

Agente: termine di logica del primo ordine che utilizza operatori

$$E \xrightarrow{a} E' \} \text{evento}$$

Agente E evolve tramite azione "a" in agente E'

SOS Rules

$$\frac{\text{transition } A}{\text{transition } B}$$

se trans A è possibile,
possiamo dedurre trans B

CCS:

Calculus of communicating systems

Act

Insieme di tutte le possibili azioni

$a, \bar{a} \in \text{Act}$ azione e coazione

$\{\tau\} \cup \text{Act}$, τ azione invisibile

Sintassi

$$E ::= a.E \mid E + E \mid E \parallel E \mid E \setminus R \mid E[f] \mid (E) \mid 0$$

Semantica

$a.E$ action prefixing/concatenazione

L'agente $a.E$ (con $a \in \text{Act}$) prima esegue l'azione "a" e poi si comporta come un agente E

L'assioma corrispondente è:

$$\overline{a.E \xrightarrow{a} E}$$

$E + F$

Choice/scelta

l'esecuzione di $E+F$ avviene scegliendo (in maniera non predeterminata) E o F e poi Comportandosi come l'agente scelto

$$\frac{E \xrightarrow{a} E'}{E + F \xrightarrow{a} E'}$$

$$\frac{F \xrightarrow{a} F'}{E + F \xrightarrow{a} F'}$$

E, R Restriction/restrizione

$$\frac{E \xrightarrow{\alpha} E', \quad \alpha, \bar{\alpha} \notin R}{E \setminus R \xrightarrow{\alpha} E' \setminus R}$$

$E[\gamma]$ Relabeling

$$\frac{E \xrightarrow{\alpha} E'}{E[\gamma] \xrightarrow{\gamma(\alpha)} E'[\gamma]}$$

$E \parallel E$

Concurrent Composition /composizione concorrente

L'esecuzione di $E \parallel E$ avviene scegliendo (sempre non deterministicamente) una tra 3 possibilità:

$$\frac{E \xrightarrow{\alpha} E'}{E \parallel F \xrightarrow{\alpha} E' \parallel F}$$

E esegue α , F non cambia

$$\frac{F \xrightarrow{\alpha} F'}{E \parallel F \xrightarrow{\alpha} E \parallel F'}$$

F esegue α , E non cambia

$$\frac{E \xrightarrow{\alpha} E', \quad F \xrightarrow{\bar{\alpha}} F'}{E \parallel F \xrightarrow{\tau} E' \parallel F'}$$

E esegue α e F la coazione $\bar{\alpha}$
 $E \parallel F$ si incrinizzano su questa azione e la loro azione viene chiamata tau

CSP

Calculus of Sequential Process

Sintassi

$$E ::= Nil \mid a.E \mid E+E \mid E \parallel E \mid E/R \mid E[f]$$

Nil, "stesso di 0"

nessun assioma è legato a lui, non può compiere alcuna azione

$a.E$

$$\frac{}{a.E \xrightarrow{a} E}$$

$E+E$

$$\frac{E \xrightarrow{a} E'}{E+F \xrightarrow{a} E'+F}$$

$$\frac{F \xrightarrow{a} F'}{E+F \xrightarrow{a} F'}$$

$E \parallel E$

$$\frac{E \xrightarrow{a} E', a \notin S}{E \parallel_S F \xrightarrow{a} E' \parallel F}$$

$$\frac{F \xrightarrow{a} F', a \notin S}{E \parallel_S F \xrightarrow{a} E \parallel F'}$$

$$\frac{E \xrightarrow{a} E', F \xrightarrow{a} F', a \in S}{E \parallel_S F \xrightarrow{a} E' \parallel F'}$$

PN system

graficamente rappresentabile aggiungendo una etichettatura alla rappresentazione della PN

Basato sul concetto di Marking (marking è MPTPN)

$$M \in \mathbb{N}^{|P|} \text{ (vettor)} \quad \text{Assegna ad ogni posto } P \text{ un valore } \mathbb{N}^+$$

P/T Petri system

$$S = \langle N, m_0 \rangle \quad m_0 \text{ marcatura iniziale}$$

Evoluzione di un P/T Petri system.

