Università degli Studi di Salerno

DIPARTIMENTO DI INFORMATICA

REGULAR MODEL CHECKING

VERIFICA DI SISTEMI PARAMETRICI

CORSO DI
TECNICHE AUTOMATICHE PER LA CORRETTEZZA DEL SOFTWARE

ANNO ACCADEMICO
2016/2017

ALESSANDRA ZULLO

PROF. SALVATORE LA TORRE

STRUTTURA DELLA PRESENTAZIONE

- Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)
- Model Checking e i sistemi parametrici
- Introduzione al Regular Model Checking
- Sistemi parametrici: token passing protocol
- Verifica di sistemi
- RMC per verifica di sistemi parametrici

STRUTTURA DELLA PRESENTAZIONE

- **Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)**
- Model Checking e i sistemi parametrici
- Introduzione al Regular Model Checking
- Sistemi parametrici: token passing protocol
- Verifica di sistemi
- RMC per verifica di sistemi parametrici

INTRODUZIONE

Uno degli obiettivi principali nell'ambito della **verifica software** consiste nel capire come un sistema/software lavora.

Per comprendere come un sistema lavora sono utilizzati dei **metodi formali** cioè dei metodi basati su *modelli matematici* che permettono di ragionare su un sistema.

Lo **scopo** principale della ricerca in quest'ambito è quello di trovare dei metodi che permettano di **automatizzare** tali ragionamenti.

Una delle tecniche che negli ultimi anni ha avuto ampio successo è la tecnica del **model checking**.

COS'È IL MODEL CHECKING

Nel **model checking** il sistema che deve essere analizzato è modellato utilizzando definizioni di *insiemi di stati* e *relazioni di transizioni* al fine di determinare come un sistema cambia nel tempo.

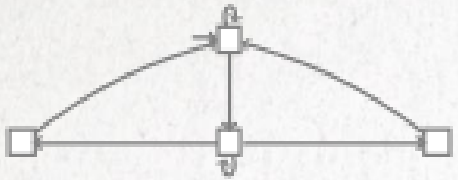
L'**obiettivo** principale del model checking è verificare se un dato sistema possiede determinate proprietà.

Dato che il model checking deve elaborare un *modello del sistema* anche le **proprietà** saranno espresse utilizzando un linguaggio formale, più nello specifico le proprietà sono specificate utilizzando la **logica temporale** *(es. LTL, CTL, PLTL, ecc).*

La **logica temporale** non è altro che un'estensione della *logica proposizionale* al fine di rappresentare delle realtà che mutano nel tempo.

Essa utilizza i classici operatori della **logica booleana** *(es. AND, OR, NOT, ecc)* ai quali sono aggiunti dei **connettivi temporali** *(es. next,eventually, until, ecc).*

MODEL CHECKER



Modello di un sistema
FSM - Kripke

$\top$

Proprietà del sistema
LTL

φ

Model Checker

SI

NO (controesempio)

MODEL CHECKING – DEFINIZIONE DI MODELLO

- Con il termine **modello** si fa riferimento ad un modo per *rappresentare il comportamento dinamico* (cioè che può mutare) di un sistema, che consiste nell'uso di descrizioni matematiche basate sulla definizione di *insiemi di stati e di transizioni.*

- Una **configurazione** di un sistema è la rappresentazione degli *stati del sistema nel mondo reale* (ad es. il valore di qualche variabile in un programma).

- Una **configurazione** è una parola sull'alfabeto del sistema mentre un'**esecuzione** è l'insieme di configurazioni del sistema.

MODEL CHECKING – PROPRIETÀ

- Per specificare il comportamento viene utilizzata la **logica temporale**.
- La logica temporale **specifica il comportamento** e non il sistema che si comporta secondo tale logica.
- Vi sono differenti logiche temporali.
- Nel RMC per studiare le esecuzioni del sistema si lavora su *due dimensioni* : *spazio* e *tempo* che sono rappresentati mediante una **matrice**.
- **Monadic Second Order Logic** (MSO) è composta dall'insieme di *costrutti* per la gestione dello spazio e del tempo.
- **RMC** usa **Mona*** che lavora su insiemi regolari di parole.

*J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige,T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic

STRUTTURA DELLA PRESENTAZIONE

- Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)
- **Model Checking e i sistemi parametrici**
- Introduzione al Regular Model Checking
- Sistemi parametrici: token passing protocol
- Verifica di sistemi
- RMC per verifica di sistemi parametrici

MODEL CHECKING E SISTEMI PARAMETRICI

Il **model checking** ha avuto molto successo nella *verifica di sistemi* aventi un numero *limitato di stati* (sistemi a stati finiti).

I **sistemi parametrici** sono quei sistemi che contengono un *numero illimitato* di componenti* e rientrano tra quei sistemi composti da un *numero infinito/illimitato di stati.***

Il principale problema per la verifica di tali sistemi è *riuscire a definire una rappresentazione **finita** per un insieme infinito di stati.*

* Programmi sequenziali per stringhe o array oppure algoritmi distribuiti per un numero non noto di attori;

** B. Boigelot and P. Wolper. Symbolic verification with periodic sets. Springer Verlag,1994.

STRUTTURA DELLA PRESENTAZIONE

- Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)
- Model Checking e i sistemi parametrici
- **Introduzione al Regular Model Checking**
- Sistemi parametrici: token passing protocol
- Verifica di sistemi
- RMC per verifica di sistemi parametrici

REGULAR MODEL CHECKING – OVERVIEW (1)

Come per altri framework anche il RMC ha le seguenti proprietà:

- I sistemi sono modellati mediante dei **sistemi di transizione** composti da configurazioni e transizioni tra esse;
- Le **configurazioni** sono *parole finite* definite su un alfabeto finito;
- Le **transizioni** sono definite mediante *relazioni di transizioni* che permettono il passaggio da una parola all'altra

REGULAR MODEL CHECKING – OVERVIEW (2)

DEFINIZIONE 1: Sistema di transizione

Un **sistema di transizione** è una tupla $(C, I, \Rightarrow)$ dove

- C è l'*insieme delle configurazioni;*
- I è l'*insieme delle configurazioni iniziali;*
- $\Rightarrow \subseteq CxC$ è la *relazione di transizione.*

Un'**esecuzione** del sistema di transizione $(C, I, \Rightarrow)$ è una sequenza (finita o infinita) di configurazioni $c_0 c_1 c_2 \ldots$ tali che $c_0 \in I$ e $c_\mathrm{i} \Rightarrow c_{i+1}$ $\forall \mathrm{i} \geq 0$

REGULAR MODEL CHECKING - MODELLO

Esso si basa sulla *teoria dei linguaggi formali* e modella il sistema utilizzando delle **parole** per descrivere gli stati.

