Verifica formale di proprietà mediante model checking

Angelo Gargantini

May 8, 2024

Outline

- model checking algorithm for CTL logics

TESTO DI RIFERIMENTO: M.R.A. Ruth, M.D. Ryan Logic in Computer Science Modelling and Reasoning about systems - Capitolo 3 - allegato a questi appunti

5.1 Model-checking algorithms

Model checking algorithm

- We want to solve this problem $M, s \overset{???}{\models} \phi$
- There are everal approaches to formal verification
  - (automated) theorem prover
  - **model checking** (several types) ... Spin, PRISM, JavaPathFinder, ...

Model checking algorithm - CTL

- We want to solve this problem $M, s \overset{???}{\models} \phi$
  - model checking
  - by a labelling algorithm:
    - INPUT: a model M = (S, $\rightarrow$, L) and a CTL formula φ .
    - OUTPUT: the set of states of M which satisfy φ.

  1. First, change φ in terms of the connectives AF, EU, EX, $\bigwedge$, $\neg$ and $\bot$ using the equivalences given earlier.
  2. Next, label the states of M in which φ holds

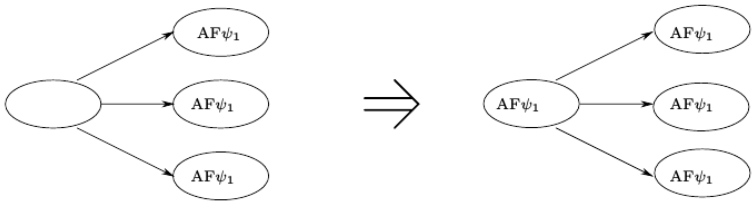
Labeling algorithm

Case analysis over ψ. If ψ is

- $\bot$: then no states are labelled with $\bot$
- p: then label every s such that p $\in L(s)$
- $\psi 1 \bigwedge \psi 2$:
  - do labelling with $\psi 1$ and with $\psi 2$
  - label s with $\psi 1 \bigwedge \psi 2$ if s is already labelled both with $\psi 1$ and with $\psi 2$
- $\neg\psi$:
  - do labelling with ψ
  - label s with $\neg\psi 1$ if s is not labelled with ψ.

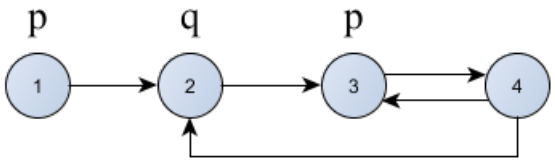
Labeling algorithm - AF

- AF ψ :
  - do labeling with ψ
    - If any state s is labelled with ψ , label it with AF ψ.
    - Repeat: label any state with AF ψ if all successor states are labelled with AF ψ , until there is no change. See picture

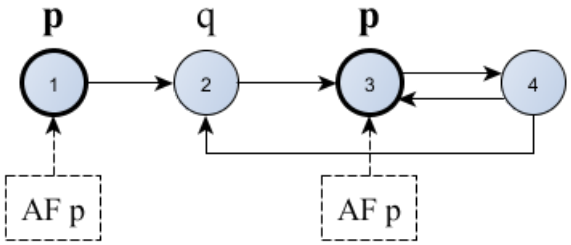


Esempio: *labeling* AF

Sia il seguente schema dove è già stato fatto il labeling per p e q:



Si farà il labeling per $AF\,p$: dove vale p si mette anche $AF\,p$.

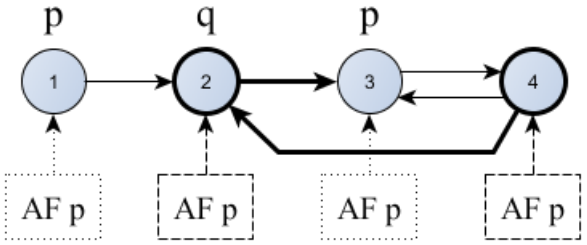


Ora si guarda di stato in stato per vedere se etichettarli come $AF\,p$ (1) Già etichettato con $AF\,p$

(2) Ho la transizione (2) $\rightarrow$ (3) e quindi avendo nello stato (3) $AF\,p$ allora anche (2) è etichettato.

