Model Checker per la logica STL

Gianluca Grilletti

26 agosto 2014

Nel seguente documento vengono presentati brevemente la sintassi della logica STL (*space-temporal logic*) e i comandi principali del modelchecker che implementa tale logcia. Viene in seguito presentato un esempio d'uso del model checker commentato.

In seguito verranno utilizzate le notazioni introdotte in [1] sugli spazi topologici quasi-discreti.

1 La logica STL

La sintassi della logica STL è data dalla seguente grammatica, dove p varia tra un insieme P di proposizioni atomiche

$$\Phi ::= p \mid \bot \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{N}\Phi \mid [\Phi]\mathcal{S}[\Phi] \mid E\Psi \mid A\Psi \tag{1}$$

$$\Psi ::= X\Phi \mid F\Phi \mid [\Phi]U[\Phi] \tag{2}$$

Un modello per tale logica è una struttura $M = ((X, C), (S, R), V_{s\in S})$ dove

- (X, C) è uno spazio topologico quasi-discreto
- (S, R) è un *Kripke frame*
- V_s è una funzione da P (l'insieme delle proposizioni atomiche) a $\mathcal{P}(X)$

Nel seguito ci riferiremo ad X come allo spazio del modello, ad S come al tempo, a $s \in S$ come ad un istante ed a V come all'ambiente del modello.

Quello che rappresenta un modello è uno spazio su cui possono essere valutate delle proprietà. Tali proprietà possono però variare nel tempo ed abbiamo quindi bisongo di avere una struttura temporale che descriva tale cambiamento.

Nel seguito con $\mathcal{P}_s$ indicheremo le sequenze infinite di istanti che partono da s. Dato $\sigma \in \mathcal{P}_s$, con $\sigma(n)$ indichiamo l'n-esimo istante della sequenza. La valutazione semantica in un certo punto dello spazio x, per un certo istante s è data dalle seguenti clausole induttive

- $M, x, s \nvDash \bot$
- $M, x, s \vDash p$ se e solo se $x \in V_s(p)$
- $M, x, s \vDash \neg\phi$ se e solo se $M, x, s \nvDash \phi$
- $M, x, s \vDash \phi \wedge \psi$ se e solo se $M, x, s \vDash \phi$ e $M, x, s \vDash \psi$
- $M, x, s \vDash \mathcal{N}\phi$ se e solo se

$$x \in C\left(\{y \in X \mid M, y, s \vDash \phi\}\right)$$

- $M, x, s \vDash [\phi]S[\psi]$ se e solo se

$$\exists A \subseteq X.x \in A \wedge \forall y \in A.M, y, s \vDash \phi \wedge \forall z \in \mathcal{B}^+(A).M, z, s \vDash \psi$$

- $M, x, s \vDash A\phi$ se e solo se $\forall \sigma \in \mathcal{P}_s.M, x, \sigma \vDash$
- $M, x, s \vDash E\phi$ se e solo se $\exists \sigma \in \mathcal{P}_s.M, x, \sigma \vDash$
- $M, x, \sigma \vDash X\phi$ se e solo se $M, x, \sigma(1) \vDash \phi$
- $M, x, \sigma \vDash [\phi]U[\psi]$ se e solo se $\exists n.M, x, \sigma(n) \vDash \psi$ e $\forall n' < n.M, x, \sigma(n') \vDash \phi$
- $M, x, \sigma \vDash F\phi$ se e solo se $\exists n.M, x, \sigma(n) \vDash \psi$

Si può mostrare in particolare che con questa semantica le formule $\bot$, p, $\Phi \wedge \Phi$, $\neg\Phi$, $\mathcal{N}\Phi$, $[\Phi]\mathcal{S}[\Phi]$, $EX\Phi$, $AF\Phi$ ed $E[\Phi]U[\Phi]$ sono una base della logica introdotta.

Altri operatori che possono essere definiti sono

- $AG\phi ::= \neg(E[T]U[\neg\phi])$
- $EG\phi ::= \neg AF(\neg\phi)$

2 Il model checker

Definiamo un modello finito se X e S sono finiti. Dato un modello M finito il model checker introdotto permette di testare formule della logica STL su M. In particolare la valutazione di una formula ϕ restituisce l'insieme delle coppie (x, s) tali che $M, x, s \vDash \phi$.

Nell'implementazione attuale si assume che lo spazio sia una griglia $m \times n$ ottenuta caricando un'immagine od una serie di immagini. Il model checker mostra in maniera interattiva l'immagine relativa ad un certo istante di tempo deciso dall'utente. Valutando una formula i punti che la verificano vengono colorati.