1 Enabling.

$$t \text{ enabled at } M \text{ iff } M \geq \text{Pre}[P, t]$$

una transizione t è abilitata da (at) M se tutti il n° di token in input \geq del peso degli archi che collegano input-transizione

2 Firing

Se t abilitata da M , t può scattare

$$M \xrightarrow{t} M' \quad M' = M + C[P, t]$$

P grande, input
 $C[P, t]$ è vettore
colonne di C

lo scatto toglie tanti token dai posti collegati in input a t quanto è il peso degli archi che li collega e li aggiunge allo stesso modo ai posti in output di T

3 Firing sequence (da M)

$$\sigma = t_1, \dots, t_k \in T \quad \text{t.c.} \quad M \xrightarrow{t_1} M_1 \dots \xrightarrow{t_k} M_k$$

scriviamo $M \xrightarrow{\sigma} M_k$

L_2 proof scump: $\sigma = [t_1, \dots, t_k] \in T$.

Scump: $M \xrightarrow{\sigma} M'$ se $\exists \{M_0, \dots, M_k\} : \forall i \in [1, \dots, k] \quad M_{i-1} \xrightarrow{t_i} M_i$

4. bis Firing vptm

$\bar{\sigma}$ vettore costituito dalla sequenza σ
sopra sceltibile in m

$$m' = m + C \cdot \bar{\sigma} \quad \text{state equation}$$

5. Linguaggio

$L(s)$ è l'insieme delle firing sequenze sceltibili
da m_0

Basandosi sull'evoluzione definiamo 2 strumenti (RS e RG).

RS Reachability set.

$RS(s)$ è l'insieme di tutti i markings raggiungibili
da m_0 tramite una sequenza $\in L(s)$

RG Reachability Graph.

$RG(s)$ rapp. grafico di $RS(s)$

• $V = RS(s)$ L'insieme dei nodi = Reachability Set

• $(v_1, v_2) = (m_1, m_2) \in E \iff \exists t \in T \text{ t.c. } m_1 \xrightarrow{t} m_2$

c'è un arco tra due nodi se esiste una transizione tra le due
marcature corrispondenti

Colored P/T nets

$$N = \langle P, T, P \times P, P \times T, C, cd \rangle$$

C : color classes (classi di colori)

notare che i colori sono classi
quindi C è un set di set

$cd: P \cup T \rightarrow C$ definire il colore di ogni posto e trans.

Analisi

Boundedness

$$b(p) = \sup \{ m[p] \mid m \in RS(N, m_0) \}$$

$b(p)$ "bound" di p

posto p è bound in $\langle N, m_0 \rangle$ s.p. $b(p) < \infty$

$\langle N, m_0 \rangle$ è bounded s.p. $b(p) < \infty \quad \forall p \in P$ di N

$$S = \langle N, m_0 \rangle \text{ bounded} \iff RS(S) \text{ finito}$$

N è structuralmente bound $\iff \forall m \langle N, m \rangle$ è bound

Deadlock

$$\langle N, m_0 \rangle \text{ è deadlock-free} \iff \forall m \in RS(S) \exists t \in T$$

t.c. t è abilitata in m ($m \geq P_{up}(P, t)$)

Liveness

$m \in t \rightarrow$

$$t \text{ è live in } S \iff \forall m \in RS(S) \exists \sigma \text{ t.c. } m \xrightarrow{\sigma} m' \wedge m[t]$$

Cioè t è live in S se da ogni marcatura si può sempre tornare ad una marcatura in cui t è abilitata. Ergo t può scattare infinitamente spesso.

$$S = \langle N, m_0 \rangle \text{ è live} \iff \forall t \in T \quad t \text{ è live in } S$$

N è structuralmente live s.p. $\exists m$ t.c. $\langle N, m \rangle$ è live

Reversibility

$$\text{Il marking } m \text{ è un home state} \iff \forall m' \in RS(S) \exists \sigma \text{ t.c. } m' \xrightarrow{\sigma} m$$

$$S = \langle N, m_0 \rangle \text{ è reversibile} \iff \forall m \in RS(S) \exists \sigma \text{ t.c. } m \xrightarrow{\sigma} m_0$$

(cioè m_0 è home state di N)

Analisi performativa

Precedability graph e Coverability graph.
Basati sul concetto di copertura.

