Automi Ibridi

Carla Piazza[1]

[1] Dipartimento di Matematica ed Informatica
Università di Udine
carla.piazza@dimi.uniud.it

Indice del Corso (Dis)Ordinato

- Automi Ibridi: Sintassi e Semantica
- Sistemi a stati finiti (breve ripasso)
- Il problema della Raggiungibilità
- Risultati di Indecidibilità
- Classi notevoli di Automi Ibridi: timed, rectangular, o-minimal, ...
- Tecniche di Decisione: (Bi)Simulazione, Cylindric Algebraic Decomposition, Teoremi di Selezione, Semantiche approssimate
- ... e tanto altro:
  - Logiche temporali
  - Composizione di Automi
  - Il caso Stocastico
  - Stabilità, Osservabilità, Controllabilità
  - Strumenti Software
  - Applicazioni

In particolare parleremo di . . .

- Alcune Idee di Base sul CTL Model Checking
- Un po' di Storia e Personaggi
- Model Checking Esplicito
- State Explosion Problem
- Model Checking Simbolico
- Altri Approcci: On-the-fly MC, Abstract MC, (Bi)simulation

Intuitivamente il Model Checking . . .

Consideriamo

- un linguaggio di specifica per esprimere proprietà
- un sistema hardware (o software)
- una specifica espressa nel linguaggio

Vogliamo testare se

il sistema soddisfa la specifica

Un po' più formalmente il Model Checking . . .

Abbiamo un sistema reattivo concorrente hardware/software
Vogliamo verificare se il sistema soddisfa alcune specifiche

Un po' più formalmente il Model Checking . . .

Abbiamo un sistema reattivo concorrente hardware/software
Vogliamo verificare se il sistema soddisfa alcune specifiche

H/S Sistema S	$\Rightarrow$	Sistema di Transizione $\mathcal{M}$
Specifica F	$\Rightarrow$	Formula Temporale f

Un po' più formalmente il Model Checking . . .

Abbiamo un sistema reattivo concorrente hardware/software
Vogliamo verificare se il sistema soddisfa alcune specifiche

H/S Sistema S	$\Rightarrow$	Sistema di Transizione $\mathcal{M}$
Specifica F	$\Rightarrow$	Formula Temporale f

Il problema ora è:

$$\mathcal{M} \models f$$

i.e., $\mathcal{M}$ soddisfa la formula f?

Model Checking: facile o difficile?

Il problema $\mathcal{M} \models f$ sembra quasi banale

Model Checking: facile o difficile?

Il problema $\mathcal{M} \models f$ sembra quasi banale

Dobbiamo risolverlo efficientemente

Model Checking: facile o difficile?

Il problema $\mathcal{M} \models f$ sembra quasi banale

Dobbiamo risolverlo efficientemente

Più in dettaglio:

- $\mathcal{M}$ è un grafo etichettato su nodi e archi
- f è una formula che parla di propertà del grafo

Model Checking: facile o difficile?

Il problema $\mathcal{M} \models f$ sembra quasi banale

Dobbiamo risolverlo efficientemente

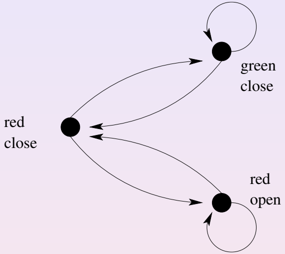
Più in dettaglio:

- $\mathcal{M}$ è un grafo etichettato su nodi e archi
- f è una formula che parla di propertà del grafo

Possiamo risolverlo in tempo polinomiale? E in tempo lineare?

What about space complexity?

