$E \xrightarrow{a} E'$ evento

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

SOS Rules

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

se trans A è possibile, possiamo dedurre transB

CCS: Calculus of comunicating systems

Act insieme di tutte le possibili azioni

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

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

sintassi

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

Spiegazione

$a.E$ action prefixing/concatenazione

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

L'assioma corrispondente è: $\dfrac{}{a.E \xrightarrow{a} E}$

$E+E$ Choice/scelta

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

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

$E \setminus R$ Restriction/restrizione

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

$E[f]$ Relabeling

$$\frac{E \xrightarrow{a} E'}{E[f] \xrightarrow{f(a)} E'[f]}$$

$E \| E$

Concurrent Composition /composizione concorrente

L'esecuzione di E||E avviane scegliendo (sempre non deterministicamente) una tra 3 possibilità:

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

E esegue a, F non cambia

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

F esegue a, E non cambia

$$\frac{E \xrightarrow{a} E', \quad F \xrightarrow{\bar{a}} F'}{E \| F \xrightarrow{\tau} E' \| F'}$$

E esegue a e F la coazione !a E||F si incronizzano su questa azione e la loro azione viene chiamata tau

CSP

Calculus of Sequential Process

Sintassi

$$E ::= Nil \mid a.E \mid E+E \mid E \| E \mid E/R$$

Nil: "stop"

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

$a.E$

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

$E+E$

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

$E \| E$

$$\frac{E \xrightarrow{a} E', \ a \notin S}{E \|_S F \xrightarrow{a} E' \| F} \quad // \quad \frac{F \xrightarrow{a} F', \ a \notin S}{E \|_S F \xrightarrow{a} E \| F'}$$

$$\frac{E \xrightarrow{a} E', \ F \xrightarrow{a} F', \ a \in S}{E \|_S F \xrightarrow{a} E' \| F'}$$

PN

Graficamente rappresentabile come grafo diretto, bipartito, pesato

Una rete di Petri è una quadrupla

$$N = (P, T, F, W)$$

P, T insiemi finiti t.c. $P \cap T = \emptyset$

F flow: $F \subseteq (P \times T) \cup (T \times P)$

W weight: $w : F \longrightarrow \mathbb{N}^+$

In Lapp inizialmente invece

$$N = \langle P, T, Pre, Post \rangle$$

P, T insiemi finiti t.c. $P \cap T = \emptyset$

$Pre, Post \in \mathbb{N}^{|P| \times |T|}$ oppure 2 funzioni:

- $Pre : P \times T \to \mathbb{N}$

$$pre(p,t) = \begin{cases} w(p,t) & \text{se } (p,t) \in F \\ 0 & \text{se } (p,t) \notin F \end{cases}$$

- $Post : P \times T \to \mathbb{N}$

$$post(t,p) = \begin{cases} w(t,p) & \text{se } (t,p) \in F \\ 0 & \text{se } (t,p) \notin F \end{cases}$$

PN system — graficamente rappresentabile aggiungendo una etichettatura alla rappresentazione della PN

Basato sul concetto di Marking (marking di una rete N)

$m \in \mathbb{N}^{|P|}$ (vettore) — Assegna ad ogni posto P un valore $\mathbb{N}^+$

P/T net system

$S = \langle N, m_0 \rangle$ m_0 marcatura iniziale

Evoluzione di un P/T net system.

1 Enabling.

t enabled at m iff $m \geq Pre[P, t]$

una transizione t è abilitata da (at) m se tutti il n° di token in input e >= del peso degli archi che collegano input-transizione

P grande, insieme C[P, t] è vettore colonna di C

2 Firing

Se t abilitata da m, t può scattare

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

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)

$\sigma = t_1 \dots t_k \in T$ t.c. $m \xrightarrow{t} m_1 \dots \xrightarrow{t_k} m_k$

scriviamo $m \xrightarrow{\sigma} m_k$

La prof scrive: $\sigma = [t_1, \dots, t_k] \in T$.

Scriviamo $m \xrightarrow{\sigma} m'$ se $\exists \{m_0, \dots, m_k\} : \forall i \in [1, \dots k]\ m_{i-1} \xrightarrow{t_i} m_i$

4-bis Firing vector

$\bar{\sigma}$ vettore caratteristico della sequenza σ

σ è scattabile in m

$m' = m + C \cdot \bar{\sigma}$ stato equazione

5 Linguaggio

$L(S)$ è l'insieme delle firing sequence scattabili da m_0

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

RS reachability set.

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

RG Reachability Graph.