$m' > m \iff \forall p \in P \quad m'(p) > m(p)$
 $m' \text{ copre } m$ in ogni posto di N ci sono + token

Algoritmo RG.

- 1 $RG = \{m_0\}$ inizio con solo la marcatura iniziale
- 2 scelgo un nodo (m) non taggato (non visitato) e lo taggo
- 3 $\forall t \text{ t.c. } m[t > :$
 - calcolo m' t.c. $m \xrightarrow{t} m'$
 - se esiste m'' t.c. $m' \leq m'' \wedge m'' > m$
(l'algoritmo finisce. upper bound).
 - se m' non è presente in RG lo aggiungo e lo aggiungo.
 - aggiungo \hookrightarrow scelgo t_2 in $p \in m'$ e lo chiamo t

Algoritmo Coverability Graph.

- 1 $RG = \{m_0\}$ inizio con solo la marcatura iniziale
- 2 scelgo un nodo (m) non taggato (non visitato) e lo taggo
- 3 $\forall t \text{ t.c. } m[t > :$
 - calcolo m' t.c. $m \xrightarrow{t} m'$
 - se esiste m'' t.c. $m' \leq m'' \wedge m'' > m$
invece m come ω e non lo visito più
 - se m' non è presente in RG lo aggiungo e lo aggiungo.
 - aggiungo \hookrightarrow scelgo t_2 in $p \in m'$ e lo chiamo t

Liveness & Marking invariant (di proprietà Π)

\downarrow
 $\forall m \in RS(S), \exists m' \in RS(\langle N, m \rangle)$ t.c. m' soddisfa Π

per ogni marcatura in RS esiste una marcatura raggiungibile che soddisfa PI

Marking

$\forall m \in RS(S), m$ soddisfa Π

PI è soddisfatto da tutte le marcature in RS

Algorithm Marking invariant.

1 tutti gli elementi $RS(S)$ sono visitati.

2 \forall elemento visitato m

— se m non rispetta Π

RETURN FALSE

3 RETURN TRUE.

Algorithm Liveness invariant.

