UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Corso di laurea magistrale in Ingegneria dell'Automazione
A.A. 2021/2022

RETI DI PETRI

DOCENTE: PROF. PUPO

STUDENTE: IVONNE RIZZUTO

SOMMARIO

Reti di Petri	• Introduzione e definizione delle caratteristiche • Proprietà e rappresentazione
Reti di Petri Temporizzate	• Descrizione delle tipologie • Introduzione alle TTPN
Progetto di un magazzino a ricircolo	• Modellazione e simulazione • Discussione dei risultati

INTRODUZIONE ALLE RETI DI PETRI

DEFINIZIONI ED ELEMENTI STRUTTURALI

INTRODUZIONE ALLE RETI DI PETRI

Le reti di Petri costituiscono un formalismo grafico e matematico, introdotto da *Carl Adam Petri*, nel 1962, nella sua tesi di dottorato, per far fronte alla necessità di dover «descrivere, in modo preciso ed unitario, il maggior numero di fenomeni inerenti alla trasmissione ed all'elaborazione di informazioni».

Questo strumento, utilizzato per la modellazione di sistemi dinamici ad eventi discreti, consente, dunque, di descrivere globalmente un processo, seguendone l'evoluzione.

Rispetto agli automi a stati finiti, le reti di Petri permettono di modellare esplicitamente la concorrenza tra processi paralleli ed indipendenti tra loro.

Inoltre sono modulari e di conseguenza facilmente modificabili, oltre che idonee per mettere in risalto le caratteristiche di interesse del sistema, astraendo da quelle che non ne influenzano lo studio e le proprietà.

DEFINIZIONE DI UNA RETE DI PETRI

Una rete di Petri si può rappresentare a partire da un grafo orientato ed è definita come:

$$\mathbf{P/T} = < \mathbf{P}, \mathbf{T}, \mathbf{F}, \mathbf{W}, \mathbf{M_0} >$$

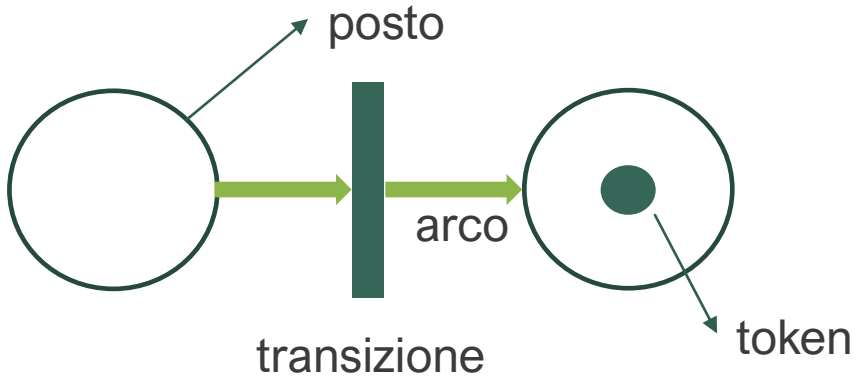
Gli elementi che la compongono sono i seguenti:

- L'insieme finito dei posti P, che contengono le risorse;
- L'insieme finito di transizioni T, che costituiscono gli eventi che possono verificarsi nel sistema;
- La relazione di flusso F, realizzata dagli archi che collegano i posti alle transizioni e viceversa;
- I token (o marche), che indicano le risorse;
- La funzione W, che assegna dei pesi, dal valore non negativo, agli archi;
- La marcatura iniziale M_0, cioè lo stato con cui si presenta inizialmente la rete.

STRUTTURA DI UNA RETE DI PETRI

Formalmente, gli elementi di una rete Posto/Transizione si possono descrivere come segue:

- $P = \{P_0, \dots, P_i\}$, con $i = 1, \dots, |P|$, dove $|P| < \infty$;
- $\mathrm{T} = \{T_0, \dots, T_j\}$, con $\mathrm{j} = 1, \dots, |T|$, dove $|T| < \infty$;
- $F = (P\ x\ T) \cup (T\ x\ P)$;
- $W : F \rightarrow N^*$, dove $N^* = N \backslash \{0\}$;



Inoltre, i posti e le transizioni non possono avere elementi in comune, ovvero:

- $P \cap T = 0$;
- $P \cup T \neq 0$;

DEFINIZIONE DI TRANSIZIONE

In una rete Posto/Transizione, lo stato viene modificato, con conseguente cambiamento della marcatura, quando si verifica lo scatto di una transizione.