$RG(S)$ rappresentazione del $RS(S)$

- $V = RS(S)$ L'insieme dei nodi = Reachability Set
- $(v_1, v_2) = (m_1, m_2) \in E \Leftrightarrow \exists t \in T$ 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

$N = \langle P, T, Pre, Post, C, cd \rangle$

C: color classes (classi di colori) notare che i colori sono classi quindi C è un set di set

$cd: P \cup T \to C$ definisce 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 $\langle N, m_0 \rangle$ sp $b(p) < \omega$

$\langle N, m_0 \rangle$ è bounded sp $b(p) < \omega \quad \forall p \in P$ di N

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

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

Deadlock

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

t.c. t è abilitata in m $(m > Pre(P, t))$

Liveness

$m[t>$

t è live in $S \iff \forall m \in RS(S) \; \exists \sigma$ 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.

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

N è strutturalmente live sp $\exists m$ t.c. $\langle N, m \rangle$ è live

Reversibility

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

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

(cioè m_0 è home state di N)

Analisi qualitative

Reachability graph e Coverability Graph.

Basati sul concetto di copertura.

$M' > M \iff \forall p \in P \quad M'(p) > M(p)$

M' copre M in ogni posto di M 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 $\forall t$ t.c. $M[t\rangle$:

- calcolo M' t.c. $M \xrightarrow{t} M'$
- se esiste M'' t.c. $M'' \xrightarrow{\sigma} M' \wedge M'' < M$ (algoritmo finisce, applico unbound.
- se M' non è presente in RG ~~lo taggo~~ e lo aggiungo.
- aggiungo un arco tra M e 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 $\forall t$ t.c. $M[t\rangle$:

- calcolo M' t.c. $M \xrightarrow{t} M'$
- se esiste M'' t.c. $M'' \xrightarrow{\sigma} M' \wedge M'' < M$ marco M come ω e non lo visito più
- se M' non è presente in RG ~~lo taggo~~ e lo aggiungo.
- aggiungo un arco tra M e M' e lo chiamo t

Liveness e Marking invariant (di proprietà Π)

$\forall m \in RS(S), \exists m' \in RS(\langle N, m \rangle)$ 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

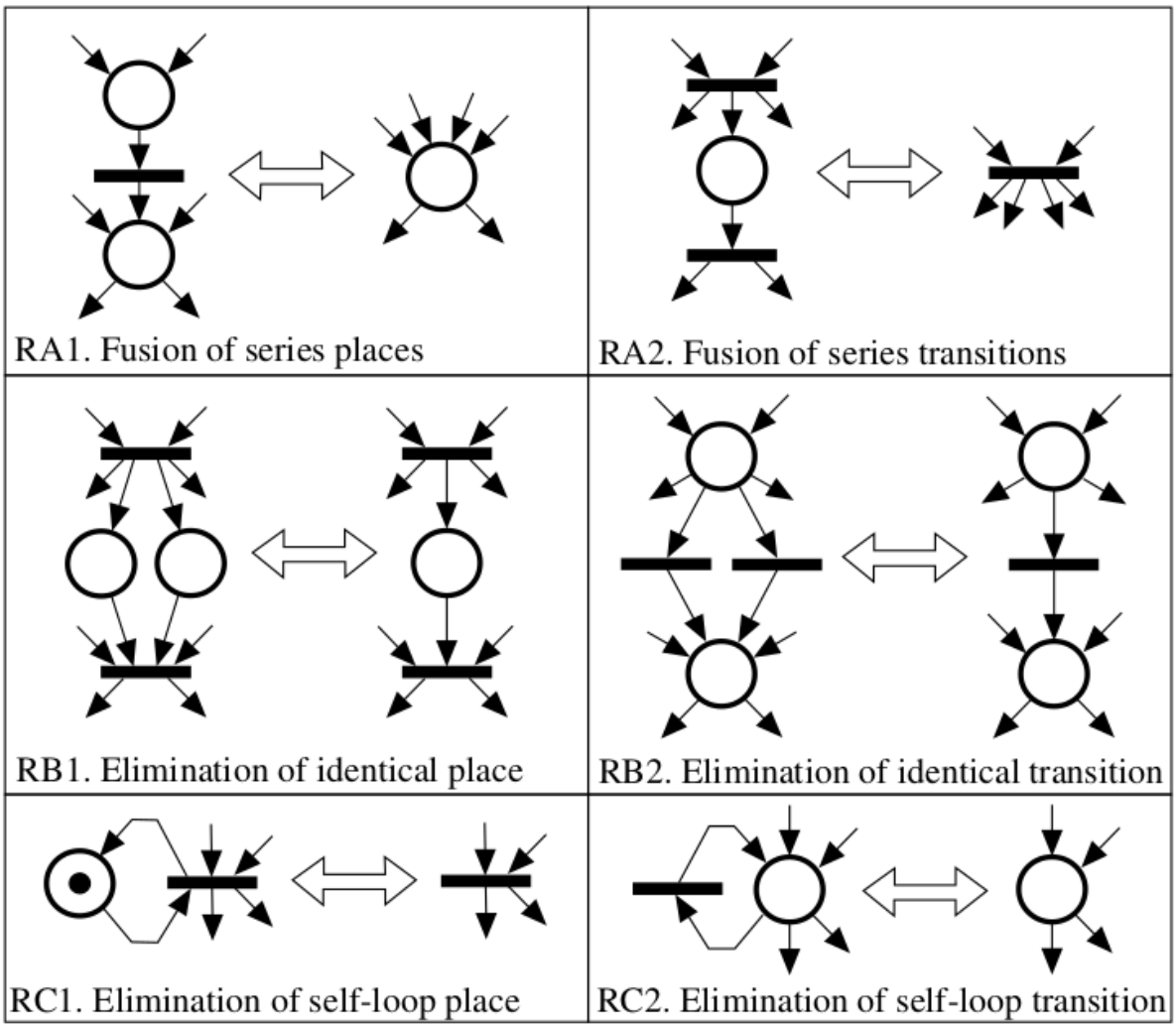
Algoritmo marking invariant:

1 tutti gli elementi RS(S) sono visitati.
2 A elemento visitato m
- se m non rispetta Π
RETURN FALSE
3 RETURN TRUE.

Algoritmo Liveness invariant.

1 Scomporre RG(S) nelle strongly connected components
2 Creo un nuovo grafo dove i nodi sono le SCC e gli archi gli archi che connettono le SCC
3 Calcolo 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 algebric 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-bounded ness, deadlock, ...)

Esprimendo la proprietà all'interno dell eq di stato per poi risolverla come 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

$(y : P \to \mathbb{N})$

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

p-semiflow ⟶ token conservation law

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

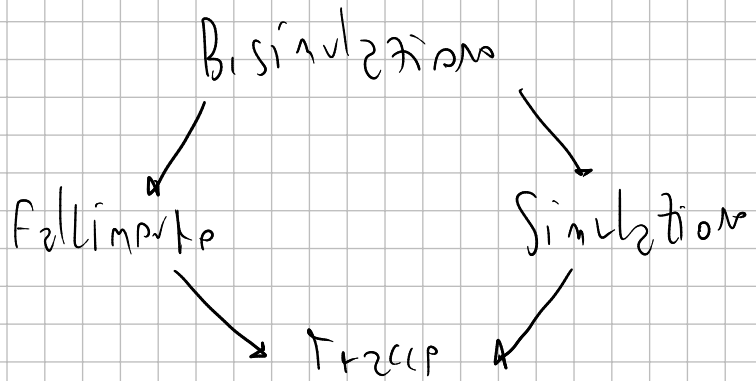
The weighted sum of tokens is constant for all reachable markings

t-semiflow ⟶ cyclic behaviour 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 equivalenze.



se $P \approx R$ in simulazione $\Rightarrow P \simeq Q$ in traccia. (Vedi verso ↓)

Traccia: dato $T(E)$ (traccia di E) l'insieme di tutte le sequenze finite che possono essere prodotte dall'evoluzione di E

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

Fallimento

Un fallimento per l'agente E è la coppia (σ, A)

con $\sigma \in T(E)$, $A \subseteq Act$ per cui

$E \xrightarrow{\sigma} F$ e nessuna azione in A può essere ripetuta da F

E attraverso sigma evolve in F

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

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

Simulazione:

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

- $E \; R \; F$
- $E \xrightarrow{a} E'$, $\exists F'$ t.c. $F \xrightarrow{a} F'$
- $E' \; R \; F'$

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 ogni diciamo che F simula E ($E \lesssim_{sim} F$) se esiste una relazione di simulazione R e Q t.c.

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

Bisimulazione.

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

Weak bisimulation, Bisimulo senza considerare azioni τ

$n \xrightarrow{a} n'$ se $n \xrightarrow{\tau} \dots \xrightarrow{a} \dots \xrightarrow{\tau} n'$

LTL

sintassi.