1 Scomporre $RG(S)$ nelle strongly connected components

2 Crea un nuovo grafo dove i nodi sono le SCC
e gli archi gli archi che completano le SCC

3 Calcola l'insieme F delle SCC terminali

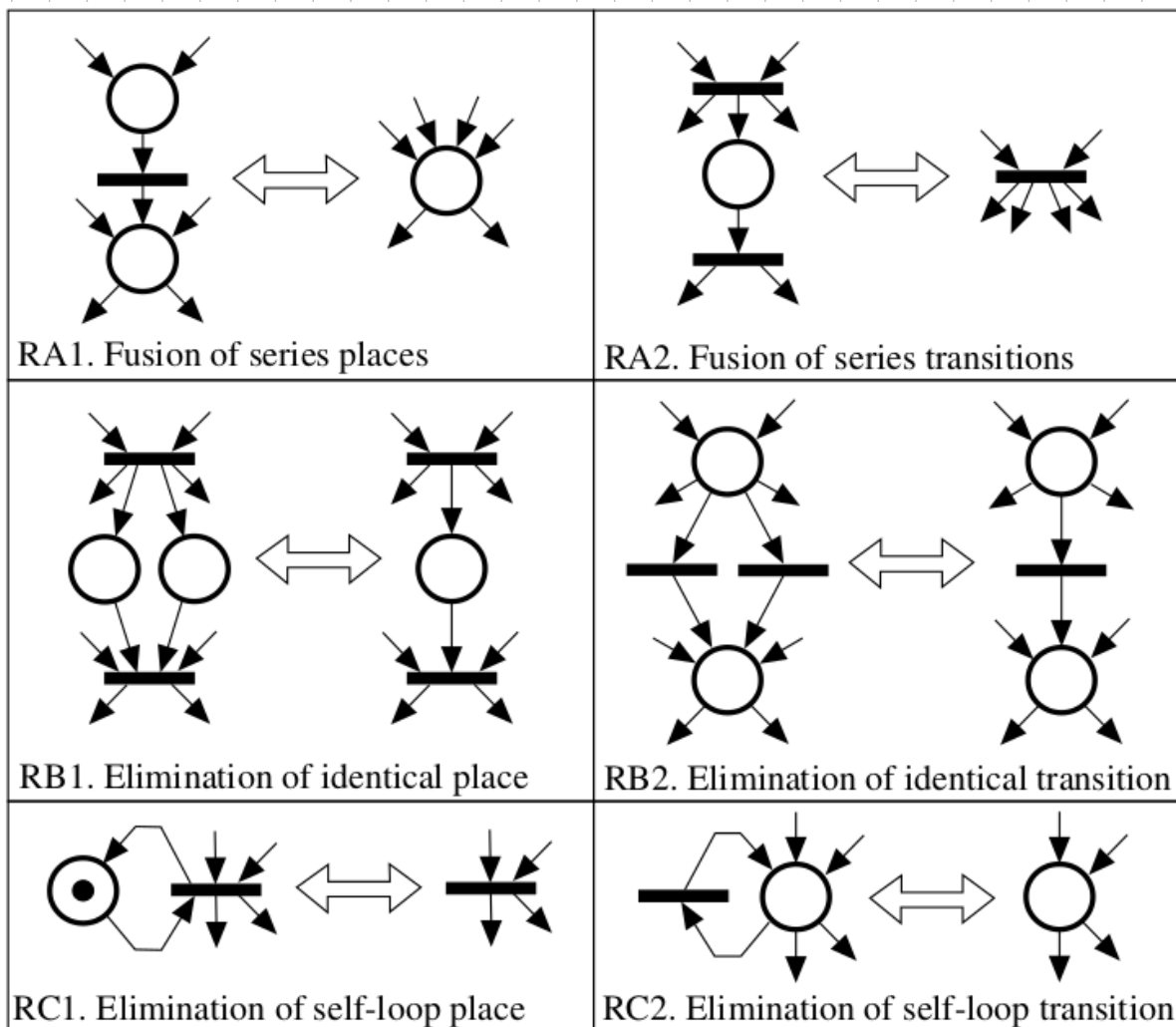
4 For all C in F

— se C non contiene una marcatura m che soddisfa Π

RETURN FALSE

5 RETURN TRUE

Analisi di riduzione



Analisi strutturale (Linear algebraic techniques)

L'idea è quella di usare l'eq di stato

$$m = m_0 + C \cdot \sigma$$

per la verifica di proprietà che dipendono solo da N
(K-boundedness, deadlock, ...)

Esprimendo le proprietà all'interno dell'eq di stato per poi risolverle con l'eq lineare.

Ad esempio se voglio mostrare che i posti P1 e P2 sono in mutua esclusione mi basta verificare che il seguente sistema non ha soluzione:

$$\begin{cases} m = m_0 + C \cdot \sigma \\ m[p_1] > 0 \\ m[p_2] > 0 \end{cases}$$

p-flow: vettore y t.c. $y \cdot C = 0$

t-flow: vettore y t.c. $C \cdot y = 0$

p/t-semiflow: flows con valori naturali positivi

I semiflow sono utilizzati per generare invarianti lineari, abbiamo 2 classi di invarianti

p-semiflow \rightarrow token conservation law

$$y \in \mathbb{N}^n, y \cdot C = 0 \Rightarrow \forall m_0, \forall m \in RS(N, m_0) \quad y \cdot m = y \cdot m_0$$

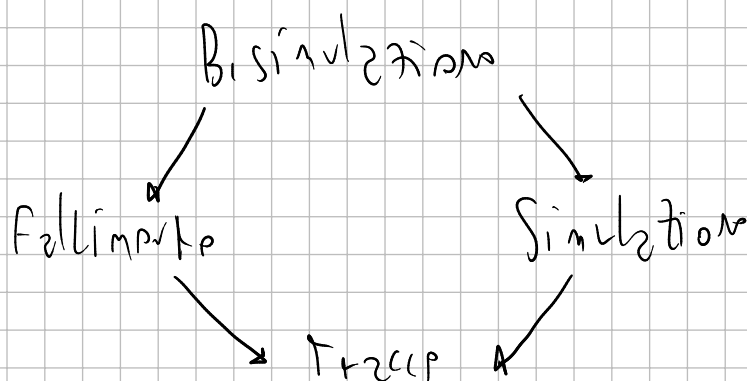
The weighted sum of tokens is constant for all reachable markings

t-semiflow \rightarrow cyclic behavior law

$$y \in \mathbb{N}^n, C \cdot y = 0 \Rightarrow \exists m_0, \exists \sigma \in L(N, m_0) \text{ t.c. } m_0 \xrightarrow{\sigma} m_0 \wedge \sigma = y$$