Esempio: Passaggio a Livello



- Non vogliamo che il semaforo sia verde per il treno quando il passaggio a livello è aperto (safety)

$$AG\neg(green \wedge open)$$

- Non vogliamo che il treno aspetti per sempre (liveness)

$$red \rightarrow EF(green)$$

I Dettagli Tecnici

1. Come trasformo un sistema in un grafo?
2. Come trasfromo le specifiche da testare in formule?

Esempio: Mutua Esclusione

$$P :: \quad m : \text{cobegin } P_0 | P_1 \text{ coend } m'$$

$$\begin{array}{ll} P_0 :: \quad l_0 : \text{while TRUE do} & \\ \qquad nc_0 : \text{wait}(turn = 0); & \\ \qquad cr_0 : \text{crit}(turn := 1); & \\ \text{end while}; \quad l_0' & \end{array}$$

$$\begin{array}{ll} P_1 :: \quad l_1 : \text{while TRUE do} & \\ \qquad nc_1 : \text{wait}(turn = 1); & \\ \qquad cr_1 : \text{crit}(turn := 0); & \\ \text{end while}; \quad l_1' & \end{array}$$

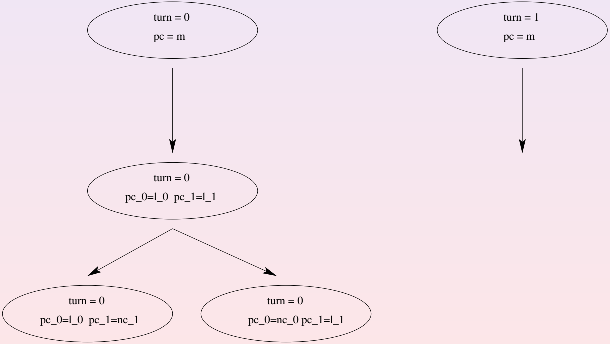
Il programma ha 4 variabili: $turn$, pc, pc_0, e pc_1

È garantita la mutua esclusione? È garantita la non-starvation?

Clarke, Grumberg e Peled. Model Checking. 1999.

Esempio: Mutua Esclusione

- Stato: assegnamento di valori alle variabili
- Transizione: passaggio di stato



Sistemi di Transizione (Kripke Structures)

Definition (Sistema di Transizione)

Sia AP un insieme di proposizioni atomiche
Un sistema di transizione è una quatrupla $\mathcal{M} = (S, S_0, R, L)$ in cui

- S è un insieme finito di stati
- $S_0 \subseteq S$ è l'insieme degli stati iniziali
- $R \subseteq S \times S$ è la relazione di transizione
- $L : S \to 2^{AP}$ è una funzione etichettatrice che stabilisce le proposizioni atomiche vere in uno stato

Le Specifiche da Testare

- Invariance Conditions: qualcosa vale in ogni stato raggiungo
- Safety Properties: qualcosa di cattivo non succede mai lungo nessuna computazione
- Liveness Properties: qualcosa di buono prima o poi succede lungo ogni computazione
- ...

Serve una logica che parli di computazioni, ovvero che parli di cammini su grafi

La Logica Temporale CTL - Sintassi

Definition (CTL - Sintassi)

Sia AP un insieme di proposizioni atomiche

- ogni proposizione atomica $p \in AP$ è una formula
- se f e g sono formule allora lo sono anche

$\neg f$	$f \vee g$
$\mathbf{AX}\, f$	$\mathbf{EX}\, f$
$\mathbf{A}\,(f\ \mathbf{U}\ g)$	$\mathbf{E}\,(f\ \mathbf{U}\ g)$

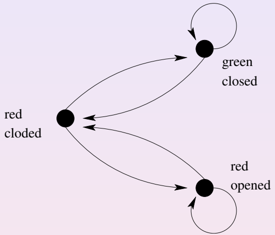
Si definiscono da queste $\mathbf{AF}\, p$, $\mathbf{EF}\, p$, $\mathbf{EG}\, p$, $\mathbf{AG}\, p$

La Logica Temporale CTL - Semantica

Definition (CTL - Semantica)

- $(S, S_0, R, L), s \models p$ sse $p \in L(s)$
- $(S, S_0, R, L), s \models \mathbf{AX}\, f$ sse per ogni s' tale che $(s, s') \in R$ vale $(S, S_0, R, L), s' \models f$
- $(S, S_0, R, L), s \models \mathbf{EX}\, f$ sse esiste s' tale che $(s, s') \in R$ e $(S, S_0, R, L), s' \models f$
- $(S, S_0, R, L), s \models \mathbf{A}\,(f\, \mathbf{U}\, g)$ sse per ogni cammino che parte da s è vera f finchè non diventa vera g
- $(S, S_0, R, L), s \models \mathbf{E}\,(f\, \mathbf{U}\, g)$ sse esiste un cammino che parte da s lungo cui è vera f finchè non diventa vera g