Durante l'esecuzione possono essere fissati un punto dello spazio s_0, un istante di tempo t_0 ed una formula f_0, in modo da poter controllare la validita della formula sui singoli punti oltre che visivamente sull'immagine.

Diamo ora i comandi del model checker:

show store; Mostra le formule attualmente salvate in memoria.

show status; Restituisce posizione, tempo e formula zero. Inoltre restituisce la valutazione della formula zero nella posizione e nel tempo zero.

show future; Mostra i possibili tempi futuri e la valutazione della formula zero nella posizione zero.

show space; Mostra la posizione zero. Sulla mappa, colora un pixel corrispondente alla posizione zero.

show time; Mostra il tempo zero.

show formula; Mostra la formula zero.

set space ⟨int⟩ ⟨int⟩; Imposta la posizione zero.

set time ⟨int⟩; Imposta il tempo zero e carica l'immagine corrispondente.

let ⟨ide⟩ = ⟨fsyntax⟩; Definisce una nuova formula e la salva nello store.

sem ⟨color⟩ ⟨ide⟩;

sem ⟨color⟩ ⟨ide⟩ ⟨fsyntax⟩ ... ⟨fsyntax⟩;

sem ⟨color⟩ ⟨fsyntax⟩; Colora i punti in cui la formula richiamata risulta vera (per richiamare formule con parametri serve definirle precedentemente col comando let).

save store; Salva lo store attuale nel file formula.fr (cancella eventuali dati precedentemente salvati).

load store; Carica lo store salvato in formula.fr

save image ⟨filename⟩; Salva lo stato attuale del sistema. [save image prova;] salva una bitmap per ogni punto del tempo (ad esempio, al punto 2 corrisponde il file prova2.bmp)

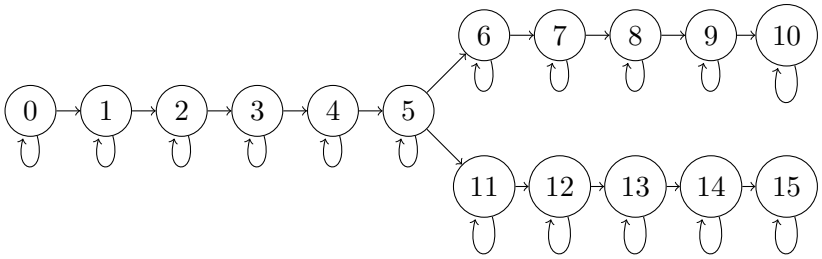
reset; Riporta il sistema allo stato di avvio.

refresh; Ricarica l'immagine.

exit; Ferma il programma.

3 Un esempio

Forniamo ora un esempio di utilizzo del model checker. Nel nostro esempio il Kripke frame è il seguente



cioè ci sono due possibili evoluzioni per il futuro all'istante 5.
La sequenza di immagini caricate è

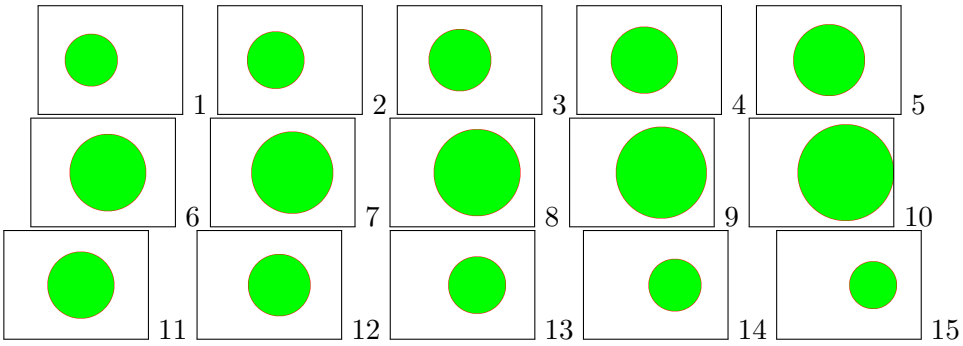


Figura 1: Figure 1

Il centro del cerchio nelle figure si sta muovendo a velocità costante verso destra. Il suo raggio cresce a velocità costante v dalla figura 1 alla figura 5. Nel primo futuro (figure da 6 a 10) il raggio continua a crescere, mentre nel secondo futuro (figure da 11 a 15) il raggio diminuisce a velocità costante v.

Le proposizioni atomiche sono g ed r che valutate restituiscono rispettivamente i punti verdi ed i putni rossi nelle figure.

