da considerare **insiemi di simboli**

$$L : S \rightarrow 2^\Sigma$$

- Una parola $w = a_0 a_1 \dots$ è **accettata** da un LBA sse esiste un accepting run

$$\sigma = s_0 s_1 \dots \quad \text{tale che} \quad a_i \in L(s_i) \quad \forall i \geq 0$$

- Consideriamo $\Sigma = 2^{AP}$, così gli stati saranno etichettati con **insiemi di insiemi di proposizioni atomiche**

Idee alla base del Model Checking LTL (1)

- Modifichiamo la funzione di etichettatura degli stati in modo da considerare **insiemi di simboli**

$$L : S \rightarrow 2^\Sigma$$

- Una parola $w = a_0 a_1 \dots$ è **accettata** da un LBA sse esiste un accepting run

$$\sigma = s_0 s_1 \dots \quad \text{tale che} \quad a_i \in L(s_i) \quad \forall i \geq 0$$

- Consideriamo $\Sigma = 2^{AP}$, così gli stati saranno etichettati con **insiemi di insiemi** di proposizioni atomiche

Idee alla base del Model Checking LTL (2)

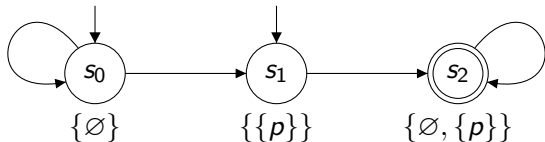
Si vuole associare ad ogni formula LTL ϕ un LBA il cui linguaggio accettato corrisponda alle sequenze di proposizioni atomiche che rendono valida ϕ

Corrisponde a

Fp

Idee alla base del Model Checking LTL (2)

Si vuole associare ad ogni formula LTL ϕ un LBA il cui linguaggio accettato corrisponda alle sequenze di proposizioni atomiche che rendono valida ϕ

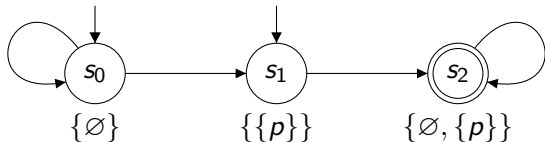


Corrisponde a

$F p$

Idee alla base del Model Checking LTL (2)

Si vuole associare ad ogni formula LTL ϕ un LBA il cui linguaggio accettato corrisponda alle sequenze di proposizioni atomiche che rendono valida ϕ



Corrisponde a

$F p$

Codifica delle formule proposizionali

Gli insiemi di insiemi di proposizioni atomiche **codificano** le formule proposizionali

- Se $AP_1, \dots, AP_n \subseteq AP$, allora ogni AP_i codifica la formula

$$\llbracket AP_i \rrbracket = (\forall p \in AP_i : p) \wedge (\forall p \in AP - AP_i : \neg p)$$

- L'insieme $\{AP_{k_1}, \dots, AP_{k_m}\}$, per $m \geq 1$ e $0 < k_j \leq n$ codifica

$$\llbracket AP_{k_1} \rrbracket \vee \dots \vee \llbracket AP_{k_m} \rrbracket$$

Gli insiemi di insiemi di proposizioni atomiche non possono essere **vuoti**

Codifica delle formule proposizionali

Gli insiemi di insiemi di proposizioni atomiche **codificano** le formule proposizionali

- Se $AP_1, \dots, AP_n \subseteq AP$, allora ogni AP_i codifica la formula

$$\llbracket AP_i \rrbracket = (\forall p \in AP_i : p) \wedge (\forall p \in AP - AP_i : \neg p)$$

- L'insieme $\{AP_{k_1}, \dots, AP_{k_m}\}$, per $m \geq 1$ e $0 < k_j \leq n$ codifica

$$\llbracket AP_{k_1} \rrbracket \vee \dots \vee \llbracket AP_{k_m} \rrbracket$$

Gli insiemi di insiemi di proposizioni atomiche non possono essere **vuoti**

Codifica delle formule proposizionali

Gli insiemi di insiemi di proposizioni atomiche **codificano** le formule proposizionali

- Se $AP_1, \dots, AP_n \subseteq AP$, allora ogni AP_i codifica la formula

$$\llbracket AP_i \rrbracket = (\forall p \in AP_i : p) \wedge (\forall p \in AP - AP_i : \neg p)$$

- L'insieme $\{AP_{k_1}, \dots, AP_{k_m}\}$, per $m \geq 1$ e $0 < k_j \leq n$ codifica

$$\llbracket AP_{k_1} \rrbracket \vee \dots \vee \llbracket AP_{k_m} \rrbracket$$