Per la rappresentazione degli **insiemi di stati** e di **relazioni di transizioni** sono utilizzati gli ***insiemi regolari***; per cui l'obiettivo principale è quello di trovare una **codifica delle parole** che genera insiemi regolari.

Non è stato provato che il RMC sia il metodo più efficiente ma attualmente è quello in grado di verificare automaticamente e uniformemente tutti i sistemi citati in precedenza.

Il **regular model checking** nasce per essere un framework per la *verifica automatica di sistemi a stati infiniti*.

Tra questi sistemi rientrano:

- I sistemi con contatori;
- I sistemi con stacks;
- I sistemi con queues;
- I **sistemi parametrici**.

I loro stati sono codificabili mediante l'utilizzo di parole in modo tale che l'insieme degli stati e quindi delle parole sono **insiemi regolari**.

RMC – INSIEMI REGOLARI

Nel RMC è di fondamentale importanza rappresentare l'*insieme delle configurazioni iniziali* e le *relazioni di transizioni* in un modo uniforme che permetta di **rappresentare un sistema con un numero arbitrario di processi**. Come visto in precedenza per tale motivo si utilizzano gli **insiemi regolari** e le **relazioni regolari**.

DEFINIZIONE 2: Insieme regolare

Dato un alfabeto Σ .

Un **insieme regolare** su Σ è un sottoinsieme di Σ^* ($\mathcal{L} \subseteq \Sigma^*$) che può essere costruito dall'insieme di simboli derivanti dall'unione di $\mathcal{L}_1 \cup \mathcal{L}_2$, dalla concatenazione di $\mathcal{L}_1 \cdot \mathcal{L}_2$ o dalla chiusura di Kleene $\mathcal{L}^*$.

Ad es. $a \cdot b^+$ *denota l'insieme delle parole abb……b*

RMC – RELAZIONI REGOLARI

DEFINIZIONE 3:Prodotto cartesiano

Dato un *alfabeto* Σ e date due *parole di lunghezza n* $w_1, w_2 \in \Sigma^n$ il **prodotto cartesiano** di w_1 e w_2 denotato con $w_1 \times w_2$ è la *parola* $w \in (\Sigma x \Sigma)^n$ tale che $w(\mathrm{i}) = \big(w_1(i), w_2(\mathrm{i})\big)$ per $i<n$

Esempio $W_1 = aab$ e $W_2 = baa$ allora il prodotto cartesiano sarà $W = W_1 \times W_2 = ((a,b)\ ,\ (a,a),\ (b,a))$

DEFINIZIONE 4:Relazione regolare

Una **relazione regolare** sulle parole è una *relazione binaria R che preserva la lunghezza* tale che $\{w \times w' : R(w, w')\}$ è un *insieme regolare* che descrive *R*.

RMC – ESEMPIO DI MODELLO - STATI

1. **RAPPRESENTAZIONE DELL'INTERO n**

Nel RMC esso è rappresentato dalla **parola** $a^n \perp^m$ per qualche m sull'alfabeto $\{a, \perp\}$.

Il simbolo permette di specificare che la transizione **preserva la lunghezza** (cioè non comporta la modifica della lunghezza delle parole) inoltre fornisce un modo per indicare che la parola possa mutare.

2. **RAPPRESENTAZIONE DEGLI INTERI PARI**

Nel RMC essi sono rappresentati dall'espressione regolare $(aa)^* \perp^*$

OBIETTIVO: Trovare una codifica che permetta di non ottenere insiemi non regolari.

REGULAR MODEL CHECKING – OVERVIEW (3)

Dato che il RMC utilizza gli insiemi regolari e le relazioni regolari è utilizzata la nozione di sistema di transizione regolare.

DEFINIZIONE 5: Sistema di transizione regolare

Un **sistema di transizione regolare** è una tupla $(\Sigma^*, I, \Rightarrow)$ dove

- $I \subseteq \Sigma^*$ è un *insieme regolare* di parole *(configurazioni);*
- $\Rightarrow \subseteq \Sigma^* \, x \Sigma^*$ è una *relazione regolare* definita sulle parole

REGULAR MODEL CHECKING – OVERVIEW (4)

Per la rappresentazione di un **insieme regolare** è utilizzato un automa di parole.

DEFINIZIONE 6: Automa di parole

Sia Σ un insieme. Un **automa di parole** A su Σ è una tupla $(Q, S, \rightarrow, F)$dove:

- Q è l'*insieme* di stati (finito o infinito);
- $S \subseteq Q$è l'*insieme degli stati iniziali ;*
- $\rightarrow Q \times \Sigma \times Q$ è la *relazione di transizione;*
- $F \subseteq Q$ è l'*insieme degli stati di accettazione.*

Un'**esecuzione** di A su una parola w è una *sequenza di stati* $q_0 q_1 \ldots q_n$ tali che $q_i \xrightarrow{w_i} q_{i+1}$ per ogni $i<n$.

Una **parola** è **accettata** se esiste un'esecuzione $q_0 q_1 \ldots q_n$ tale $q_0 \in S$ e $q_n \in F$.

REGULAR MODEL CHECKING – OVERVIEW (5)

Per la rappresentazione di una **relazione regolare** è utilizzato un trasduttore.

DEFINIZIONE 7: Trasduttore

Un **trasduttore** T su Σ è un *automa di parole* $(Q, S, \rightarrow, F)$ definito su $\Sigma \times \Sigma$ dove:

- Q è l'*insieme* di stati (finito o infinito);
- $S \subseteq Q$è l'*insieme degli stati iniziali ;*
- $F \subseteq Q$ è l'*insieme degli stati di accettazione.;*
- $\rightarrow: Q \times (\Sigma \times \Sigma) \times Q$ è la *relazione di transizione* definita come segue:

$$q_i \xrightarrow{w_i \times w_i} q_{i+1}$$

STRUTTURA DELLA PRESENTAZIONE

- Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)
- Model Checking e i sistemi parametrici
- Introduzione al Regular Model Checking
- **Sistemi parametrici: token passing protocol**
- Verifica di sistemi
- RMC per verifica di sistemi parametrici

SISTEMI PARAMETRICI

TOKEN PASSING PROTOCOL

- Consideriamo un sistema di **processi** connessi in **lineare**;
- Il sistema è composto da *n* processi;
- Ogni processo può trovarsi in uno dei due seguenti stati:
  1. $\perp$: il processo non possiede il token;
  2. **T** : il processo possiede il token.
- Le **configurazioni** sono definite mediante le parole su $\{\perp, T\}^n$ (insieme di parole di *n* elementi);
- Con $q[i]$ denotiamo il contenuto della parola q in posizione i ($1 \leq i \leq n$) e rappresenta lo *stato* del *processo i*;

