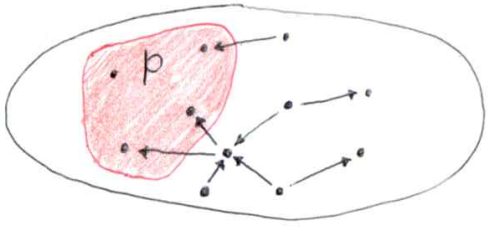
Un algoritmo per CTL

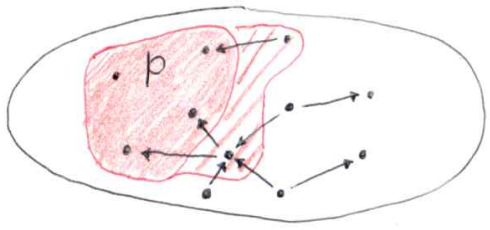
Teoremi di punto fisso e logiche temporali



Fp

Un algoritmo per CTL

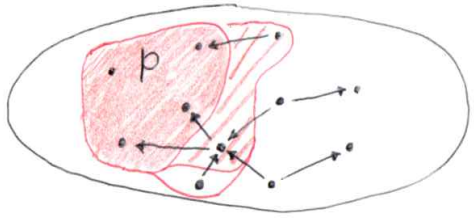
Teoremi di punto fisso e logiche temporali



Fp

Un algoritmo per CTL

Teoremi di punto fisso e logiche temporali

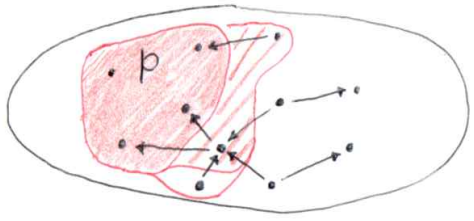


$F\,p$

min punto fisso

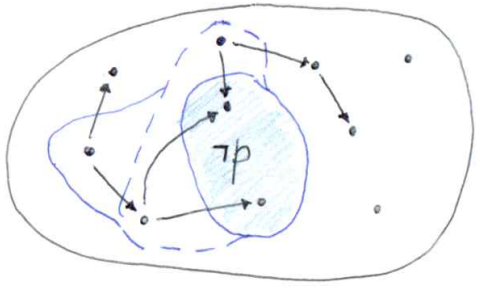
Un algoritmo per CTL

Teoremi di punto fisso e logiche temporali



Fp

min punto fisso



Gp

max punto fisso

Un algoritmo per CTL

$$M = (Q, T, I)$$

Sia α una formula; definiamo l'estensione di α:

$$[\![\alpha]\!] = \{q \in Q \mid M, q \models \alpha\}$$

Un algoritmo per CTL

$$M = (Q, T, I)$$

Consideriamo la formula $\alpha \equiv AF\,\beta$

A questa formula associamo una funzione $f_\alpha : 2^Q \longrightarrow 2^Q$

Per ogni $H \subseteq Q$,

$$f_\alpha(H) = [\![\beta]\!] \cup \{q \in Q \mid \forall (q, q') \in T : q' \in H\}$$

Osservazione: $f_\alpha(\emptyset) = [\![\beta]\!]$

$[\![\alpha]\!]$ è il minimo punto fisso di f_α

Un algoritmo per CTL

$$M = (Q, T, I)$$

Consideriamo la formula $\alpha \equiv EG\,\beta$

A questa formula associamo una funzione $g_\alpha : 2^Q \longrightarrow 2^Q$

Per ogni $H \subseteq Q$,

$$g_\alpha(H) = [\![\beta]\!] \cap \{q \in Q \mid \exists (q, q') \in T : q' \in H\}$$

Osservazione: $g_\beta(Q) = [\![\beta]\!]$

$[\![\alpha]\!]$ è il massimo punto fisso di g_α

Algoritmi per LTL

Automi finiti che riconoscono parole infinite su un alfabeto finito Σ (a. di Büchi)

$$\mathcal{B} = (Q, q_0, \delta, F)$$

- Q: insieme finito di stati (*locations*)
- $q_0 \in Q$: stato iniziale
- $\delta \subseteq Q \times \Sigma \times Q$: relazione di transizione
- $F \subseteq Q$: insieme degli stati accettanti

Una parola infinita $w = a_0 a_1 \cdots$ è accettata da $\mathcal{B}$ se la sequenza corrispondente di stati $q_0 q_1 \cdots$ passa infinite volte per almeno uno stato in F

Algoritmi per LTL

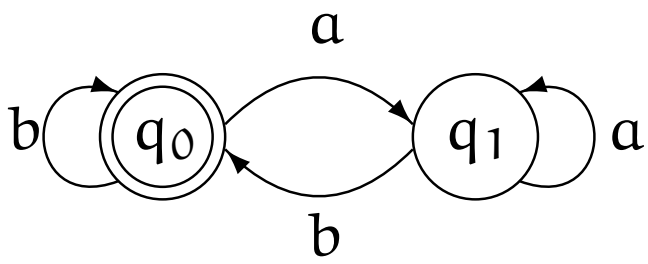
Automi finiti che riconoscono parole infinite su un alfabeto finito Σ (a. di Büchi)

$$\mathcal{B} = (Q, q_0, \delta, F)$$

- Q: insieme finito di stati (*locations*)
- $q_0 \in Q$: stato iniziale
- $\delta \subseteq Q \times \Sigma \times Q$: relazione di transizione
- $F \subseteq Q$: insieme degli stati accettanti