Gli insiemi di insiemi di proposizioni atomiche non possono essere **vuoti**

Codifica delle formule proposizionali

Gli insiemi di insiemi di proposizioni atomiche **codificano** le formule proposizionali

- Se $AP_1, \dots, AP_n \subseteq AP$, allora ogni AP_i codifica la formula

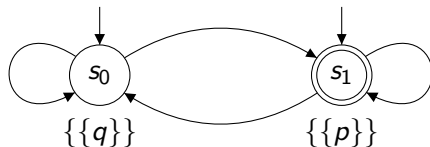
$$\llbracket AP_i \rrbracket = (\forall p \in AP_i : p) \wedge (\forall p \in AP - AP_i : \neg p)$$

- L'insieme $\{AP_{k_1}, \dots, AP_{k_m}\}$, per $m \geq 1$ e $0 < k_j \leq n$ codifica

$$\llbracket AP_{k_1} \rrbracket \vee \dots \vee \llbracket AP_{k_m} \rrbracket$$

Gli insiemi di insiemi di proposizioni atomiche non possono essere **vuoti**

Esempio codifica formule



$$AP = \{p, q\}$$

$$s_0: (q \wedge \neg p)$$

$$s_1: (p \wedge \neg q)$$

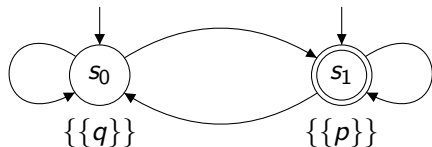
Formula LTL:

$$G[(q \wedge \neg p) U (p \wedge \neg q)]$$

Stati \rightarrow formule proposizionali
 Transizioni \rightarrow comportamento temporale

} \implies *Formule LTL*

Esempio codifica formule



$$AP = \{p, q\}$$

$$s_0: (q \wedge \neg p)$$

$$s_1: (p \wedge \neg q)$$

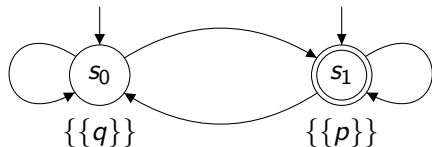
Formula LTL:

$$G[(q \wedge \neg p) U (p \wedge \neg q)]$$

Stati \rightarrow formule proposizionali
 Transizioni \rightarrow comportamento temporale

} \implies *Formule LTL*

Esempio codifica formule



$$AP = \{p, q\}$$

$$s_0: (q \wedge \neg p)$$

$$s_1: (p \wedge \neg q)$$

Formula LTL:

$$G[(q \wedge \neg p) U (p \wedge \neg q)]$$

Stati \rightarrow formule proposizionali
Transizioni \rightarrow comportamento temporale

$\left. \begin{array}{l} \text{Stati} \rightarrow \text{formule proposizionali} \\ \text{Transizioni} \rightarrow \text{comportamento temporale} \end{array} \right\} \Rightarrow \text{Formule LTL}$

Model Checking (primo approccio)

Il metodo più semplice è quello di verificare che tutti i comportamenti del sistema siano desiderabili:

- 1 Costruire l'LBA per $\phi \implies A_\phi$
- 2 Costruire l'LBA per il modello del sistema $\implies A_{sys}$
- 3 Verificare se $\mathcal{L}_\omega(A_{sys}) \subseteq \mathcal{L}_\omega(A_\phi)$

Ma decidere l'inclusione tra i linguaggi accettati è un **problema NP**

Model Checking (primo approccio)

Il metodo più semplice è quello di verificare che tutti i comportamenti del sistema siano desiderabili:

- 1 Costruire l'LBA per $\phi \implies A_\phi$
- 2 Costruire l'LBA per il modello del sistema $\implies A_{sys}$
- 3 Verificare se $\mathcal{L}_\omega(A_{sys}) \subseteq \mathcal{L}_\omega(A_\phi)$

Ma decidere l'inclusione tra i linguaggi accettati è un **problema NP**

Model Checking (secondo approccio)

Il problema di verificare l'inclusione tra linguaggi può essere aggirato poiché

$$\mathcal{L}_\omega(A_{sys}) \subseteq \mathcal{L}_\omega(A_\phi) \Leftrightarrow \mathcal{L}_\omega(A_{sys}) \cap \mathcal{L}_\omega(\overline{A_\phi}) = \emptyset$$

$\overline{A_\phi}$ è l'LBA **complementare** di A_ϕ e accetta il linguaggio $\Sigma^\omega \setminus \mathcal{L}_\omega(A_\phi)$

In generale la costruzione di $\overline{A_\phi}$ è **quadraticamente esponenziale**

$$A_\phi \text{ ha } n \text{ stati} \implies \overline{A_\phi} \text{ ha } c^{n^2} \text{ stati} \quad (c > 1)$$

Model Checking (secondo approccio)