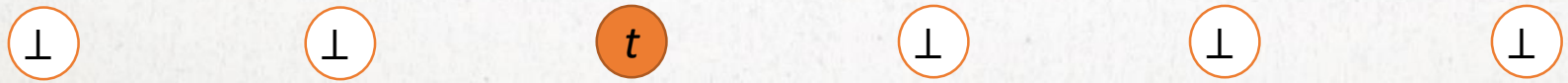
Figura 1: 5 processi connessi in lineare

SISTEMI PARAMETRICI- ESEMPIO

CONFIGURAZIONE INIZIALE

TRANSIZIONE

TRANSIZIONE



SISTEMI PARAMETRICI – RMC (1)

Il *token passing protocol* è modellato dal seguente **sistema di transizione**:

- $C = \{\perp, \mathrm{T}\}^*$;
- I è l'insieme delle *parole* nella forma $\mathrm{T}\ \perp^*$ cioè le *configurazioni in cui il primo processo possiede il token e gli altri processi non lo possiedono;*
- $\Rightarrow$ è l'insieme delle coppie ($(\{\perp, \mathrm{T}\}^*\ \mathrm{T}\ \perp\ \{\perp, \mathrm{T}\}^*)$, $(\{\perp, \mathrm{T}\}^*\ \perp\ \mathrm{T}\ \{\perp, \mathrm{T}\}^*)$).

Un'esecuzione del sistema è rappresentata dalla seguente matrice:

$$
\begin{array}{l}
T\ \perp\perp\perp\perp\perp\perp \\
\perp\ T\ \perp\perp\perp\perp\perp \\
\perp\perp\perp\ T\ \perp\perp\perp \\
\perp\perp\perp\perp\ T\ \perp\perp \\
\perp\perp\perp\perp\perp\ T\ \perp \\
\perp\perp\perp\perp\perp\perp\ T
\end{array}
$$

Ogni riga rappresenta una configurazione e le coppie di righe rappresentano un passo di computazione

SISTEMI PARAMETRICI – RMC (2)

Il **sistema di transizione regolare** per il *token passing protocol* è definito da $(\Sigma^*, I, \Rightarrow)$ dove:

- $\Sigma^* = \{\bot, \mathrm{T}\}^*$;
- I è l'*insieme regolare* $\mathrm{T}\ \bot^*$;
- $\Rightarrow$ è definita come segue:

$$w \Rightarrow w' \text{ se e solo se } w \times w' \text{ è nell'insieme } \{\bot, \bot\}^* \quad (\mathrm{T}\ \bot)(\bot\ \mathrm{T})\{\bot, \bot\}^*$$

SISTEMI PARAMETRICI- NOTAZIONE 1

Per modellare il sistema è quindi utilizzato il sistema di transizione.

Allo stesso modo utilizziamo un **automa di parole** per rappresentare l'*insieme regolare* che definisce l'insieme degli stati iniziali del sistema del *token passing protocol.*

L'insieme degli stati iniziali è T $\perp^*$ ed è rappresentato dal seguente automa:

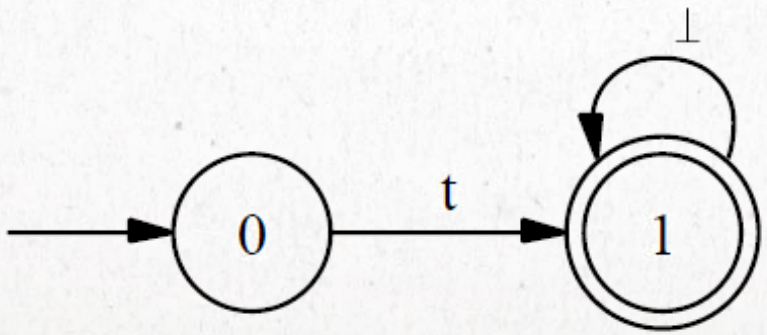


Figura 2: Insieme iniziale degli stati

SISTEMI PARAMETRICI- NOTAZIONE 2

Per modellare la relazione di transizione utilizziamo un **trasduttore** in grado di rappresentare una relazione regolare.

La relazione di transizione del token passing protocol in esame è la seguente:

$$\{\perp, \perp\}^* \quad (T\ \perp)(\perp\ T)\{\perp, \perp\}^*$$

ed è rappresentata dal trasduttore T:

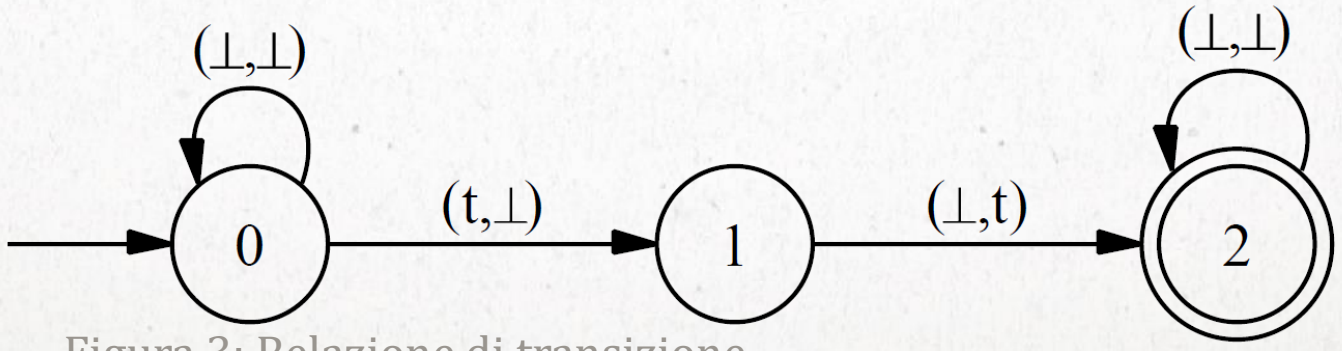


Figura 3: Relazione di transizione

STRUTTURA DELLA PRESENTAZIONE

- Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)
- Model Checking e i sistemi parametrici
- Introduzione al Regular Model Checking
- Sistemi parametrici: token passing protocol
- **Verifica di sistemi**
- RMC per verifica di sistemi parametrici

VERIFICA DI SISTEMI

Per **verifica** di un **sistema** si intende quell'insieme di tecniche che permettono di *verificare che un sistema sia* ***corretto***.

Un **sistema** è **corretto** se si può provare che soddisfa una *specifica* cioè che si comporti in un certo modo.

Una **specifica** è l'*insieme di proprietà* che il sistema deve possedere.

Le **proprietà** di un sistema possono essere classificate in due grandi categorie:*

1. **Safety** : nessun evento catastrofico deve verificarsi;
2. **Liveness** : qualcosa di buono può eventualmente accadere.

* L. Lamport. Proving the correctness of multiprocess programs.

VERIFICA DI SISTEMI ESEMPIO

Tra gli esempi di verifica maggiormente riportati vi è la verifica di un **sistema di controllo del traffico.**

Un sistema di controllo del traffico è composto da un certo numero di semafori.