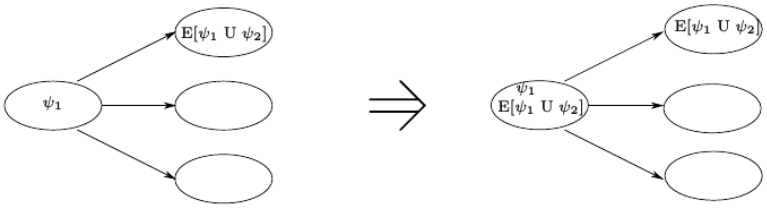
(3) Già etichettato con $AF\,p$

(4) Come prima, ho (4) $\rightarrow$ (2) con (2) etichettato $AF\,p$, quindi anche (4)è etichettato. Si è ottenuto il seguente grafico:



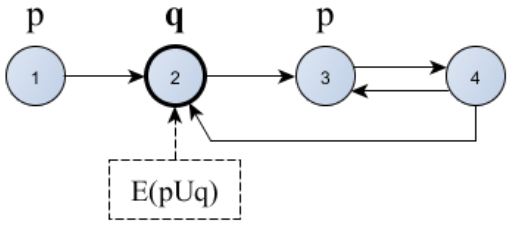
Labeling algorithm - EU

- E[ψ1 U ψ2]
  - do labeling for ψ1 and ψ2
    - If any state s is labelled with ψ2, label it with E[ψ1 U ψ2]
    - Repeat: label any state with E[ψ1 U ψ2] if it is labelled with ψ1 and at least one of its successors is labelled with E[ψ1 U ψ2], until there is no change.

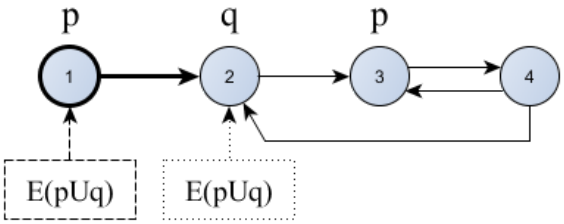


Esempio: *labeling EU*

Riprendiamo lo stesso schema dell'esercizio precedente ed etichettiamo per $E(p\,U\,q)$: si inizia mettendo l'etichetta $E(p\,U\,q)$ a tutti gli stati con q.



Cerco negli stati etichettati con p se esiste uno stato successivo etichettato con $E(p\,U\,q)$: in caso affermativo, anche lo stato con p è etichettato. (1) Ho (1) $\rightarrow$ (2) con (2) etichettato, quindi si etichetta anche (1)
(2) Non ha p
(3) Non esiste uno stato successivo etichettato con $E(p\,U\,q)$
(4) Non ha p Quindi si è ottenuto il seguente grafico:



Labeling algorithm - EX

- EX ψ :
  - do labeling for ψ
    - label any state with EXψ if one of its successors is labelled with ψ

Complessità

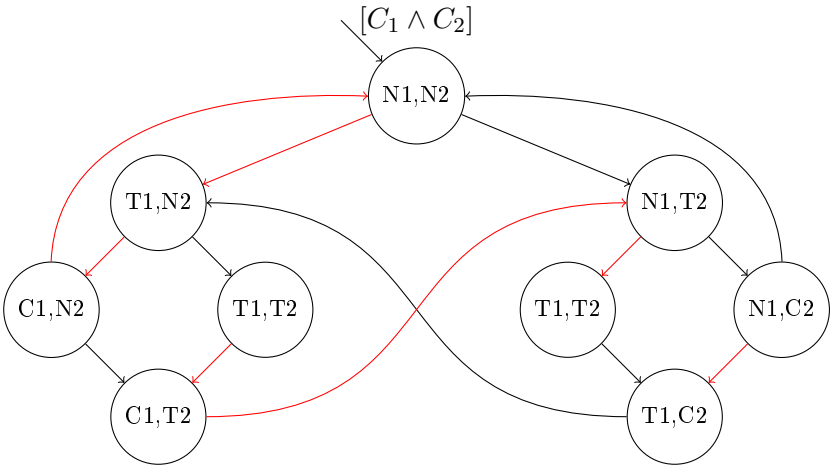
- The complexity of this algorithm is O(f · V · (V + E)), where f is the number of connectives in the formula, V is the number of states and E is the number of transitions; the algorithm is linear in the size of the formula and quadratic in the size of the model.