Esiste una firing sequence che da un initial marking riporta allo stesso, tale firing sequence è y

Analisi equivalente NP.



se $P \approx Q$ in
simulation $\Rightarrow P \approx Q$
in trace.
(valido verso \downarrow)

Trace: dato $T(E)$ (tracce di E) l'insieme delle sequenze finite che possono essere prodotte dall'evoluzione di E

$$E \approx_{tr} F \iff T(E) = T(F)$$

Falimento

Un falimento per l'evento E è la coppia (σ, A) con $\sigma \in T(E)$, $A \subseteq Act$ per cui

$E \xrightarrow{\sigma} F$ e l'azione σ in A può essere eseguita da F

E attraverso σ evolve in F

dato $Fail(E)$ l'insieme dei falimenti di E

$$E \approx_{FI} F \iff Fail(E) = Fail(F)$$

Simulation:

R relazione di simulation. $R \subseteq S \times S$

$$- E R F$$

$$- E \xrightarrow{\sigma} E', \exists F' \text{ t.c. } F \xrightarrow{\sigma} F'$$

$$- E' R F'$$

Cioè ogni azione di E può essere ripetuta da F andando a finire in un nuovo stato che mantiene la relazione di simulazione

dati E ed F eguali diremo che F simula E ($E \approx_{sim} F$)

se esistono due relazioni di simulazione R e Q t.c.

$$E R F \wedge F Q E$$

Bisimulation:

Una simulation in cui le due relazioni Q ed R sono uguali ($Q = R$)

weak bisimulation, Bisimulo senza considerare azioni τ

$$m \xrightarrow{\tau} m' \text{ se } m \xrightarrow{\tau} \dots \xrightarrow{\tau} \dots \xrightarrow{\tau} m'$$

LTL

simboli.

Kathorn $\phi := p \mid \neg \phi \mid \phi \vee \psi \mid X \phi \mid \phi U \psi$

Proprieta aggiunte

$\phi \wedge \psi \mid F \phi \mid G \phi$

non impararli, sappi solo
si possono derivare

$\neg(\neg \phi \wedge \neg \psi) \mid \neg \neg \phi \mid \neg F \neg \phi$

Semantica

Kathorn

$M = (S, R, L)$

basato sul "modello"

S : insieme di stati $\neq \{\emptyset\}$

$R: R(s) \subseteq S$ ad ogni stato associa un successore unico

$L: label \subseteq 2^AP$ ad ogni stato associa le prop atomiche valide in S

Dato una formula ϕ diremo $(M, s) \models \phi$ se ϕ è

una formula valida nello stato s del modello M .

Formalmente

$M, s \models p \iff p \in L(s)$

cioè M è il linguaggio del modello
 P_N è l'insieme delle
proposizioni soddisfatte

$M, s \models \neg \phi \iff \neg (M, s \models \phi)$

$M, s \models \phi \vee \psi \iff (M, s \models \phi \vee M, s \models \psi)$

$M, s \models X \phi \iff M, R(s) \models \phi$

modello M

$s \models \phi U \psi \iff \exists j \geq 0$ t.e. $R^j(s) \models \psi$

$\forall k \ 0 \leq k \leq j \ R^k(s) \models \phi$

future e globally si derivano ma comunque sono:

$$S \models F \phi \iff \exists j, j \geq 0 \text{ s.t. } R^j(S) \models \phi$$
$$S \models G \phi \iff \forall k, k \geq 0 \quad R^k(S) \models \phi$$

prob: ✓ basate su "path"

Descriviamo il sistema in termini di esecuzioni lineari infinite (con stuttering nel caso sia finita)
Si definisce la regola sulla singola possibile esecuzione e si ripete la verifica per ogni possibile esecuzione.

σ path s_0, s_1, s_2, \dots $s_i \in S$

σ' suffisso di σ s_i, s_{i+1}, \dots $\sigma^0 = \sigma$

$\sigma' \models p \iff s_i \models p$
anzichè non uguali a $K \models \phi$, con σ' al posto di (M, s)

$\sigma' \models X \phi \iff \sigma^{i+1} \models \phi$

$\sigma' \models \phi \vee \psi \iff \exists j, j \geq i \quad \sigma^j \models \phi \wedge \forall k, i \leq k \leq j \quad \sigma^k \models \phi$
gli altri si deducano.