Un sistema di controllo del traffico **corretto** è un sistema in cui non si verifica mai il caso in cui *tutti i semafori sono verdi nello stesso istante*, in tal caso si dice che il sistema è **safe**.

Allo stesso modo un sistema di controllo del traffico *non deve mai rimanere sempre nello stato in cui tutti i semafori sono rossi*, altrimenti il sistema sarebbe **inutilizzabile**.

Infine un sistema di controllo del traffico è corretto e soddisfa la *proprietà di **liveness*** se prima o poi ogni semaforo diviene verde.

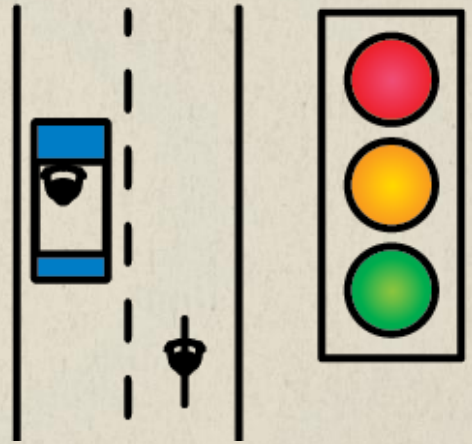


Figura 4: Sistema di controllo del traffico

VERIFICA DI SISTEMI - 2

Per verificare una proprietà si ragiona sulle **esecuzioni** del sistema: se vi è una esecuzione in cui la proprietà non è soddisfatta allora il sistema non è corretto

VERIFICA DI UNA PROPRIETA' SAFETY

E' identificato un insieme di **configurazioni non buone** *(bad)* cioè configurazioni per le quali la proprietà safety non è soddisfatta.

Se la proprietà non è verificata significa che qualche configurazione non buona è raggiungibile a partire da qualche configurazione iniziale.

Per verificare se la proprietà di safety è soddisfatta bisogna determinare **l'insieme di configurazioni non buone** *B,* poi determinare **l'insieme di configurazioni raggiungibili** da una configurazione iniziale $I \Rightarrow^*$ e infine intersecare i due insiemi; se **l'insieme risultante è *non vuoto**** allora la proprietà non è soddisfatta.

$^* (I \Rightarrow^*) \cap B \neq \emptyset$

VERIFICA DI SISTEMI - 3

VERIFICA DI UNA PROPRIETA' LIVENESS

Si presenta nella forma *"qualcosa di buono accadrà durante l'esecuzione del sistema."*

Spesso questa proprietà è verificata utilizzando dei **requisiti di correttezza** che affermano che prima o poi le azioni devono essere eseguite oppure rifiutate *(scheduler).*

Per verificare che tale proprietà è soddisfatta si procede come segue:

Dato un insieme di configurazioni iniziali, un insieme di configurazioni di accettazione e una relazione di transizione, si cerca una computazione infinita che visita l'insieme di configurazioni di accettazioni un numero infinito di volte. Se il loop non è presente allora la proprietà è soddisfatta.

Tornando all'esempio del semaforo si cerca una sequenza di configurazioni $c_0 \ldots c_n$ con $c_{0=} c_n$ e tale che per $i < n$ si ha che $c_i \Rightarrow c_{i+1}$ e c_i è una configurazione in cui il semaforo è rosso.

STRUTTURA DELLA PRESENTAZIONE

- Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)
- Model Checking e i sistemi parametrici
- Introduzione al Regular Model Checking
- Sistemi parametrici: token passing protocol
- Verifica di sistemi
- **RMC per verifica di sistemi parametrici**

VERIFICA DI SISTEMI PARAMETRICI (1)

Genericamente quindi fare la verifica di un sistema può significare calcolare l'insieme di configurazioni raggiungibili (**reachability**) e cercare dei possibili cicli raggiungibili (**reachable loops**).

Per *sistemi a stati infiniti* tali problemi non sono sempre semplici.

*Supponiamo di voler verificare che vi sia **sempre ed esattamente un token** nel sistema.*

Consideriamo essa come una **proprietà safety** e quindi verificabile calcolando l'insieme delle configurazioni raggiungibili.

Figura 5: Sistema parametrico

VERIFICA DI SISTEMI PARAMETRICI (2)

Tale problema nei sistemi a stati finiti può esser risolto calcolando i *punti fissi raggiungibili (**reachability fixpoint**).*

$$C_0 = I$$
$$C_{k+1} = C_k \cup C_k \circ \Longrightarrow$$

Applicando tale algoritmo all'esempio del sistema parametrico si parte con $C_0 = t \perp^*$ e *applicando la relazione di transizione in maniera ripetitiva avremo:*

$$C_1 = \perp t \perp^*$$
$$C_2 = \perp\perp t \perp^*$$
$$C_3 = \perp\perp\perp t \perp^*$$
$$\vdots$$
$$C_k = \perp^k t \perp^*$$

Non importa quanto grande sia scelto *k* non ci sarà mai il caso in cui $C_k = \perp^* t \perp^*$.

VERIFICA DI SISTEMI PARAMETRICI (3)

TECNICA DI ACCELERATION (1)

- La relazione di transizione del sistema preso in esame è rappresentata dal trasduttore T;
- Lo scopo dell'utilizzo della **tecnica** di **accelerazione** è calcolare **T*** cioè determinare la ***chiusura transitiva*** di **T**.

DEFINIZIONE : Trasduttore storia

Dato un trasduttore T=$(Q, S, \rightarrow, F)$. Il **trasduttore storia** per T è il trasduttore $T_{hist} = (Q^*, S^*, \rightarrow, F^*)$ dove $\rightarrow: Q^* \times \Sigma \times Q^*$ è definita come $q_0 q_1 \cdots q_{m-1} \xrightarrow[a']{a} q'_0 q'_1 \cdots q'_{m-1}$ se e solo se esiste $a = a_0, a_1, \ldots, a_m = a'$ *tale che* $q_j \xrightarrow[a_{j+1}]{a_j} q'_j$ per ogni *j<m. Informalmente gli stati di* T_{hist} *sono parole composte dagli stati di* T

TEOREMA: Il trasduttore storia T_{hist} per T accetta T* .

*Nilsson, M. 2005. Regular Model Checking. Uppsala. ISBN 91-554-6137-9 – pagina 17

VERIFICA DI SISTEMI PARAMETRICI (4)

TECNICA DI ACCELERATION (2)

Per il trasduttore T del token passing protocol **non vi è un trasduttore a stati finiti che accetta T*** perché l'insieme delle parole $w \times w'$ tali che T*(w, w') è un ***insieme non regolare***.