- Some improvements
  - Handling EG directly
- LTL is treated differently (skip)

Altri casi

Come faccio in tutti gli altri casi????? TODO

Esempio



State Explosion problem

The 'state explosion' problem Although the labelling algorithm (with the clever way of handling EG) is linear in the size of the model, unfortunately the size of the model is itself more often than not exponential in the number of variables and the number of components of the system which execute in parallel. This means that, for example, adding a boolean variable to your program will double the complexity of verifying a property of it. The tendency of state spaces to become very large is known as the state explosion problem. A lot of research has gone into finding ways of overcoming it, including the use of:

- efficient data structure BDDs
  - ....

L'algoritmo è lineare rispetto alle dimensioni sia di φ che di $\mathcal{M}$, tuttavia la dimensione di quest'ultimo è molto spesso esponenziale rispetto al numero di variabili e al munero di componenti del sistema che eseguono in parallelo; questo problema è noto come il **problema di esplosione degli stati**: si sono creati così tanti stati da aver occupato tutta la memoria disponibile.

Una possibile soluzione è utilizzare le strutture dati BDD (Binary Decision Diagram) che utilizzano una versione grafica alternativa all'albero (utilizzato nella logica proposizionale) per rappresentare le proposizioni.

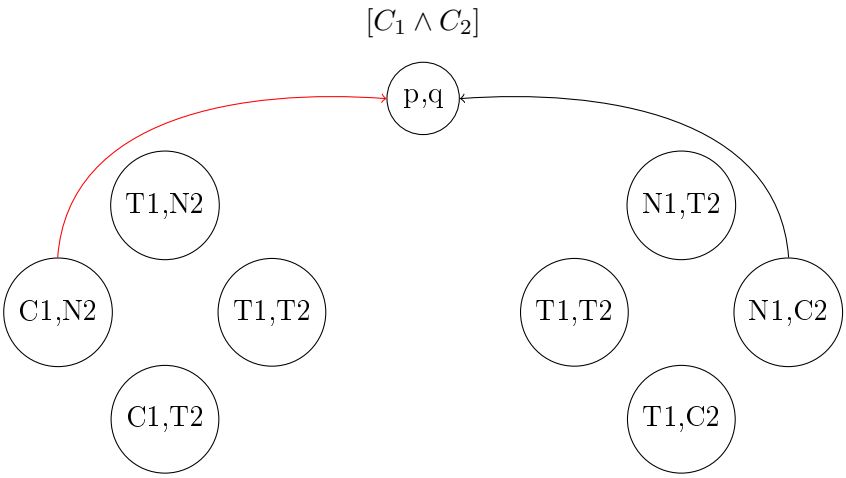
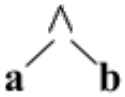
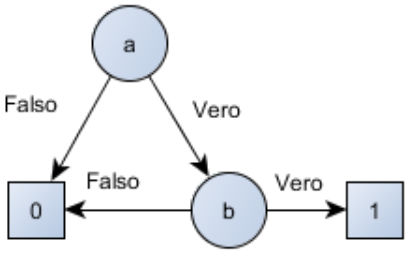


Figura 5.1: A machine

Espressione: $a \wedge b$



Albero della logica proposizionale



Binary Decision Diagram

Exercises