Per *sparo* di una transizione si intende un evento, di tipo atomico ed istantaneo, che consuma dei token dai posti di ingresso e genera dei token nei posti di uscita.

Questo può verificarsi soltanto se la transizione è abilitata, cioè se i suoi posti in ingresso contengono un numero di token che sia sufficiente a consentire lo scatto.

Inoltre, l'evento di sparo di una transizione è non deterministico, poiché, in presenza di più transizioni abilitate, non è possibile prevedere quella che effettuerà lo scatto.

L'abilitazione di una transizione t in una marcatura M si definisce come:

$$M\,[\,t\,>\,(\,\forall\, posto \in Preset(t), M(posto) \geq W(posto, transizione)\,)$$

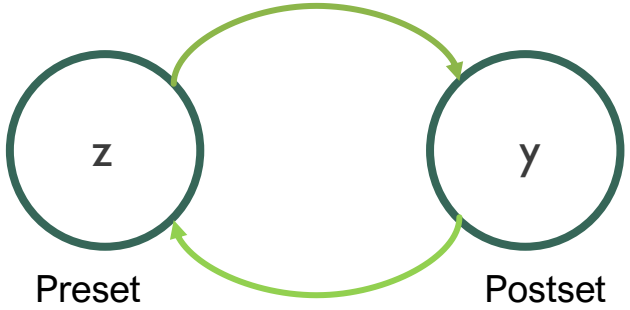
PRESET E POSTSET DI UNA RETE DI PETRI

Si definisce *preset* l'insieme dei token necessari all'attivazione della transizione, nei posti in ingresso:

$$Pre(y) = \{z \in X \mid < z, y > \in F\}, \text{dove } Pre : X \rightarrow 2^x$$

Con *postset*, invece, ci si riferisce all'insieme dei token generati nei posti in uscita:

$$Post(y) = \{z \in X \mid < y, z > \in F\}, \text{dove } Post : X \rightarrow 2^x$$



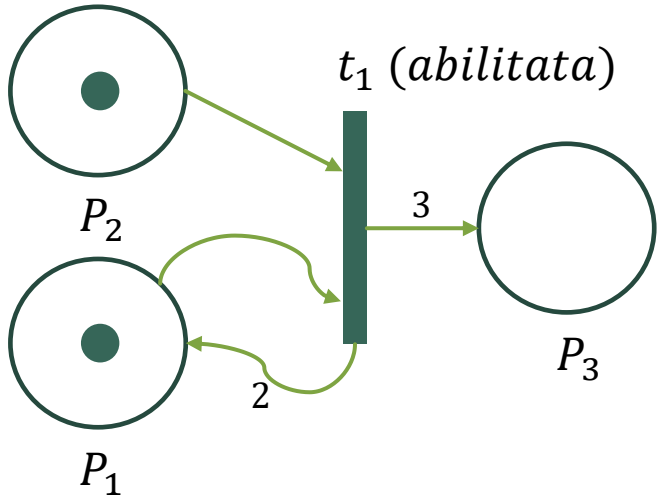
Il termine 2^x denota l'insieme delle parti di X, i cui elementi sono tutti i possibili sottoinsiemi di X, compreso l'insieme vuoto ed X stesso. Infatti $X = P \cup T$.

REGOLA DI SCATTO DI UNA TRANSIZIONE

Una transizione è *abilitata* se tutti i posti del suo preset contengono un numero di token, cioè di risorse, che sia maggiore o uguale al peso dell'arco che li connette alla transizione.