Per dimostrarlo consideriamo i due seguenti *insiemi regolari:*

1. $L_1 = (\perp T)^*$ insieme regolare;
2. $L_2 = \perp^* T^*$ insieme regolare;

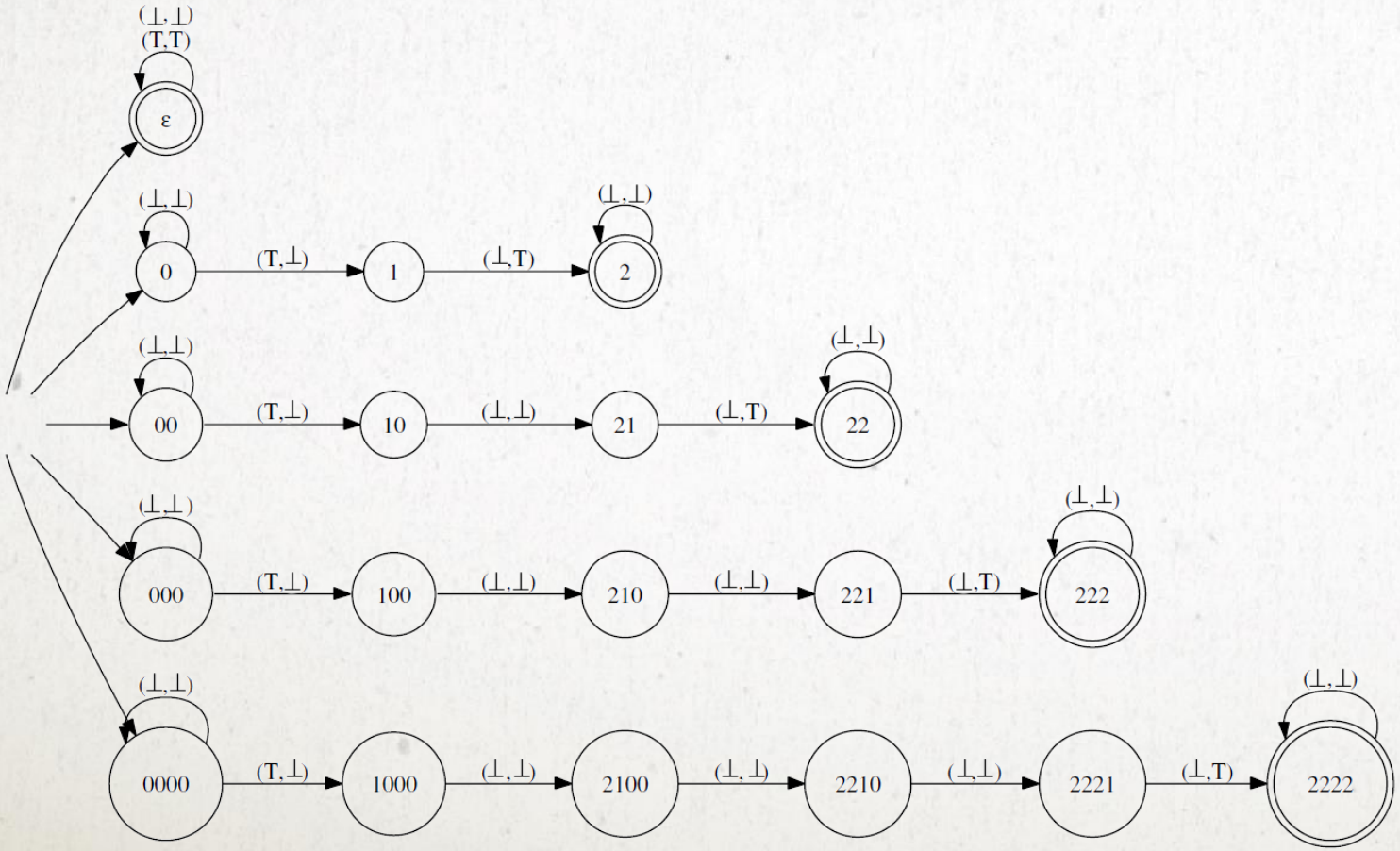
L'insieme $L_1 \circ$T*$\cap L_2 = \{\perp^n T^n : n \geq 0\}$ non è un insieme regolare, ciò viola la proprietà degli insiemi regolari quindi T* non può essere un insieme regolare.

Per risolvere tale problema la tecnica proposta consiste nel calcolare $(I \circ T^*) \cap B \neq \emptyset$.

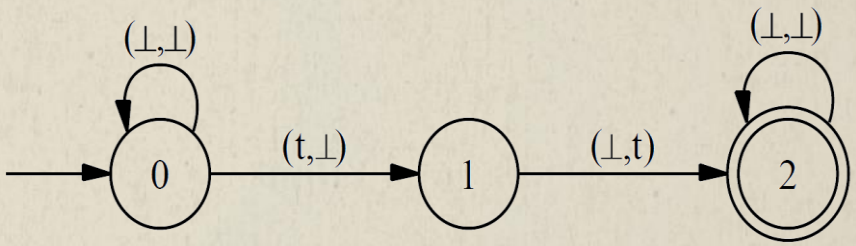
VERIFICA DI SISTEMI PARAMETRICI (5A)

TECNICA DI ACCELERATION (3)

Il trasduttore storia T_{hist} è il seguente:



Trasduttore T

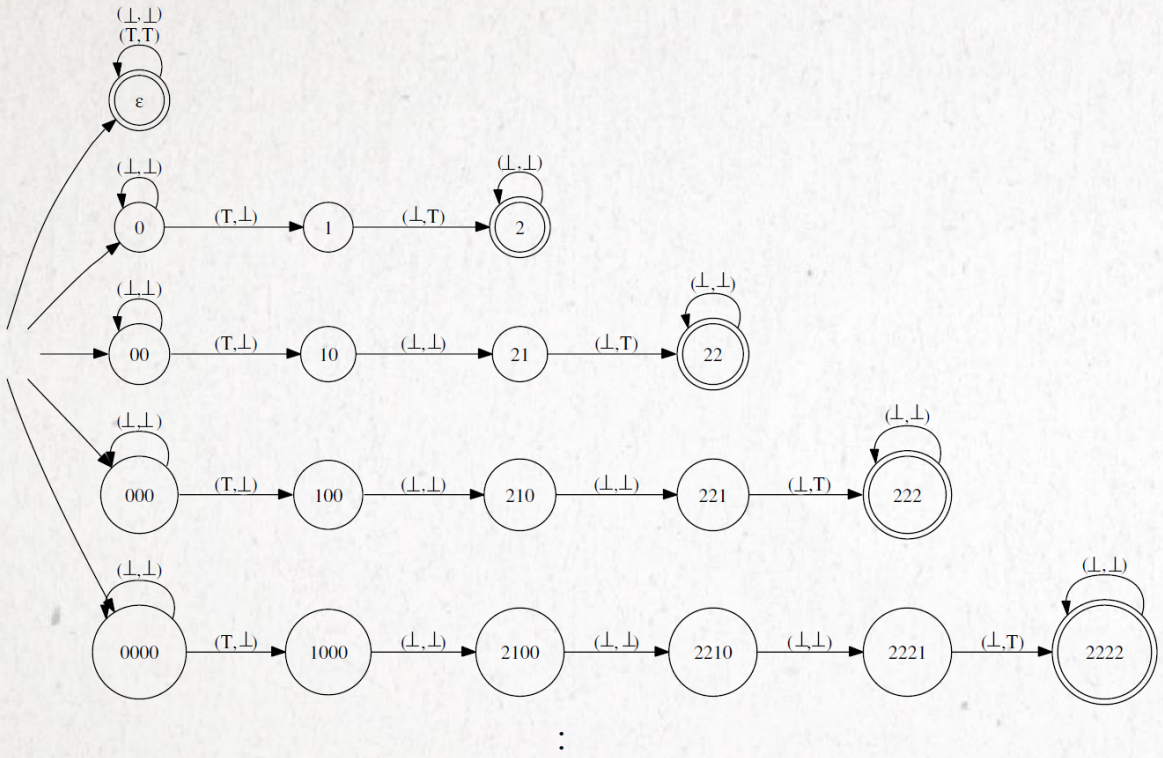


T_{hist} è un **trasduttore a stati infiniti** , il che rende impraticabile l'analisi.

Per risolvere tale problema si cerca un *trasduttore a stati finiti* che accetta lo stesso linguaggio.

VERIFICA DI SISTEMI PARAMETRICI (5B)

TECNICA DI ACCELERATION (4)



Il sotto-trasduttore con stati di lunghezza n accetta T^n

Trasduttore T^3:passaggio del token di 3 processi

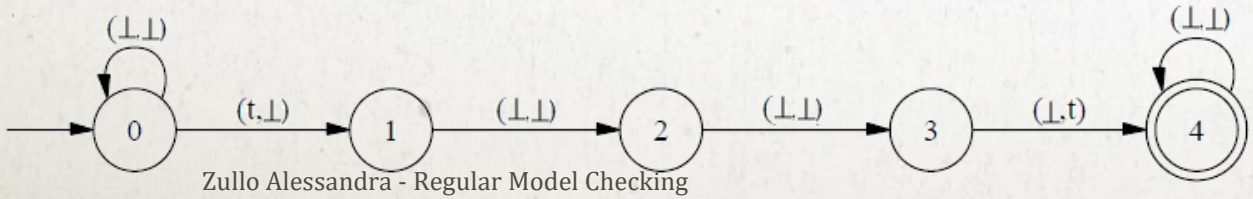


Figura 5: Trasduttore che rappresenta T^3

VERIFICA DI SISTEMI PARAMETRICI (6)

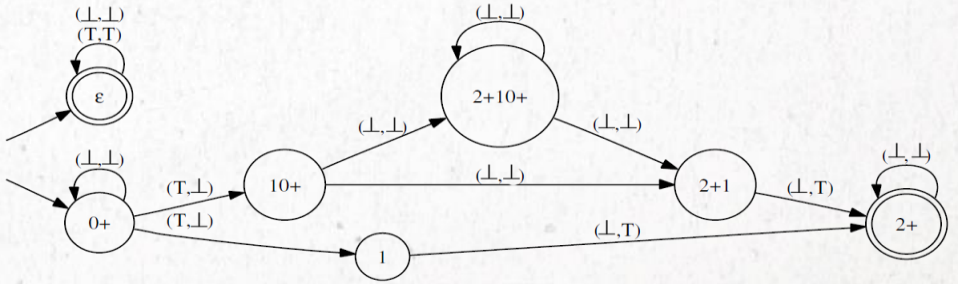
TECNICA DI ACCELERATION (5)

Si determina un **trasduttore che accetta lo stesso linguaggio del trasduttore storia** utilizzando la tecnica di **equivalenza***. Tale trasduttore può essere costruito utilizzando sia la **subset construction**** (costruzione on-the fly), sia **incremental construction*****. Applicando la tecnica di equivalenza si ottiene il trasduttore **quoziente** $T_{hist}/\simeq$ che è un **trasduttore a stati finiti**.

- Informalmente $T_{hist}/\simeq$ è il trasduttore ottenuto unendo le colonne equivalenti di T_{hist} un singolo stato.

Nel *token passing protocol* $T_{hist}/\simeq$ è ottenuto *ignorando* le **ripetizioni di 0** e **2.**

Esempio: $00112 \simeq 0112222$ *mentre* $00112 \not\simeq 11222$



* Nilsson, M. 2005. Regular Model Checking. Acta Universitatis Upsaliensis. Uppsala Dissertations from the Faculty of Science and Technology 60. 149 pp. Uppsala. ISBN 91-554-6137-9 – Capitolo 4.

**M. O. Rabin and D. Scott. Finite automata and their decision problems. o Dexter C. Kozen. Automata and Computability. Springer-Verlag,

*** Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. Regular model checking made simple and efficient.

VERIFICA DI SISTEMI PARAMETRICI (7)

PROPRIETÀ LIVENESS (1)

- La verifica della proprietà liveness è più complicata da effettuare.
- Tale proprietà si applica ad un sistema in **continuo progresso**.
- Il sistema deve essere sotto un'**assunzione di equità** (fairness assumption): *se due semafori sono controllati da due sistemi differenti non sono equi* perché se un semaforo non è mai verde potrebbe essersi verificato un guasto nel suo sistema di controllo ciò non implica che il guasto si sia verificato anche sull'altro sistema.
- Per la verifica si cerca un'esecuzione che soddisfi l'**assunzione di equità** ma viola la **proprietà di liveness**.
- Si estende il **sistema di transizione** in un **trasduttore di Büchi** cioè un sistema di transizione *esteso* composto da un *insieme di transizioni finali* che nelle esecuzioni possono essere scelte *infinite volte.*

VERIFICA DI SISTEMI PARAMETRICI (8)

PROPRIETÀ LIVENESS (2)

DEFINIZIONE : Trasduttore di Buchi

Un **trasduttore di Buchi** su un alfabeto Σ è una tupla (Σ^*, I, T, F) dove:

- (Σ^*, I, T) è un *sistema di transizione regolare*;
- F è un *trasduttore* su Σ che accetta un *insieme di transizioni finali*.

Un'**esecuzione** del trasduttore di Buchi è un'*esecuzione infinita* $w_0\, w_1$_____ su (Σ^*, I, T) tale che vi sono *infiniti indici i* con $(w_i, w_{i+1}) \in F$.

- Genericamente le proprietà liveness sono espresse usando la logica LTL(MSO). Il trasduttore di Buchi è una rappresentazione delle formule in questa logica.*

* P.A. Abdulla, B. Jonsson, Marcus Nilsson, Julien d'Orso, and M. Saksena. Regular model checking for MSO + LTL.