Esempio: Passaggio a Livello



- Non è mai verde per il treno quando il passaggio a livello è aperto (safety)
  AG $\neg$(green $\wedge$ open)
- Se è rosso prima o poi diventerà verde (liveness)
  red $\rightarrow$ **EF** green

Storia e Personaggi

- Manna e Pnueli. Logiche Temporali. 1981.
- Clarke, Emerson e Sistla. Quielle e Sifakis. Sistemi di Transizione. 1983.
- Vengono studiati algoritmi efficienti per varie logiche.
- Si evidenzia il problema dell'esplosione degli stati nelle applicazioni pratiche
- Mc Millan, Clarke ed altri. Symbolic Model Checking. 1993.
- Dams, Gerth e Grumberg. Abstract Model Checking. 1996.
- ...

Model Checking Esplicito

Theorem

Dato un sistema $\mathcal{M} = (S, S_0, R, L)$, uno stato $s \in S$ ed una formula f di CTL è possibile determinare se $\mathcal{M}, s \models f$ in tempo $O(|f|(|S| + |R|))$

- Si fanno delle visite di $\mathcal{M}$ che servono per etichettare gli stati di S che soddisfano le sottoformule di f
- se $\mathcal{M}$ non soddisfa f, viene fornito in output un contro-esempio
- Nei casi pratici $\mathcal{M}$ è troppo grande per essere mantenuto in memoria e l'algoritmo diventa inapplicabile

Model Checking Esplicito

Theorem

Dato un sistema $\mathcal{M} = (S, S_0, R, L)$, uno stato $s \in S$ ed una formula f di CTL è possibile determinare se $\mathcal{M}, s \models f$ in tempo $O(|f|(|S| + |R|))$

- Si fanno delle visite di $\mathcal{M}$ che servono per etichettare gli stati di S che soddisfano le sottoformule di f
- se $\mathcal{M}$ non soddisfa f, viene fornito in output un contro-esempio
- Nei casi pratici $\mathcal{M}$ è troppo grande per essere mantenuto in memoria e l'algoritmo diventa inapplicabile

Model Checking Esplicito

Theorem

Dato un sistema $\mathcal{M} = (S, S_0, R, L)$, uno stato $s \in S$ ed una formula f di CTL è possibile determinare se $\mathcal{M}, s \models f$ in tempo $O(|f|(|S| + |R|))$

- Si fanno delle visite di $\mathcal{M}$ che servono per etichettare gli stati di S che soddisfano le sottoformule di f
- se $\mathcal{M}$ non soddisfa f, viene fornito in output un contro-esempio
- Nei casi pratici $\mathcal{M}$ è troppo grande per essere mantenuto in memoria e l'algoritmo diventa inapplicabile

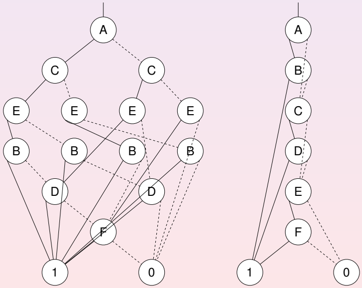
Model Checking Simbolico

- Problema: un metodo per rappresentare $\mathcal{M}$ in maniera compatta
- Soluzione: Ordered Binary Decision Diagrams
- Problema: ridisegnare tutti gli algoritmi

Ordered Binary Decision Diagrams – OBDDs

Sono grafi aciclici che rappresentano Funzioni Booleane

$$(A \wedge B) \vee (C \wedge D) \vee (E \wedge F)$$



OBDDs e Sistemi di Transizione

$S = \{0,1\}^u$, i.e. ogni nodo viene codificato in binario
$N \subseteq S$ è un insieme di stringhe binarie di lunghezza u,

$$\chi_N(n_1,\ldots,n_u) = 1 \quad \Leftrightarrow \quad \langle n_1,\ldots,n_u\rangle \in N$$