Dunque, se la transizione t viene scelta per lo scatto, la marcatura della rete si modifica in questo modo:

$$M\ [\ t >\ M'$$



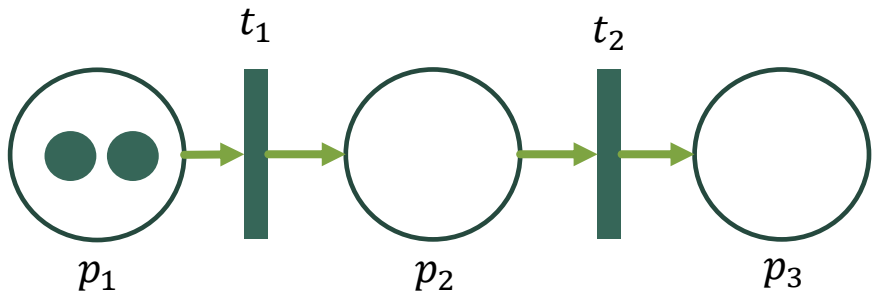
Le proprietà che vengono garantite sono le seguenti:

- $M'(p) = M(p) - W(p,t), \forall\ p\ \in Pre(t) - Post(t)$;
- $M'(p) = M(p) + W(p,t), \forall\ p\ \in Post(t) - Pre(t)$;
- $M'(p) = M(p) - W(p,t) + W(t,p), \forall\ p\ \in Pre(t) \cap Post(t)$;
- $M'(p) = M(p), \forall\ p \in P\ - (Pre(t) \cup Post(t))$;

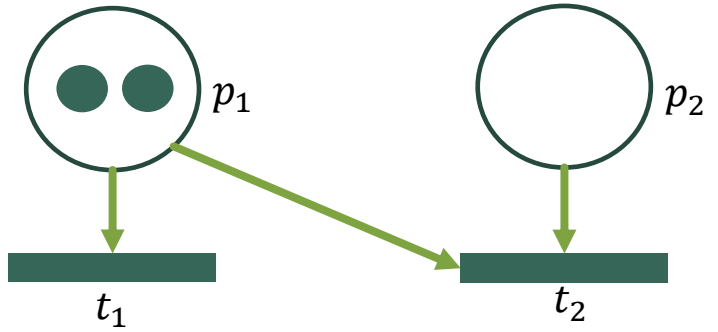
CONFIGURAZIONI DELLE RETI DI PETRI

Le reti di Petri possono presentarsi in configurazioni particolari, quali:

- *Sequenzialità*, se date due transizioni, con t_1 abilitata e t_2 non abilitata, lo scatto della prima abilita la seconda;



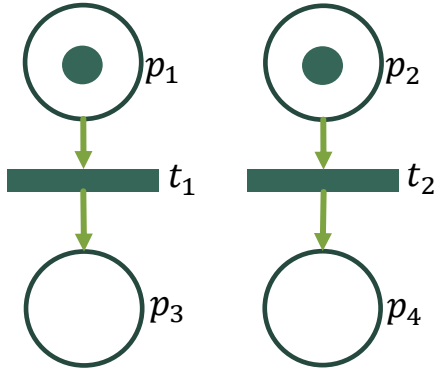
- *Conflitto strutturale*, se due transizioni hanno almeno un posto di ingresso in comune. Il conflitto si dice *effettivo* se sono entrambe abilitate ed il numero dei token nei rispettivi posti non soddisfa tutti i pesi degli archi che li collegano.



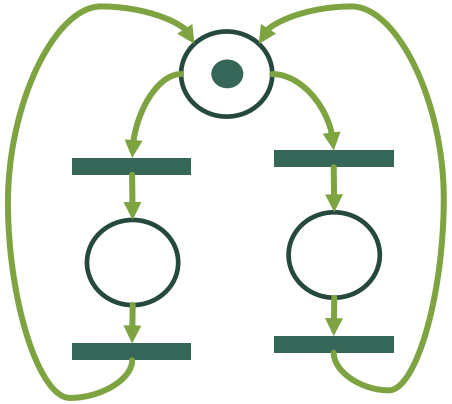
CONFIGURAZIONI DELLE RETI DI PETRI

Le reti di Petri possono presentarsi in configurazioni particolari, quali:

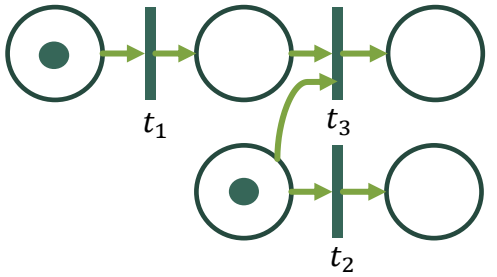
- *Concorrenza*, se date due transizioni t_1 e t_2, queste non condividono alcun posto di ingresso e perciò lo scatto di una non influenza quello dell'altra;



- *Mutua esclusione*, quando due transizioni t_1 e t_2 non saranno mai abilitate contemporaneamente;



- *Confusione*, se due transizioni t_1 e t_2 sono in concorrenza e l'ordine con cui verrà lo sparo influenzerà la risoluzione di un successivo conflitto (t_2 e t_3).



PROPRIETÀ DELLE RETI DI PETRI

DESCRIZIONE DELLE CARATTERISTICHE E DEI TIPI DI RAPPRESENTAZIONE

PROPRIETÀ DELLE RETI DI PETRI: RAGGIUNGIBILITÀ

Si consideri una generica rete di Petri P/T = (P, T, F, W, M_0), dove M_0 è la marcatura iniziale.

Una marcatura M' è *raggiungibile* a partire dalla marcatura M se esiste almeno una sequenza di transizioni la cui esecuzione permette di ottenere M' da M.

L'insieme di raggiungibilità $R(P/T, M)$ è definito come il più piccolo insieme di marcature tale che:

$$\begin{cases} M \in R(P/T, M) \\ (M \in R(P/T, M) \wedge (\exists\, t \in T, M[t > M') \Rightarrow M' \in R(P/T, M) \end{cases}$$

PROPRIETÀ DELLE RETI DI PETRI: LIMITATEZZA

Una rete P/T si dice *limitata* di ordine *k* se, in ogni marcatura raggiungibile, il numero dei token in un qualsiasi posto non supera il valore di *k*, ovvero:

$$\exists\, k \in N : \forall\, M' \in R(P/T, M)$$

$$M'(p) \leq k, \forall\ p \in P$$

Una rete non limitata può essere resa tale introducendo un posto complementare p' di p, in modo che la marcatura inziale $M[p] + M[p']$ costituisca il grado desiderato di limitatezza in p.

Un caso particolare di limitatezza è rappresentato dalla proprietà di safeness o binarietà.

Una rete P/T si definisce *binaria* se, in ogni suo posto, in ogni possibile marcatura, il numero dei token è zero oppure uno, $\forall\, M' \in R(P/T, M), \forall\, p \in P(M'(p) \leq 1)$.

PROPRIETÀ DELLE RETI DI PETRI: CONSERVATIVITÀ

Una rete P/T è *conservativa* se il numero totale dei token si mantiene costante in tutte le marcature raggiungibili.

Formalmente:

$$\sum_{p \in P} M'(p) = \sum_{p \in P} M(p),$$

con $\forall\, p \in P$ **e** $\forall\, M' \in R\,(P/T\,, M)$.

Quindi, una rete di questo tipo non perde e non guadagna token, durante la sua esecuzione.

Inoltre, per quanto riguarda gli elementi del vettore *w* dei pesi dei posti, la loro somma rimarrà costante su tutte le marcature raggiungibili.

PROPRIETÀ DELLE RETI DI PETRI:VITALITÀ

Una transizione t è caratterizzata da una *vitalità* indicata come:

- Di grado *zero*, se non può mai scattare in M; quindi una marcatura M è in deadlock se e solo se nessuna transizione è abilitata in essa;
- Di grado *uno*, se può scattare al massimo una volta soltanto;
- Di grado *due* se, fissato un intero $n > 0$, esiste almeno una sequenza di scatti in cui la transizione *t* scatta minimo *n* volte;
- Di grado *tre*, se scatta un numero infinito di volte;
- Di grado *quattro* se, considerata una qualsiasi marcatura raggiungibile, esiste almeno una sequenza di scatti che, a partire da tale marcatura, abiliti *t*; in questo caso la transizione *t* si dice *viva*.