VERIFICA DI SISTEMI PARAMETRICI (9)

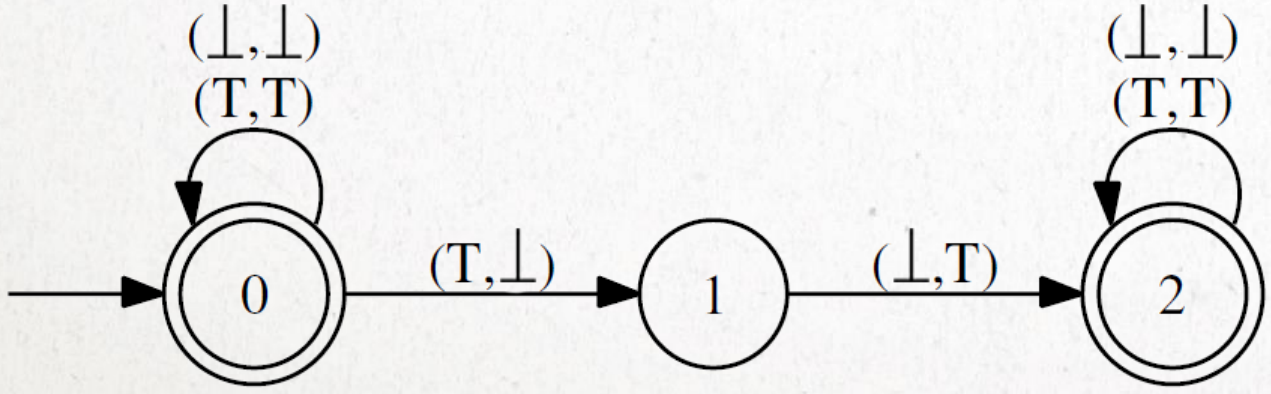
PROPRIETÀ LIVENESS (3)

- Un esempio di proprietà liveness nel *toking passing protocol* è che il **processo più a destra prima o poi deve avere il token**.
- Nell'esempio trattato, l'unica operazione effettuata dal sistema è il passaggio del token al processo a destra più vicino, per cui è semplice assumere che prima o poi il processo più a destra riceverà il token.
- Nelle situazioni reali un **sistema di token passing protocol potrebbe fare altre operazioni**, che rappresenteremo con delle **transizioni idle** (inutili).
- Le **transizioni idle** sono delle transizioni che non eseguono nessuna operazione; le utilizzeremo per poter dimostrare com'è possibile verificare se una *proprietà liveness* è soddisfatta o meno.

VERIFICA DI SISTEMI PARAMETRICI (10)

PROPRIETÀ LIVENESS (4)

Il trasduttore ottenuto aggiungendo delle transizioni idle è il seguente:



Questo trasduttore non risolve il problema, infatti nella seguente esecuzione la proprietà non è soddisfatta.

Infatti dopo due esecuzioni il sistema non fa nient'altro (sceglie transizioni idle)

$$
\begin{array}{ccccc}
T & \bot & \bot & \bot & \bot \\
\bot & T & \bot & \bot & \bot \\
\bot & \bot & T & \bot & \bot \\
\bot & \bot & T & \bot & \bot \\
\bot & \bot & T & \bot & \bot \\
\vdots & & & &
\end{array}
$$

VERIFICA DI SISTEMI PARAMETRICI (11)

PROPRIETÀ LIVENESS (5)

- L'**assunzione di equità** nel token passing protocol dice che *un processo che può passare il token alla fine deve farlo*.
- Per codificare tale assunzione ad ogni processo è aggiunta una ***variabile booleana*** che è settata a *true* quando si ha la necessità di forzare un processo al passaggio del token.
- Tale variabile rimarrà *true* fin quando il token non è passato.
- Il nuovo *alfabeto* sarà l'insieme $\{\perp, T\} \times \{f, t\}$.

VERIFICA DI SISTEMI PARAMETRICI (12)

PROPRIETÀ LIVENESS (6)

La **relazione di transizione estesa** è la seguente:

$$\{((a,b),(a',b')) \ : \ b = \mathbf{f} \ \wedge \ (b' = \mathbf{t} \iff a' = T)\}^* \cdot \{((a,b),(a',b')) \ : \ b = \mathbf{f}\}$$

Essa rappresenta le due seguenti proprietà:

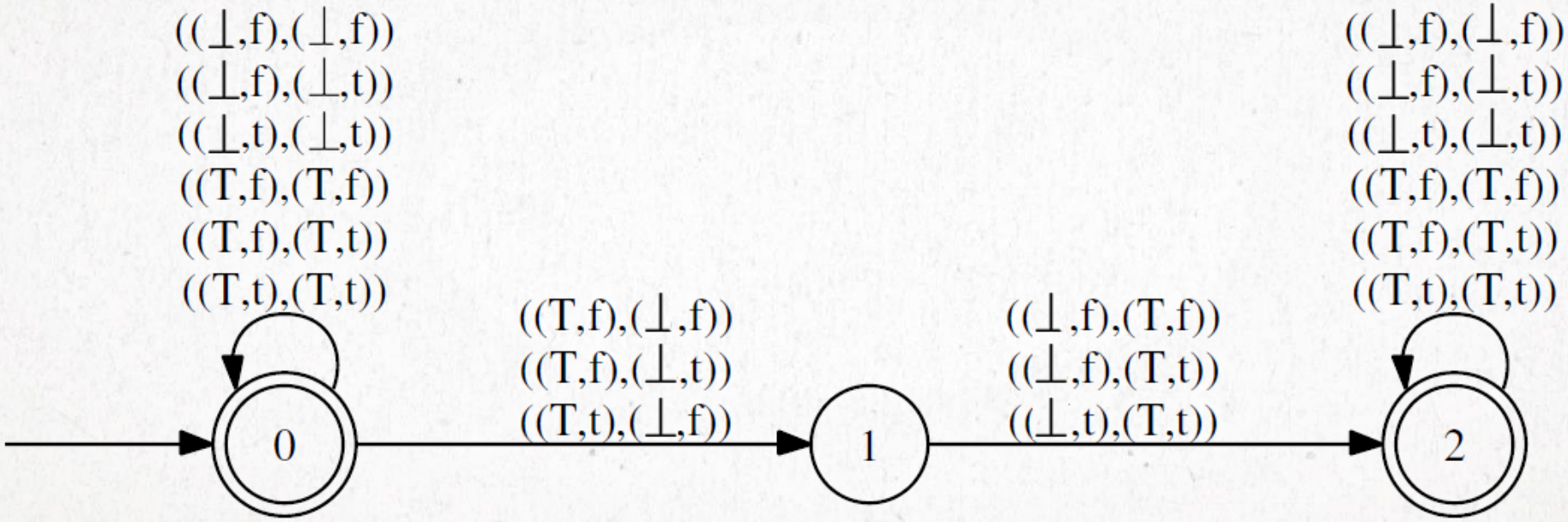
1. Quando un processo prende il token prima o poi (eventually) la variabile booleana sarà settata a *true* e ciò **assicura che esso passerà il token**;

2. Se la variabile booleana è settata a *true* allora in qualche punto successivo dell'esecuzione essa diventerà *false* e ciò **assicura che il token è stato passato**.