è una funzione booleana rappresentabile via OBDD

$R \subseteq S \times S$ è un insieme di stringhe binarie di lunghezza $2u$

$$\chi_R(x_1,\ldots,x_u,y_1,\ldots,y_u) = 1 \quad \Leftrightarrow \quad x_1,\ldots,x_u R y_1,\ldots,y_u$$

è una funzione booleana rappresentabile via OBDD

OBDDs e Sistemi di Transizione

$S = \{0,1\}^u$, i.e. ogni nodo viene codificato in binario
$N \subseteq S$ è un insieme di stringhe binarie di lunghezza u,

$$\chi_N(n_1,\ldots,n_u) = 1 \quad \Leftrightarrow \quad \langle n_1,\ldots,n_u\rangle \in N$$

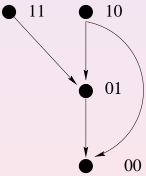
è una funzione booleana rappresentabile via OBDD

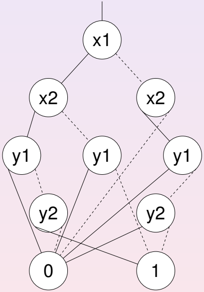
$R \subseteq S \times S$ è un insieme di stringhe binarie di lunghezza $2u$

$$\chi_R(x_1,\ldots,x_u,y_1,\ldots,y_u) = 1 \quad \Leftrightarrow \quad x_1,\ldots,x_u R y_1,\ldots,y_u$$

è una funzione booleana rappresentabile via OBDD

Esempio: da Grafo ad OBDD





OBDDs e Algoritmi

Gli algoritmi simbolici sono basati su: $\cup, \cap, \setminus, =, \ldots, R, R^{-1}$

Sia $N \subseteq S$, calcolare

$$R(N)$$

costa 1 operazione simbolica indipendentemente da $|N|$

le visite in ampliezza costano meno di quelle in profondità

L'algoritmo di Tarjan per le componenti fortemente connesse è inapplicabile

OBDDs e Algoritmi

Gli algoritmi simbolici sono basati su: $\cup, \cap, \setminus, =, \ldots, R, R^{-1}$

Sia $N \subseteq S$, calcolare

$$R(N)$$

costa 1 operazione simbolica indipendentemente da $|N|$

le visite in ampliezza costano meno di quelle in profondità

L'algoritmo di Tarjan per le componenti fortemente connesse è inapplicabile

OBDDs e Algoritmi

Gli algoritmi simbolici sono basati su: $\cup, \cap, \setminus, =, \ldots, R, R^{-1}$

Sia $N \subseteq S$, calcolare

$$R(N)$$

costa 1 operazione simbolica indipendentemente da $|N|$

le visite in ampliezza costano meno di quelle in profondità

L'algoritmo di Tarjan per le componenti fortemente connesse è inapplicabile

OBDDs e Algoritmi

Gli algoritmi simbolici sono basati su: $\cup, \cap, \setminus, =, \ldots, R, R^{-1}$

Sia $N \subseteq S$, calcolare

$$R(N)$$

costa 1 operazione simbolica indipendentemente da $|N|$

le visite in ampliezza costano meno di quelle in profondità

L'algoritmo di Tarjan per le componenti fortemente connesse è inapplicabile

Altri Approcci

- Abstract Model Checking. Sostituire $\mathcal{M}$ con una sua astrazione in cui alcuni stati sono collassati. È possibile che una formula non sia vera nel modello astratto, ma sia vera in quello concreto.
- Bisimulatione. Se si quozienta il modello usando la massima bisimulazione il ridotto modello soddisfa le stesse formule. È possibile che il modello ridotto sia troppo grande. È possibile che l'algoritmo di bisimulazione sia più costoso di quello di Model Checking. Vardi.
- Simulazione. Riduce maggiormente rispetto alla bisimulazione ed è corretta e completa per formule che hanno solo quantificatori universali (esistenziali).

Riferimenti Bibliografici

- Model Checking. Clarke, Grumberg and Peled. MIT Press, 1999
- The birth of model checking. Clarke. Springer, 2008
- The Beginning of Model Checking: A Personal Perspective. Emerson,
- Symbolic Model Checking. Mc Millan Phd Thesis, 1993