Una rete di P/T è definita *viva* se tutte le sue transizioni sono vive, ossia di grado quattro.

PROPRIETÀ DELLE RETI DI PETRI DIPENDENTI DALLA MARCATURA

Una rete di Petri può essere definita:

- *Reversibile*, se è possibile risalire alla marcatura iniziale, qualsiasi sia la marcatura in corrispondenza di cui ci si trova, ossia $M_0 \in R(M), \forall\, M \in R(M_0)$, dove M_0 è una marcatura nota.
- *Home-state*, se è raggiungibile da ogni marcatura che risulti raggiungibile da M_0, per cui $M_i \in R(M), \forall\, M \in R(M_0)$.
- *Bloccante*, se esiste una marcatura $M' \in R(P/T)$ in cui non è abilitata alcuna transizione, perciò, una volta arrivata in M', la rete non potrà più evolvere, entrando quindi in deadlock.
- *Generica*, se non esistono restrizioni topologiche.

PROPRIETÀ DELLE RETI DI PETRI: ANALISI GRAFICA

Per analizzare le proprietà di una rete di P/T limitata si può ricorrere alla costruzione dell'albero di raggiungibilità, mentre nel caso in cui si tratti di una rete non limitata, si può utilizzare l'albero di copertura.

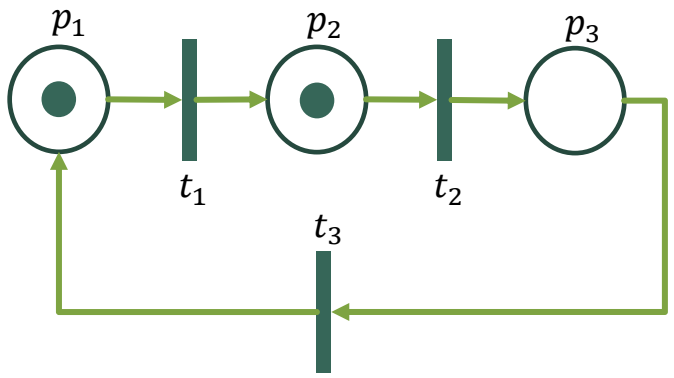
In ogni caso, tramite questo strumento, si andranno a rappresentare tutte le possibili marcature. Dunque, per ogni marcatura M' raggiunta, si aggiunge un nuovo nodo all'albero, collegandolo come figlio del nodo M e si etichetta l'arco con la transizione che ora determina il raggiungimento di M'.

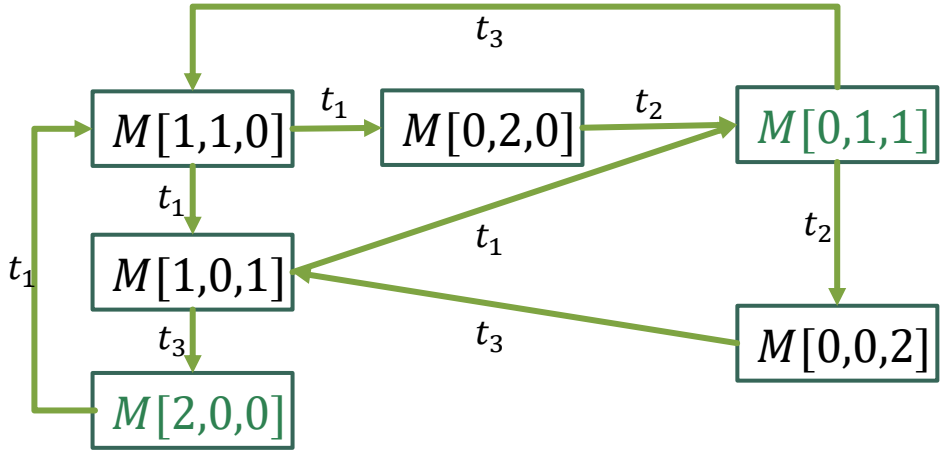
Se la marcatura in questione è già stata inserita, allora il nodo considerato verrà etichettato come "old", poiché la sua storia è già presente nell'albero.