VERIFICA DI SISTEMI PARAMETRICI (13)

PROPRIETÀ LIVENESS (7)

La relazione di transizione estesa è rappresentata dal seguente trasduttore:

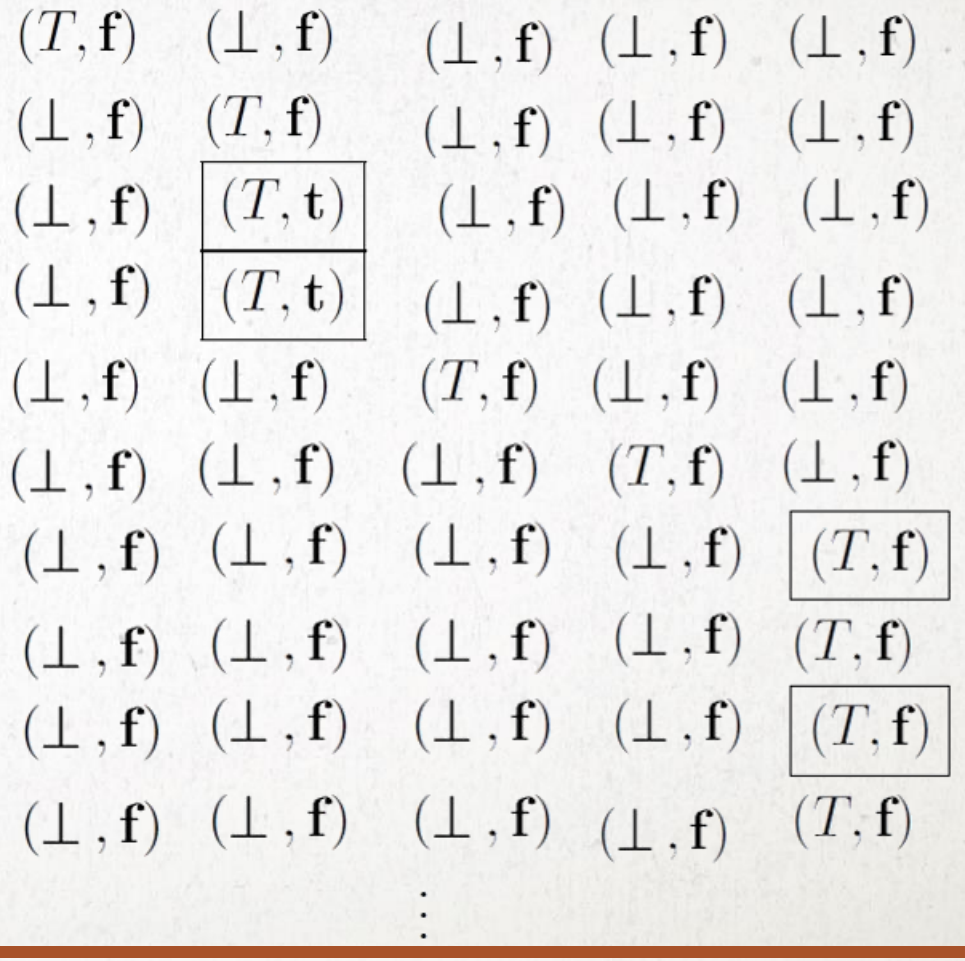


Possiamo notare che la variabile booleana può cambiare valore da un momento all'altro, in particolare se è *true* diviene *false* nel momento successivo se e solo se il processo ha passato il token.

VERIFICA DI SISTEMI PARAMETRICI (14)

PROPRIETÀ LIVENESS (8) – ESEMPIO ESECUZIONE

- Possiamo notare che per codificare la proprietà liveness « *il processo più a destra eventualmente avrà il token* » è stata aggiunta una **variabile observer**.
- L'insieme di configurazioni iniziali e la relazione di transizione sono stati modificati in modo tale che tale variabile sia *false* inizialmente e divenga *true* quando un processo passa il token.
- Nell'esecuzione considerata l'insieme di transizioni finali è composto dalle configurazioni in cui la variabile booleana del processo più a destra è *false*, ciò implica che tale variabile non diverrà mai *true* e quindi il processo più a destra non avrà mai il token. L'esecuzione è corretta perché abbiamo assunto che il sistema possa prendere transizioni idle, ma è **violata la proprietà liveness**.



STRUTTURA DELLA PRESENTAZIONE

- Introduzione (modello, proprietà, model checker, configurazioni,FSM, logica temporale, LTL, ecc)
- Model Checking e i sistemi parametrici
- Introduzione al Regular Model Checking
- Sistemi parametrici: token passing protocol
- Verifica di sistemi
- RMC per verifica di sistemi parametrici
- **CONCLUSIONI & RIFERIMENTI**

CONCLUSIONI

REGULAR MODEL CHECKING

- Nel corso del seminario è stato mostrato come **modellare** il *token passing protocol* utilizzando delle parole su un alfabeto finito;
- E' stato mostrato come **verificare** le *proprietà safety* mediante l'analisi di *reachability;*
- E' stato infine mostrato che sotto delle **assunzioni fairness** è possibile verificare le *proprietà liveness;*
- Il RMC è un framework grazie a cui è possibile verificare in maniera uniforme differenti tipologie di sistemi a stati infiniti;
- Il RMC riesce a lavorare solo se l'insieme degli stati e le relazioni di transizione sono **regolari**.

RIFERIMENTI

1. J.G. Henriksen, J. Jensen, M. Jørgensen, N. Klarlund, B. Paige,T. Rauhe, and A. Sandholm. Mona: Monadic second-order logic.
2. B. Boigelot and P. Wolper. Symbolic verification with periodic sets. Springer Verlag,1994.
3. L. Lamport. Proving the correctness of multiprocess programs.
4. Nilsson, M. 2005. Regular Model Checking. Uppsala. ISBN 91-554-6137-9 – pagina 17.
5. Nilsson, M. 2005. Regular Model Checking. Acta Universitatis Upsaliensis. Uppsala Dissertations from the Faculty of Science and Technology 60. 149 pp. Uppsala. ISBN 91-554-6137-9Capitolo 4.
6. M. O. Rabin and D. Scott. Finite automata and their decision problems. o Dexter C. Kozen. Automata and Computability. Springer-Verlag.
7. Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Julien d'Orso. Regular model checking made simple and efficient.
8. P.A. Abdulla, B. Jonsson, Marcus Nilsson, Julien d'Orso, and M. Saksena. Regular model checking for MSO + LTL.