Costanti di fissaggio:

$\forall n \text{ condition} \quad G F \psi$
prima o poi accadrà theta

$\text{wp } \psi \quad F G \phi \implies G F \psi$
se da un certo punto in poi phi diventa vera, accadrà theta

$\text{strong } \quad G F \phi \implies G F \psi$
se accade phi, accadrà theta

Facciamo model Checking costruendo il corrispondente Automa di Buchi (del modello) e costruendo l'automa di Buchi della formula negata.
L'intersezione tra i due automi fornisce tutti i controesempi (le esecuzioni che non soddisfano la formula. se l'intersezione è vuota significa che la formula è vera.

CTL

$$\varphi ::= p \mid \neg \varphi \mid \varphi \vee \psi \mid EX \varphi \mid E[\varphi \vee \psi] \mid A[\varphi \vee \psi]$$

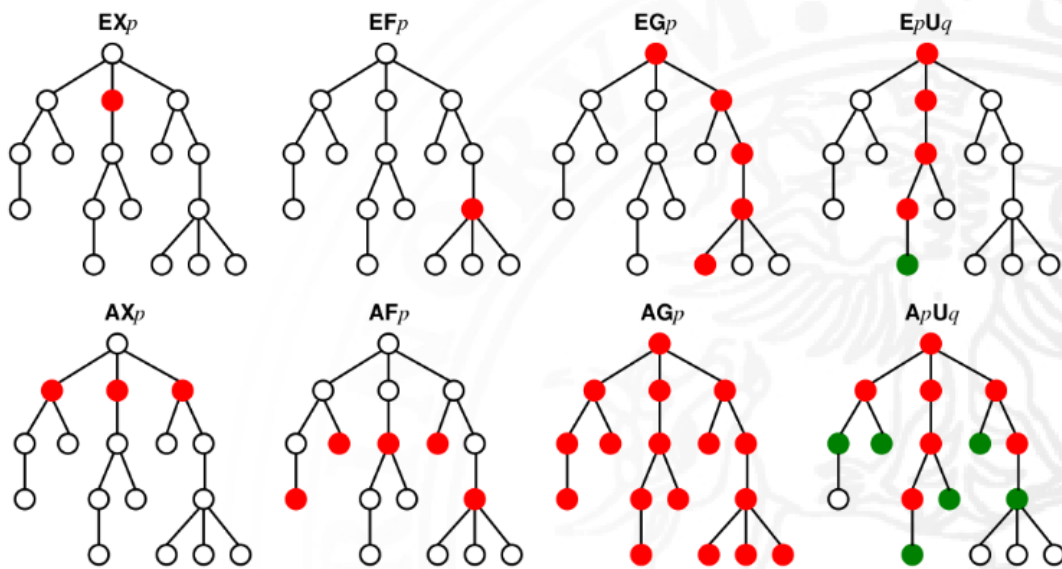
Futuro e globale, esistono sicuramente

$$EF \varphi = E[\text{true} \vee \varphi] \quad AF \varphi = A[\text{true} \vee \varphi]$$

$$EG \varphi = \neg AF \neg \varphi$$

$$AG = \neg EF \neg \varphi$$

$$AX = \neg EX \neg \varphi$$



Semantica:

si tratta in base sul modello (S, R, L)

S e L uguali. R diverso

$R: S \rightarrow 2^S$ ad ogni stato assegno l'insieme dei suoi successori

oppure $R \subseteq S \times S$

Quindi nel modello un path $\sigma = s_1, s_2, \dots$ con $(s_i, s_{i+1}) \in R$
 come in LTL $\sigma[i]$ è i -esimo elemento della sequenza.

Definizione

$$P_M(s) = \{ \sigma \in S^\omega \mid \sigma[0] = s \}$$

Cioè l'insieme di tutte le path che partono da s .