Inoltre, se a partire da un nodo che rappresenta una marcatura, non esistono più transizioni abilitate, questo verrà definito "dead".

GRAFO DI RAGGIUNGIBILITÀ

Si consideri la rete di Petri, con marcatura iniziale $M_0 = M[1,1,0]$ ed il grafo di raggiungibilità associato:





Il grafo di raggiungibilità è un grafo associato ad una rete marcata $< N, M_0 >$, in cui ogni nodo è una marcatura raggiungibile ed ogni arco è una transizione.
In questa rappresentazione, ogni marcatura raggiungibile è uno stato ed appare una sola volta.
Le transizioni, invece, possono apparire più volte, poiché ciascuna compare un numero di volte pari alle marcature che la abilitano.

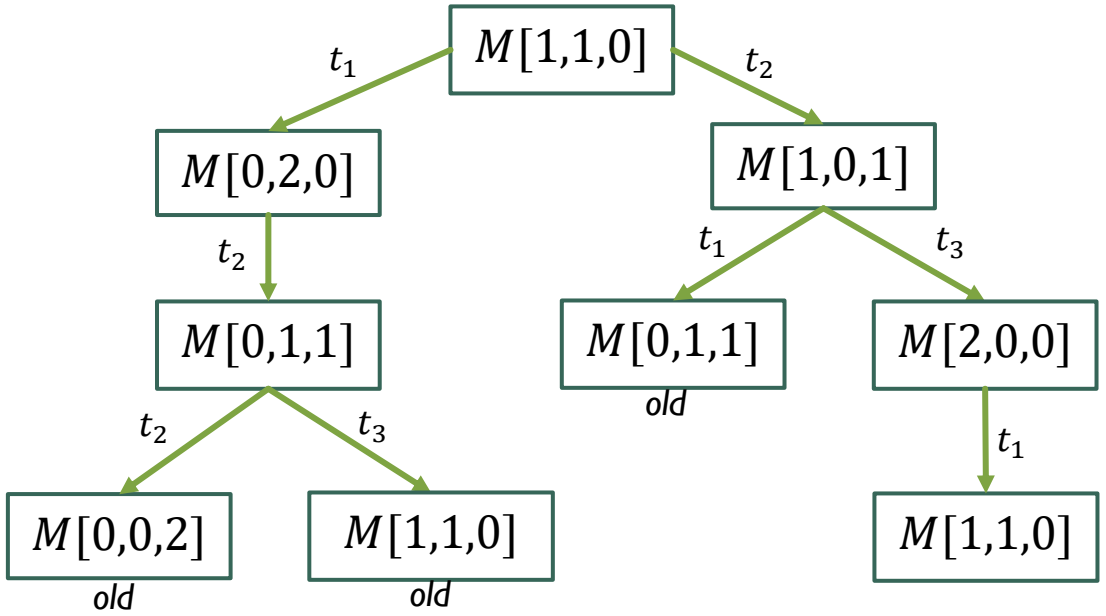
ALBERO DI RAGGIUNGIBILITÀ

Si consideri l'albero di raggiungibilità associato alla rete di Petri precedente:

La radice dell'albero costituisce la marcatura iniziale della rete P/T.

Rispetto al grafo di raggiungibilità, l'albero di raggiungibilità è una rappresentazione che permette di evidenziare le marcature duplicate, pur non esplicitando tutti i cicli.

Questo si costruisce soltanto se la rete è limitata, poiché, nel caso in cui il numero di nodi sia infinito, l'albero non verrebbe generato correttamente.