La valutazione della formula $[g]S[r]$ è restituita dal comando

```
sem blue S[<g>,<r>]
```

e colora la regione verde avente bordo rosso (**??**).

Il comando

```
sem blue S[!(<g> or <r>),<r>]
```

colora invece la regione bianca (non verde e non rossa) avente bordo rosso (**??**).

Il comando

```
sem blue AG <g>
```

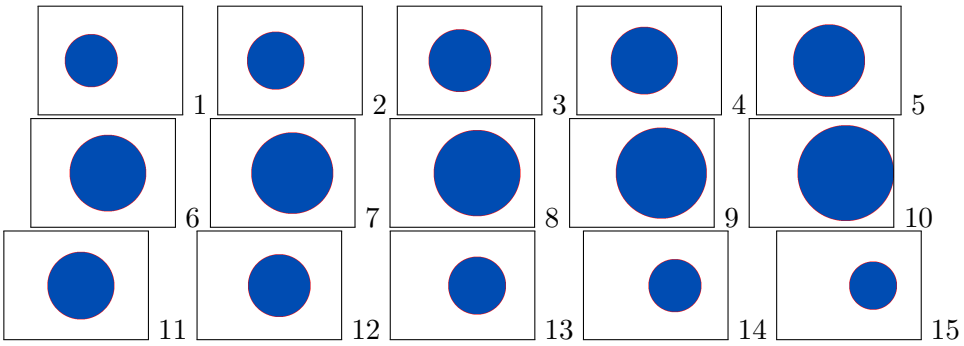


Figura 2: Figure 2

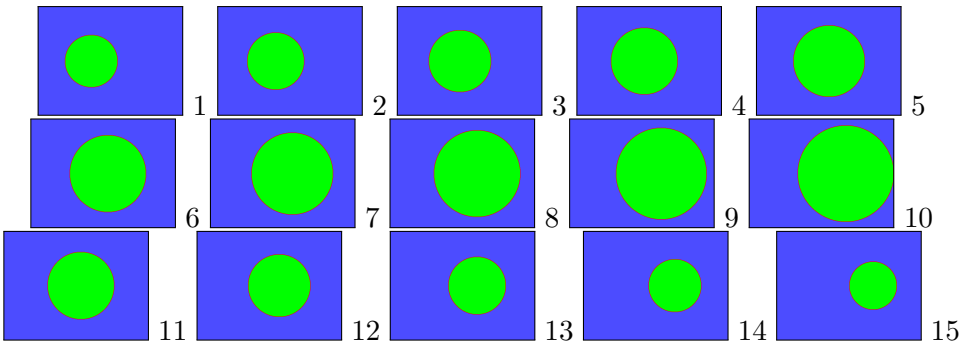


Figura 3: Figure 3

restituisce i punti che per *qualsiasi futuro* saranno verdi in ogni istante.

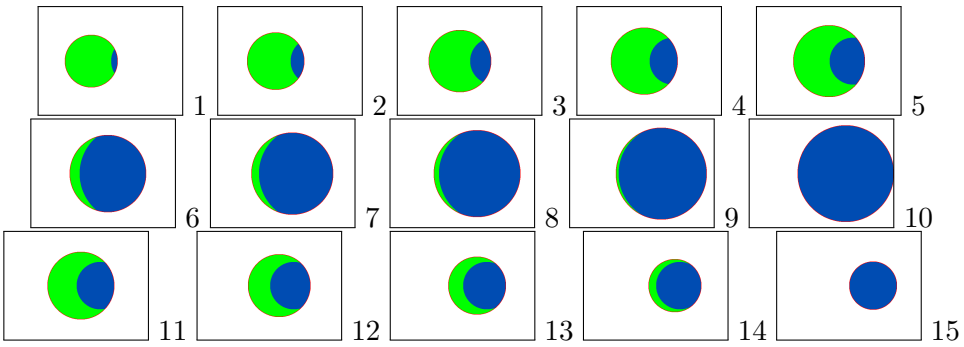


Figura 4: Figure 4

Notiamo che la valutazione sulle figure dalla 1 alla 5 risulta essere l'intersezione di due cerchi (quello della figura considerata e quello in figura 15). Nelle figure da 11 a 15 la situazione è la stessa in quanto l'istante 15 è ancora un possibile istante futuro. Invece nelle figure da 6 a 10 l'istante 15 non è più raggiungibile e quindi l'area che rimarrà definitivamente verde si amplia.

Riferimenti bibliografici

[1] Vincenzo Ciancia, Diego Latella, Michele Loreti, Mieke Massink *Specifying and Verifying Properties of Space*. 16 Maggio 2014