M, s , per brevità assumiamo M

$$s \models p \quad s, p \in L(s) \quad \text{dato } p\text{-stato}$$

$$s \models \neg p \quad s, \neg(s \models p) \quad \text{dato } p\text{-path}$$

$$s \models \varphi \vee \psi \quad s, s \models \varphi \vee s \models \psi$$

$$s \models \exists x \varphi \quad s, \exists \sigma \in P_M(s) \text{ t.c. } \sigma[1] \models \varphi$$

$$s \models \exists [\varphi \vee \psi] \quad s, \exists \sigma \in P_M(s) \text{ t.c.}$$

$$\exists j \geq 0 \quad \sigma[j] \models \varphi \wedge \forall k \quad 0 \leq k \leq j \quad \sigma[k] \models \varphi$$

$$s \models A[\varphi \vee \psi] \quad s, \forall \sigma \in P_M(s)$$

$$\exists j \geq 0 \quad \sigma[j] \models \varphi \wedge \forall k \quad 0 \leq k \leq j \quad \sigma[k] \models \varphi$$

Fairness.

weak e strong fairness non sono esprimibili in CTL

si esprime il modello di Kripke

$$M = (S, R, L, F)$$

$F \subseteq 2^S$ insieme di fairness-constraint

insieme di insiemi

$$F = \{ F_1, F_2 \} \quad \text{con } F_1 = \{ s_1, s_2, \dots \} \in S$$

un cammino sigma è F-fair se passa infinitamente spesso per almeno uno stato di ognuno degli insiemi F_i in F

Il model checking di CTL è un algoritmo che risolve il seguente problema:
dato modello M e formula ψ , quali sono gli stati s per cui (M,s) soddisfa ψ ?

L'algoritmo è un semplice algoritmo iterativo che parte da una sottoformula di ψ di lunghezza 1, vede in quali stati è valida, induttivamente prende la formula di lunghezza 2 e via così fino a riottenere ψ

Questo può essere fatto grazie alla definizione ricorsiva delle sottoformule in CTL

$$\begin{aligned} \text{Sub}(p) &= \{p\} \\ \text{Sub}(\neg \phi) &= \text{Sub}(\phi) \cup \{\neg \phi\} \\ \text{Sub}(\phi \vee \psi) &= \text{Sub}(\phi) \cup \text{Sub}(\psi) \cup \{\phi \vee \psi\} \\ \text{Sub}(\text{EX } \phi) &= \text{Sub}(\phi) \cup \{\text{EX } \phi\} \\ \text{Sub}(\text{E}[\phi \text{ U } \psi]) &= \text{Sub}(\phi) \cup \text{Sub}(\psi) \cup \{\text{E}[\phi \text{ U } \psi]\} \\ \text{Sub}(\text{A}[\phi \text{ U } \psi]) &= \text{Sub}(\phi) \cup \text{Sub}(\psi) \cup \{\text{A}[\phi \text{ U } \psi]\}. \end{aligned}$$

CTL*

$$\begin{aligned} \text{State formulas } \phi &::= a \mid \neg \phi \mid \phi \wedge \phi \mid \text{E } \phi \mid \text{A } \phi \\ \text{path formulas } \psi &::= \phi \mid \psi \wedge \psi \mid \text{X } \psi \mid \psi \text{ U } \psi \end{aligned}$$

CTL* deriva dall'interpretazione di LTL sul modello di CTL (definire la semantica in termini di Branching model)

In tal maniera possiamo quindi suddividere le formule in "formule di stato" e "formule di path".
LTL non presenta formule di path mentre CTL permette l'uso di formule di stato solo dopo X o U.

CTL formulas:

state formulas uguali a CTL*

$$\text{path formulas: } \psi = \neg \psi \mid \text{X } \phi \mid \phi \text{ U } \phi$$

in CTL* posso inserire path formulas di path formulas

Time Automata

Def. clock $v \in \mathbb{R}^+$ — tempo continuo

Def. clock constraint

Dato un clock $x \in C$, spt di clocks, e un valore $c \in \mathbb{N}$

$$z ::= x < c \mid x \leq c \mid \neg a \mid z \wedge z$$

scriviamo $\psi(c)$ oppure $\text{Cstr}(c)$

Def. Time automata

$$A = (L, l_0, E, \text{label}, C, \text{clocks}, \text{inv}, \text{guards})$$

L : spt di locazioni $\neq \{\emptyset\}$

l_0 : locazione iniziale $l_0 \in L$

E : archi $E \subseteq L \times L$

$\text{label}: L \rightarrow 2^{\text{AP}}$ il solito

C : spt di clocks

$\text{clocks}: E \rightarrow 2^C$ funzione di clock aspt. ad ogni arco può associare i clock $\in C$ e questi vengono espressi.