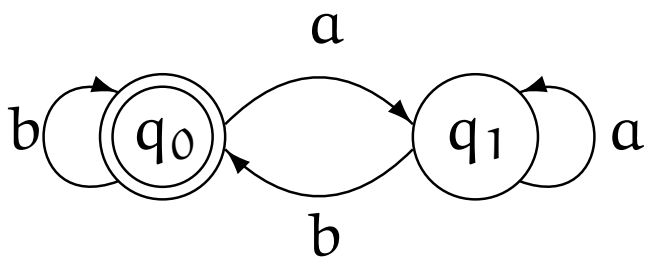
Il problema $L(\mathcal{B}) = \emptyset?$ è decidibile

Algoritmi per LTL



$w_1 = \mathrm{bbbbbbbbb}\cdots$
$w_2 = \mathrm{bbaaabbbb}\cdots$
$w_3 = \mathrm{bababababab}\cdots$
$w_4 = \mathrm{baabbbaaa}\cdots$

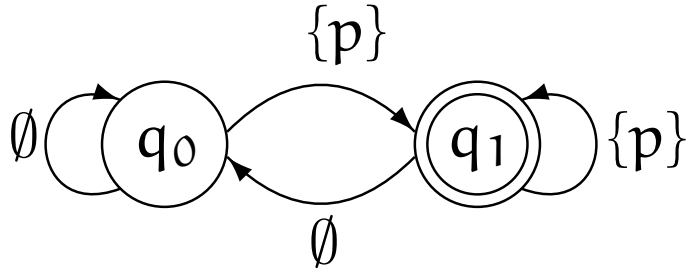
Algoritmi per LTL



$w_1 = \mathtt{bbbbbbbbb}\cdots$ sì
$w_2 = \mathtt{bbaaabbbb}\cdots$ sì
$w_3 = \mathtt{bababababab}\cdots$ sì
$w_4 = \mathtt{baabbbaaa}\cdots$ no

Un algoritmo per LTL

GFp



$w_1 = \emptyset\{p\}\{p\}\emptyset\{p\}\emptyset\emptyset\cdots$ no

$w_2 = \emptyset\{p\}\emptyset\{p\}\emptyset\{p\}\emptyset\cdots$ sì

Un algoritmo per LTL

Problema: verificare se α è vera in (M, q_0)

(1) Costruiamo l'automa $\mathcal{B}_{\neg\alpha}$

(2) Trasformiamo M in un automa etichettato da insiemi di proposizioni atomiche

(3) Calcoliamo il prodotto sincrono dei due automi $\mathcal{PS}$

(4) Se $L(\mathcal{PS}) = \emptyset$, allora $M, q_0 \models \alpha$

Il calcolo μ

Un linguaggio logico che permette di definire formule ricorsive

Il calcolo μ

Supponiamo di avere un solo operatore temporale: X. Come esprimere la proprietà $\mathrm{EF}\,\alpha$?

Il calcolo μ

Supponiamo di avere un solo operatore temporale: X. Come esprimere la proprietà $\mathrm{EF}\,\alpha$?

$$\mathrm{EF}\,\alpha \equiv \alpha \vee \mathrm{EX}\,\alpha \vee \mathrm{EXEX}\,\alpha \vee \cdots$$

Il calcolo μ

Supponiamo di avere un solo operatore temporale: X. Come esprimere la proprietà $\mathrm{EF}\,\alpha$?

$$\mathrm{EF}\,\alpha \equiv \alpha \vee \mathrm{EX}\,\alpha \vee \mathrm{EXEX}\,\alpha \vee \cdots$$

"Raccogliamo" EX:

$$\mathrm{EF}\,\alpha \equiv \alpha \vee \mathrm{EX}\,(\alpha \vee \mathrm{EX}\,\alpha \vee \mathrm{EXEX}\,\alpha \vee \cdots)$$

Il calcolo μ

Supponiamo di avere un solo operatore temporale: X. Come esprimere la proprietà $\mathrm{EF}\,\alpha$?

$$\mathrm{EF}\,\alpha \equiv \alpha \vee \mathrm{EX}\,\alpha \vee \mathrm{EXEX}\,\alpha \vee \cdots$$

"Raccogliamo" EX:

$$\begin{aligned}\mathrm{EF}\,\alpha &\equiv \alpha \vee \mathrm{EX}\,(\alpha \vee \mathrm{EX}\,\alpha \vee \mathrm{EXEX}\,\alpha \vee \cdots) \\ &\equiv \alpha \vee \mathrm{EX}(\mathrm{EF}\,\alpha)\end{aligned}$$