ALBERO DI COPERTURA

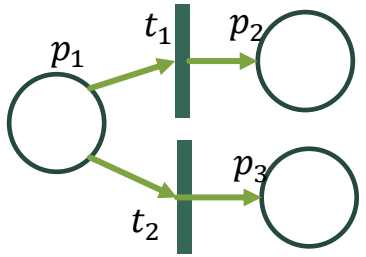
Se la rete P/T non è limitata e di conseguenza si hanno marcature duplicate, si può ricorrere all'albero di copertura.

Per definizione, una marcatura M'' copre una marcatura M' se $M''(p) \geq M'(p)$, per ogni posto p.
È analogo affermare che M' sia coperta da M''.
Per ovviare all'illimitatezza della rete si pone $M''(p) = w$ (infinito), in modo da tenere sotto controllo il numero dei nodi dell'albero.

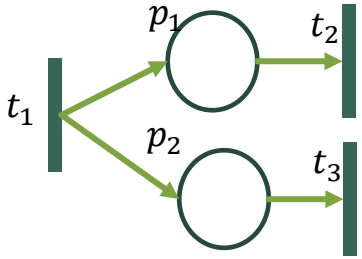
La costruzione dell'albero di copertura tramite l'algoritmo di *Karp-Miller* ha il vantaggio di terminare sempre in un numero finito di passi, anche se nella rete ci sono dei loop che incrementano delle marcature, causando il problema dell'accumulo dei token, che rende la rete non limitata.
Questo viene risolto inserendo il simbolo w, invece di incrementare il contatore del posto, in modo da riconoscere tutte le marcature duplicate generate dal ciclo, elaborando solamente le posizioni indicate da numeri naturali, poiché le w-transizioni non verranno processate.

SOTTOCLASSI DELLE RETI DI PETRI

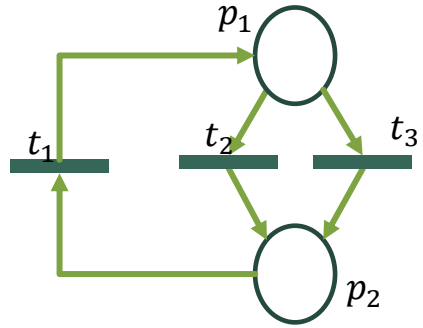
- *Macchina a stati* (State Machine), in cui ogni transizione t ha un solo posto in input ed un solo posto in output, per cui $|pre(t)| = |post(t)| = 1, \forall\, t \in T$;



- *Grafo marcato* (Marked-graph), in cui ogni posto p ha esattamente una transizione in input ed una transizione in output, per cui $|pre(p)| = |post(p)| = 1, \forall\, p \in P$;



- *Rete a scelta libera* (Free-choice net), in cui ogni posto alimenta una sola transizione oppure è l'unico posto in input ad un gruppo di transizioni, perciò $|post(p)| = 1, \forall\, p \in P$ or $pre\big(post(p)\big) = \{p\}$, quindi ogni posto che ha due o più transizioni è il solo posto in input per ciascuna;



ANALISI MATRICIALE DI UNA RETE DI PETRI

Una rete di Petri P/T, essendo un grafo orientato e pesato, può essere rappresentata algebricamente, tramite una matrice di incidenza o adiacenza, date:

- Matrice di ingresso I: $|P|x|T|$;
- Matrice di uscita O: $|P|x|T|$;
- Matrice di incidenza C: $|P|x|T|$;
- Vettore marcatura m: $|P|$, che rappresenta il numero di token presenti nel posto p_i;

Una transizione, quindi, è abilitata se nella marcatura corrente, cioè il contenuto del vettore m, è presente un numero di token sufficiente allo sparo, ossia $\forall i: p_i \in P, m[i] > I_{i,j}$.

L'effetto dello scatto della transizione t_j può essere espresso come $m' = m - I_{.,j} + O_{.,j}$ e memorizzato, poi, nella matrice $C = O - I$, in cui ogni elemento rappresenta l'incremento oppure il decremento del numero di token nel posto p_i.