Kripke $\phi := p \mid \neg\phi \mid \phi \vee \phi \mid X\phi \mid \phi U \phi$

Poi d'ora aggiungo $\phi \wedge \psi \mid F\phi \mid G\phi$

non impararli, sappi solo si possono derivare $\neg(\neg\phi \wedge \neg\psi) \mid \text{true} \vee \phi \mid \neg F \neg \phi$

Semantica

Kripke $M = (S, R, L)$ ← basata sul "modello"

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

$R: R(S) \; S \to S$ ad ogni stato associa un successore univoco

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

Data una formula ϕ diciamo $(M, s) \models \phi$ se ϕ è una formula valida nello stato s del modello M.

Formalmente

$M, s \models p \iff p \in L(s)$ ← con AP è il linguaggio della PV, è l'insieme delle proposizioni soddisfatte

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

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

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

dato M $s \models \phi U \psi \iff \exists j \geq 0$ t.c. $R^j(s) \models \psi$ $\wedge$

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

future e globally si derivano ma comunque sono:

$$s \models F\phi \iff \exists j, j \geq 0 \text{ t.c. } R^j(s) \models \phi$$

$$s \models G\phi \iff \forall k, k \geq 0 \quad R^k(s) \models \phi$$

Prob:

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 $s_0, s_1, s_2, \ldots$ $s_i \in S$

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

$\sigma \models p \iff s_i \models p$

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

$\sigma^i \models \phi U \psi \iff \exists j, j \geq i$ $\sigma^j \models \phi \wedge \forall k \; i \leq k < j \; \sigma^k \models \psi$

Costrutti definiti:

Unconditional $GF\psi$

prima o poi accadrà theta

weak $FG\phi \Rightarrow GF\psi$

se da un certo punto in poi phi diventa vera, accadrà theta

strong $GF\phi \Rightarrow GF\psi$

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

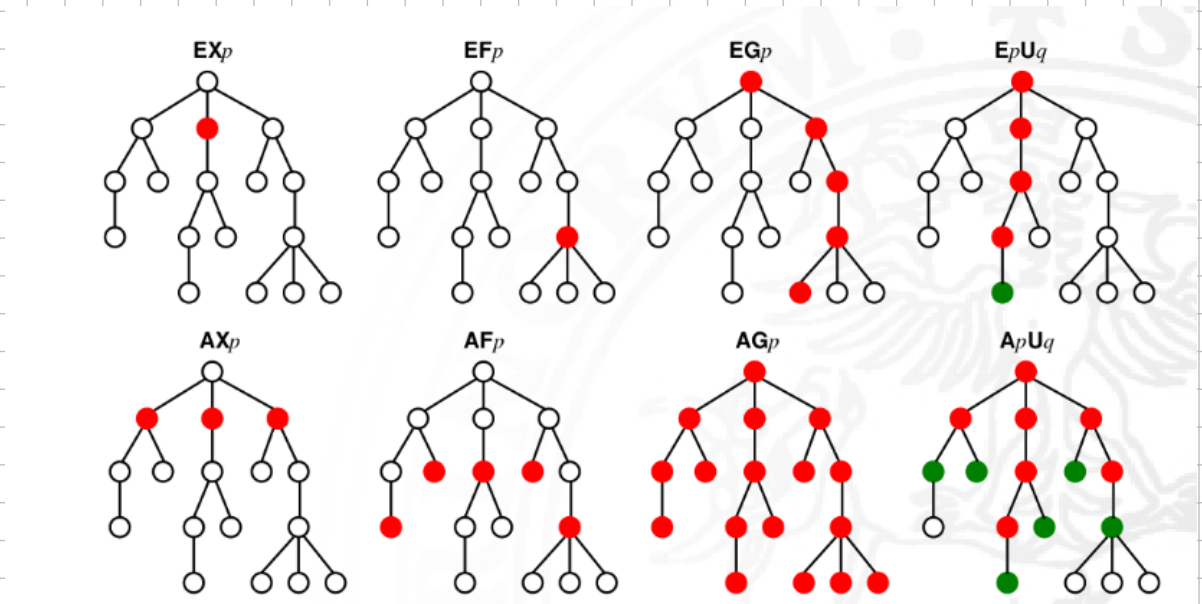
$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\,\varphi \mid E[\varphi\, U\, \varphi] \mid A[\varphi\, U\, \varphi]$$

Future e globally also can be derived

$EF\varphi = E[true\, U\, \varphi]$ $\quad AF\varphi = A[true\, U\, \varphi]$

$EG\varphi = \neg AF\neg\varphi$ $\quad AG = \neg EF\neg\varphi$

$AX = \neg EX\neg\varphi$



Semantica.