$\text{guards}: E \rightarrow \psi(C)$ ad ogni arco associa un clock constraint se violato l'arco non può essere percorso (non vi è alcun obbligo di percorrenza)

Invariant $\text{inv}: L \rightarrow \psi(C)$ ad ogni locazione associa un clock constraint (può sia vietare l'accesso che obbligare l'uscita)
Clock evaluation

Def. clock valuation: $v: C \rightarrow \mathbb{R}^+$

assegna ad ogni clock il suo valore (in quel momento)

uno stato di un Time automata A è (l, v) con $l \in L$ e $v = V(c)$

Def. clock evaluation: per $x \in C$, $v \in V(C)$, $c \in \mathbb{N}$ e $\alpha, \beta \in \text{str}(c)$

$$v \models x \leq c \quad \text{sp} \quad v(x) \leq c$$

$$v \models \neg z \quad \text{sp} \quad v(w) \neq z$$

e gli altri li deduci.

in pratica definisce quando il clock soddisfa il constraint

TTS time transition system.

Il TTS di A, scritto $M(A)$ è così definito:

$$M(A) = (S, s_0, \rightarrow)$$

$$- S = \{ (l, v) \in L, V(C) \mid v \models \text{inv}(l) \}$$

tutti gli stati la cui valutazione soddisfa gli invarianti (cioè tutti gli stati permessi)

$$- s_0 = (l_0, v_0) \quad \text{con } v_0(c) = 0 \quad \forall c \in C$$

lo stato iniziale, con tutti i clock a zero

- le transizioni \rightarrow definite da 2 regole

$$1 \text{ transizione discreta } \langle l, h \rangle \xrightarrow{t} \langle l', h' \rangle \text{ SP}$$

$$- p = (l', l) \in E \quad (\text{esiste } l' \text{ trans})$$

$$- v \models \text{guard}(p)$$

$$- h' = \text{reset clocks}(p) \text{ in } v' \quad h' \models \text{inv}(l')$$

$$2 \text{ transizione "delay" } \langle l, h \rangle \xrightarrow{d} \langle l, h + d \rangle \text{ SP}$$

$$d \in \mathbb{R}^+ \quad \forall d' \leq d, h + d' \models \text{inv}(l) \quad \text{let time pass}$$

Come distinguo un deadlock da un let time pass?

Ehpspe time.

dato un path $s_0 \xrightarrow{z_0} s_1 \xrightarrow{z_1} \dots$ e $i \in \mathbb{N}$ l'ehpspe time $\Delta(\sigma, i)$

$$\Delta(\sigma, 0) = 0$$

$$\Delta(\sigma, i+1) = \Delta(\sigma, i) = \begin{cases} 0 & \text{SP } z_i \in \text{transition}(\ast) \\ z_i & \text{SP } z_i \in \mathbb{R} \end{cases}$$

le transizioni sono istantanee

$$\text{SP } \lim_{i \rightarrow \infty} \Delta(\sigma, i) = \infty \quad \text{il camino è tempo divergente.}$$

Un path è detto zeno se non è tempo divergente (cioè tempo convergente), un automa è detto zeno se da ogni stato può partire un cammino zeno. (cioè non va mai in timelock (deadlock))

TL

$$\varphi = p \mid \alpha \mid \neg \varphi \mid \varphi \vee \psi \mid \exists x \varphi \mid E[\varphi \vee \psi] \mid A[\varphi \vee \psi] \mid z \text{ in } \varphi$$

Esiste uguale a CTL a parte α e $z \text{ in } \varphi$

$\alpha \in \Psi(C \cup D)$ Formula clock constraint.
 clocks di formula
 clock substitution

$z \text{ in } \varphi$ free variable identifier:

$(z \text{ in } \varphi)$ vale in S se φ vale in S , stato in cui z clock

z non è free variable.

Esempio

$$E[\phi \vee_{\leq 7} \varphi]$$

ϕ vale e prima z unità di tempo φ diventa
valere.