Il calcolo μ

Supponiamo di avere un solo operatore temporale: X. Come esprimere la proprietà $\mathrm{EF}\,\alpha$?

$$\mathrm{EF}\,\alpha \equiv \alpha \vee \mathrm{EX}\,\alpha \vee \mathrm{EXEX}\,\alpha \vee \cdots$$

"Raccogliamo" EX:

$$\begin{aligned}\mathrm{EF}\,\alpha &\equiv \alpha \vee \mathrm{EX}\,(\alpha \vee \mathrm{EX}\,\alpha \vee \mathrm{EXEX}\,\alpha \vee \cdots) \\ &\equiv \alpha \vee \mathrm{EX}(\mathrm{EF}\,\alpha)\end{aligned}$$

$$\mu Y.(\alpha \vee \mathrm{EX}\,Y)$$

Il calcolo μ

Supponiamo di avere un solo operatore temporale: X. Come esprimere la proprietà $AG\,\alpha$?

$$AG\,\alpha \equiv \alpha \wedge AX\,\alpha \wedge AXAX\,\alpha \wedge \cdots$$

"Raccogliamo" AX:

$$\begin{aligned} AG\,\alpha &\equiv \alpha \wedge AX\,(\alpha \wedge AX\,\alpha \wedge AXAX\,\alpha \wedge \cdots) \\ &\equiv \alpha \wedge AX(AG\,\alpha) \end{aligned}$$

$$\nu Y.(\alpha \wedge AX\,Y)$$

Il calcolo μ

$AP = \{p_1, p_2, \cdots, q, r, \cdots\}$ proposizioni atomiche

Siano α e β due formule

1. $\alpha \vee \beta$, $\neg\alpha$ sono formule
2. $EX\,\alpha$, $AX\,\alpha$ sono formule
3. $\mu Y.f(Y)$ è una formula, dove f è una formula nella quale compare Y (con restrizioni sulle negazioni)
4. $\nu Y.f(Y)$ è una formula, dove f è una formula nella quale compare Y (con restrizioni sulle negazioni)

La semantica del calcolo μ è definita su modelli di Kripke attraverso operatori di punto fisso

Il calcolo μ

$$CTL^* \subset \mu - \text{calculus}$$

Calcolo μ: massima potenza espressiva, alta complessità, potenziale "oscurità" delle formule

Complessità e aspetti algoritmici

M: modello di Kripke f: formula

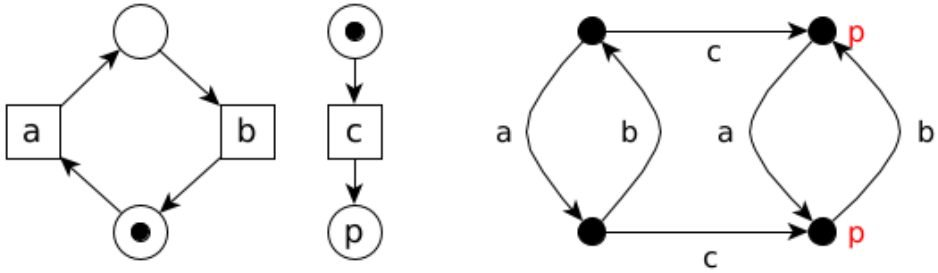
$$\text{CTL: } O(|M| \times |f|) \qquad \text{LTL: } O(|M| \times 2^{|f|})$$

Le stime di complessità vanno interpretate *cum grano salis*

Strategie algoritmiche:

- Rappresentazioni simboliche (OBDD)
- Partial order reduction (unfolding)
- Traduzione in SAT

Fairness

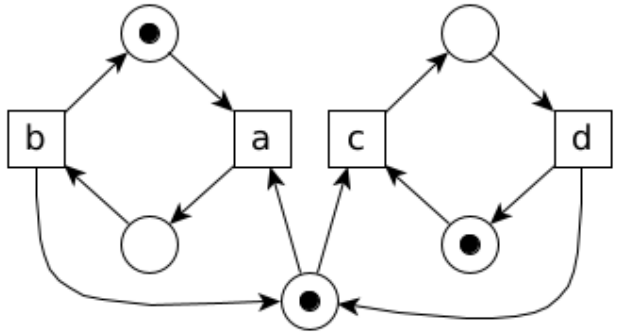


F p ?

Un'esecuzione è unfair (iniqua, ingiusta) se un evento rimane sempre abilitato da un istante in poi, ma non scatta mai

Problema: limitare la valutazione di una formula alle esecuzioni *fair*

Fairness



L'esecuzione $\mathtt{abababa}\cdots$ è debolmente *fair*

Un'esecuzione è fortemente fair se

$$\mathrm{GF}(\text{t abilitata}) \longrightarrow \mathrm{GF}(\text{t scatta})$$

$((\mathtt{ab})(\mathtt{cd})^{10})^{\infty}$ è fortemente *fair*

Software per il model-checking

Spin	LTL (Promela)
NuSMV	CTL, LTL
mCLR2	calcolo μ
TiNA	LTL (reti di Petri)

e molti altri...