Also based on model (S, R, L)

S ed L uguali. R diverso

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

oppure $R \subseteq S \times S$

Quindi nel modello un path $\sigma = s_1, s_2 \ldots$ con $(s_i, s_{i+1}) \in R$

Come in LTL $\sigma[i]$ è i-esimo elemento della sequenza.

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

cioè l'insieme dei path che partono da s.

M, S, per brevità assumiamo M

$s \models p$ sse $p \in L(s)$ — dato p-state un path di p-state è detto p-path

$s \models \neg\varphi$ sse $\neg(s \models p)$

$s \models \varphi \vee \phi$ sse $s \models \varphi \vee s \models \phi$

$s \models EX\varphi$ sse $\exists \sigma \in P_M(s)$ t.c. $\sigma[1] \models \varphi$

$s \models E[\varphi U \psi]$ sse $\exists \sigma \in P_M(s)$ t.c.

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

$s \models A[\varphi U \psi]$ sse $\forall \sigma \in P_M(s)$

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

Fairness.

weak e strong fairness non sono esprimibili in CTL

si estende il modello di Kripke

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

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

insieme di insiemi $F = \{F_1, F_2\}$, $F_1 \in S$, $F_1 = \{s_1, s_2 \dots\}$

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

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

L'algoritmo è un semplice algoritmo iterativo che parte da una sottoformula di psi di lunghezza 1, vede in quali stati è valida, induttivamente prende la formula di lunghezza 2 e via così fino a riottenere psi
Questo può essere fatto grazie alla definizione ricorsiva delle sottoformule in CTL

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

CTL*

state formulas $\phi ::= a \mid \neg\phi \mid \phi \wedge \phi \mid E\psi \mid A\psi$

path formulas $\psi ::= \phi \mid \psi \wedge \psi \mid X\psi \mid \psi U \psi$

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*

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

in CTL* posso inserire path formulas di path formulas

Timed Automata

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

Def Clock constraint

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

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

scriviamo φ $\psi(C)$ oppure $Cstr(C)$

Def Timed automata

$$A = (L, l_0, E, labl, C, clocks, inv, guards)$$

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

l_0: locazione iniziale $l_0 \in L$

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

$labl: L \to 2^{AP}$ il solito

C: set di clocks

$clocks: E \to 2^C$ Funzione di clock resp. ad ogni arco può associare i clock $\in C$ e questi vengono resettati.

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

Invariante $inv: L \to \psi(C)$ ad ogni locazione associa un clock constraint (può sia vietare l'accesso che obbligare l'uscita

Clock evaluation

Def clock valutation: $v: C \to \mathbb{R}^+$

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

Uno stato di un Timed automaton 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 $\varphi, \beta \in Cstr(C)$

$v \models x \leq c$ sse $v(x) \leq c$

$v \models \neg\varphi$ sse $V(v) \not\models \varphi$

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:

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

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

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

- $s_0 = (l_0, v_0)$ con $v_0(x) = 0 \quad \forall x \in C$

lo stato iniziale, con tutti i clock a zero

- le relazioni $\longrightarrow$ definite da regole

1 transizione discrete $\langle l, v \rangle \xrightarrow{t} \langle l', v' \rangle$ SP

$\exists p = (l, l') \in E$ (esiste l'arco)

$v \models guard(p)$

$v' = reset\ clocks(p)$ in v, $v' \models inv(l')$

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

$d \in \mathbb{R}^+ \quad \forall d' \leq d, v + d \models inv(l)$ for time elapse

---

Come distinguo un deadlock da un let time elapse?

Elapsed time.

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

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

$$\Delta(\sigma, i+1) = \Delta(\sigma, i) = \begin{cases} 0 & \text{se } z_i \text{ è transizione } (*) \\ z_i & \text{se } z_i \in \mathbb{R} \end{cases}$$

le transizioni sono istantanee

se $\lim_{i \to \infty} \Delta(\sigma, i) = \infty$ il cammino 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))

TCTL

$$\varphi = p \mid \alpha \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid E[\varphi U \varphi] \mid A[\varphi U \varphi] \mid z \text{ in } \varphi$$

tutti uguali a CTL a parte α e z in φ

- $\alpha \in \Psi(C \cup D)$ Formula clock constraint.
  - C: clocks dell'automa
  - D: clocks di formula

- z in φ freeze identifier:

$(z \text{ in } \varphi)$ vale in s se φ vale in s, stato in cui il clock z viene fatto partire.

Esempio $E[\phi \, U_{\leq z} \, \varphi]$

ϕ vale e parto z unità di tempo φ diventa vera.