Il problema di verificare l'inclusione tra linguaggi può essere aggirato poiché

$$\mathcal{L}_\omega(A_{sys}) \subseteq \mathcal{L}_\omega(A_\phi) \Leftrightarrow \mathcal{L}_\omega(A_{sys}) \cap \mathcal{L}_\omega(\overline{A_\phi}) = \emptyset$$

$\overline{A_\phi}$ è l'LBA **complementare** di A_ϕ e accetta il linguaggio $\Sigma^\omega \setminus \mathcal{L}_\omega(A_\phi)$

In generale la costruzione di $\overline{A_\phi}$ è quadraticamente esponenziale

$$A_\phi \text{ ha } n \text{ stati} \implies \overline{A_\phi} \text{ ha } c^{n^2} \text{ stati} \quad (c > 1)$$

Model Checking (terzo approccio)

Infine, osservando che $\mathcal{L}_\omega(\overline{A_\phi}) = \mathcal{L}_\omega(A_{\neg\phi})$, si arriva ad un metodo efficiente di model checking:

- 1 Costruire l'LBA per $\neg\phi \implies A_{\neg\phi}$
- 2 Costruire l'LBA per il modello del sistema $\implies A_{sys}$
- 3 Verificare se $\mathcal{L}_\omega(A_{sys}) \cap \mathcal{L}_\omega(A_{\neg\phi}) = \emptyset$

(i run in comune tra A_{sys} e $A_{\neg\phi}$ violano ϕ , quindi sono indesiderati)

Il terzo passo dell'algoritmo è decidibile in **tempo lineare**

Model Checking (terzo approccio)

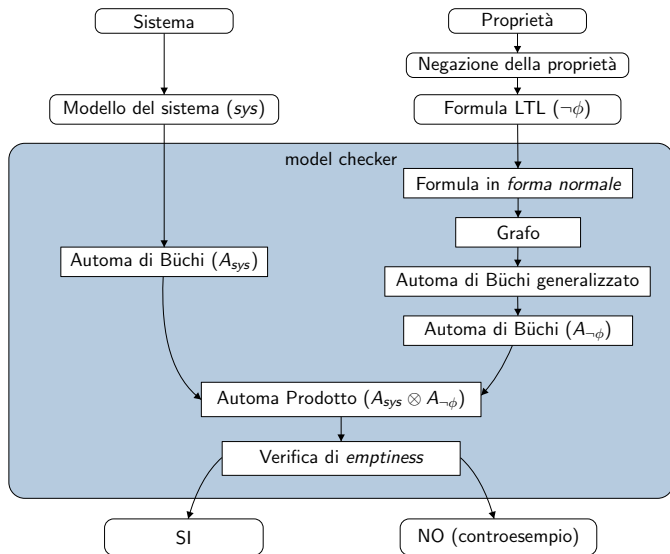
Infine, osservando che $\mathcal{L}_\omega(\overline{A_\phi}) = \mathcal{L}_\omega(A_{\neg\phi})$, si arriva ad un metodo efficiente di model checking:

- 1 Costruire l'LBA per $\neg\phi \implies A_{\neg\phi}$
- 2 Costruire l'LBA per il modello del sistema $\implies A_{sys}$
- 3 Verificare se $\mathcal{L}_\omega(A_{sys}) \cap \mathcal{L}_\omega(A_{\neg\phi}) = \emptyset$

(i run in comune tra A_{sys} e $A_{\neg\phi}$ violano ϕ , quindi sono indesiderati)

Il terzo passo dell'algoritmo è decidibile in tempo lineare

Model Checker completo



Complessità

Se definiamo con S_{sys} l'insieme degli stati dell'LBA A_{sys} , si ha che

Nel caso peggiore la **complessità temporale** del model checking LTL è $O(|S_{sys}|^2 \times 2^{|\phi|})$

(il fattore $2^{|\phi|}$ è legato alla costruzione del grafo)

Anche se la complessità è esponenziale nella lunghezza della formula LTL, nella pratica le formule sono **abbastanza corte** (max 2 o 3 operatori)

Complessità

Se definiamo con S_{sys} l'insieme degli stati dell'LBA A_{sys} , si ha che

Nel caso peggiore la **complessità temporale** del model checking LTL è $O(|S_{sys}|^2 \times 2^{|\phi|})$

(il fattore $2^{|\phi|}$ è legato alla costruzione del grafo)

Anche se la complessità è esponenziale nella lunghezza della formula LTL, nella pratica le formule sono **abbastanza corte** (max 2 o 3 operatori)



Joost-Pieter Katoen.

Concepts, Algorithms and Tools for Model Checking.

Lecture Notes of the Course “Mechanised Validation of Parallel Systems” (1998/1990).