Lezione 22 08/01/2024

Abbiamo visto che per stabilire se una formula è vera dobbiamo considerare i possibili cammini massimali del sistema sottostante. Possiamo considerarli indipendentemente l'uno dall'altro.

Dato un automa a stati finiti, diremo che una parola di lunghezza infinita è accettato dall'automa se questo passerà infinite volte per uno degli stati finali.

Negli automi le transizioni sono etichettate un un certo alfabeto, mentre nei modelli di kripti le transizioni non sono etichettate, quello che conta sono le proposizioni atomiche che sono vere nei vari stati. Per ora lasciamo da parte questa cosa.

Problema: dato un modello di Kripke su AP e una formula LTL, decidere se la formula è verificata nello stato iniziale

Non possiamo applicare l'algoritmo per CTL

Automi di Büchi

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

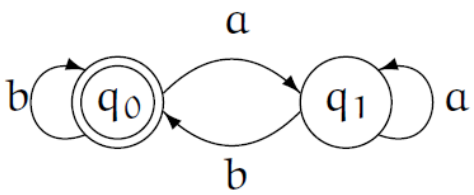
$$\mathcal{B} = (Q, q_0, \delta, F)$$

- 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

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

Il problema di stabilire se il linguaggio riconosciuto da un'automa è vuoto, è decidibile. Ovvero esiste un automa che lo decide in tempo finito.



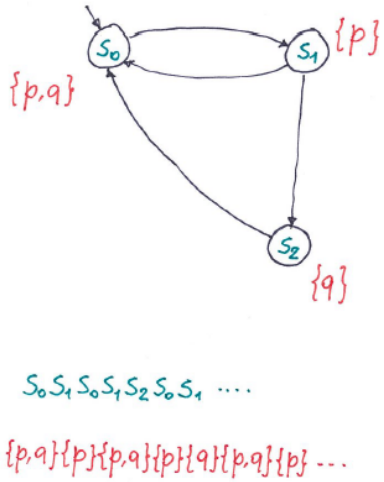
$w_1 = \text{bbbbbbbbb}\cdots$ sì
$w_2 = \text{bbaaabbbb}\cdots$ sì
$w_3 = \text{babababab}\cdots$ sì
$w_4 = \text{baabbbaaa}\cdots$ no

parola 2 dopo un certo punto ha tutte b

parola 3 alterna a e b all'infinito

parola 4 dopo un certo punto ha tutte a

Torniamo al problema che i modelli di kripti non hanno etichette sulle transizioni. Abbiamo bisogno di trasformare il modello per spostare le transizioni atomiche sugli archi.

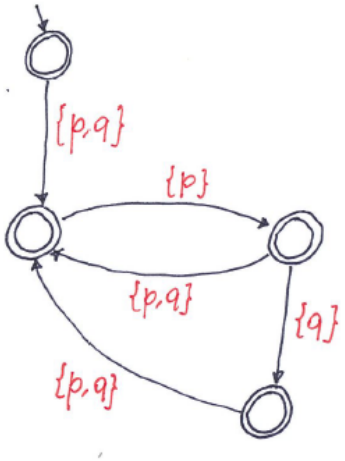


Consideriamo il cammino massimale che rimane in S0 e S1.

Per ogni stato sappiamo quali sono le proposizioni atomiche vere nello stato, quindi possiamo descrivere il cammino massimale attraverso le proposizioni vere nello stato.

Questa sequenza di insiemi può essere vista come una parola su un linguaggio che ha come simboli del linguaggio, insiemi di proposizioni atomiche.

Possiamo trasformare il modello in un automa di Buchi che riconosce tutte e sole le parole infinite di questo modello.



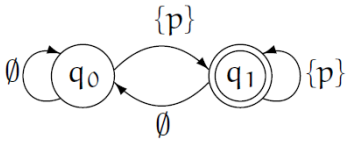
Ogni stato è finale, ci interessano tutti i cammini finiti.

Dallo stato iniziale dell'automa con {p, q} andiamo in quello che è lo stato iniziale del modello originale.

La parola generata nell'automa corrisponde alla sequenza di transizioni atomiche del modello.

Possiamo disegnare un'automa di Buchi in modo tale che l'automa riconosca le sequenze di proposizioni vere. ?

GFp



$w_1 = \emptyset\{p\}\{p\}\emptyset\{p\}\emptyset\emptyset\cdots$ no
$w_2 = \emptyset\{p\}\emptyset\{p\}\emptyset\{p\}\emptyset\cdots$ sì

In quali casi una sequenza di sottoinsiemi di proposizioni atomiche soddisfa questa formula?

La formula dice che è sempre vero che prima o poi p. Ovvero p sarà vera infinite volte nel cammino.

in w1 poi c'è l'insieme vuoto all'infinito

Se un modello di Kripke non soddisfa una formula vuol dire che c'è almeno un cammino massimale dove la formula non è vera.

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

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:

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

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

L'ultima riga prima di quella blu è una definizione ricorsiva.

Questa ricorsione si può definire in un linguaggio.

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

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

Questa logica espressiva è più espressiva di CTL*

Complessità e aspetti algoritmici

M: modello di Kripke f: formula

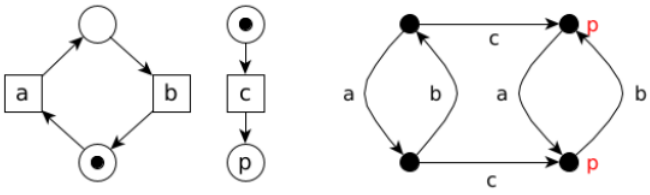
CTL: $O(|M| \times |f|)$ LTL: $O(|M| \times 2^{|f|})$

Le stime di complessità vanno interpretate *cum grano salis*

Strategie algoritmiche:

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

Fairness

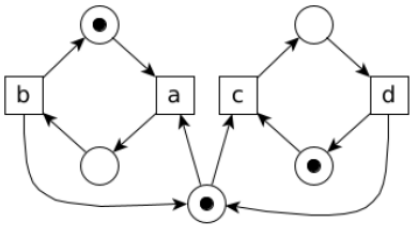


F p ?

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*

Nello stato iniziale apparentemente c'è una scelta tra a e c, ma nel sistema reale (a sx) questi sono indipendenti. Per uscire da questo problema possiamo ricorrere alla nozione di fairness.



L'esecuzione $abababa\cdots$ è debolmente *fair*

Un'esecuzione è fortemente fair se

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

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

cosa si aspetta che sappiamo di queste cose:

- rispetto a LTL dobbiamo avere chiaro lo schema dell'algoritmo, cos'è l'automa di … e quali proprietà si sfruttano per l'analisi di formule.
- Calcolo di u sapere che esiste e perchè
- Sulla fairness che la sappiamo spiegare