ANALISI MATRICIALE: EQUAZIONE DI STATO

A partire dalla rappresentazione matriciale di una rete di Petri si ottiene la seguente *equazione di stato*:

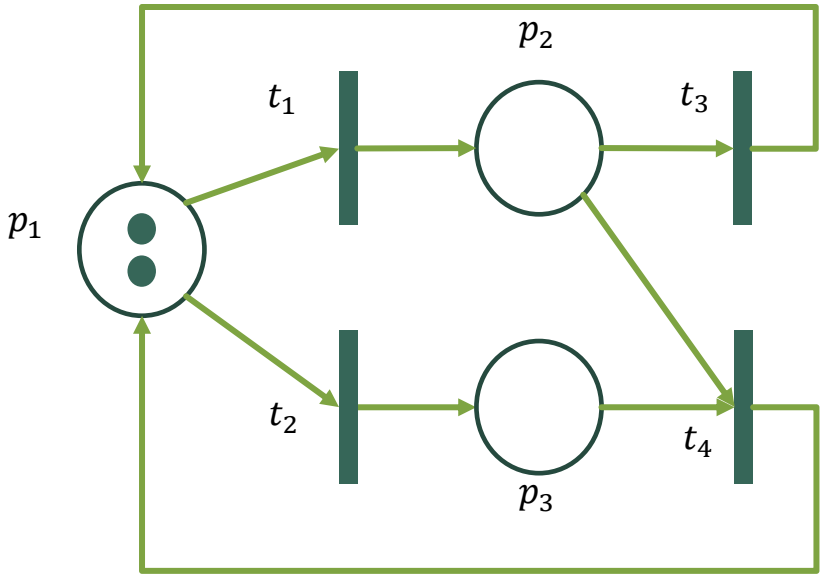
$$\boldsymbol{m' = m + C * s}$$

Dove s è un vettore colonna, denominato vettore delle occorrenze ed associato ad una sequenza di scatti, in cui l'elemento s_i rappresenta il numero di volte in cui la transizione t_i si verifica nella sequenza di scatti s.

Questa *equazione fondamentale* descrive la dinamica della rete e permette di calcolarne la marcatura successiva, se sono noti quella precedente e l'evento, cioè lo scatto della transizione che si è verificato.

Inoltre evidenzia il carattere intrinseco di linearità che si ha nel calcolo dell'evoluzione della rete stessa.

ESEMPIO DI ANALISI MATRICIALE DI UNA RETE DI PETRI



L'equazione fondamentale di questa rete è:

m		C				s		m′
2	+	-I	-I	I	I	2	=	I
0		I	0	-I	-I	I		0
0		0	I	0	-I	I		0
						I		

$\mathbf{I}$
$p_i \rightarrow t_j$

	t_1	t_2	t_3	t_4
p_1	I	I	0	0
p_2	0	0	I	I
p_3	0	0	0	I

$\mathbf{O}$
$t_j \rightarrow p_i$

	t_1	t_2	t_3	t_4
p_1	0	0	I	I
p_2	I	0	0	0
p_3	0	I	0	0

$\boldsymbol{C = O - I}$
Effetto netto

	t_1	t_2	t_3	t_4
p_1	-I	-I	I	I
p_2	I	0	-I	-I
p_3	0	I	0	-I

INVARIANTI DI UNA RETE DI PETRI

La rappresentazione matriciale di una rete P/T consente di identificare alcune proprietà che non mutano a seconda della marcatura raggiungibile, ovvero gli invarianti.

Questi si distinguono in:

- P-Invarianti, se riferiti ai posti;
- T-Invarianti, se riferiti alle transizioni;

Per *P-Invariante* si intende la somma pesata ottenuta dal prodotto del numero di token che, in una qualsiasi marcatura raggiungibile, si trovano in ogni posto, in base al peso che vi è associato.

Con *T-Invariante*, invece, ci si riferisce ad una sequenza di scatti ciclica, in grado di riportare la rete nella marcatura di partenza, lo stato "home".

ESTENSIONI DELLE RETI DI PETRI

Per aumentare il potere di modellazione delle reti di Petri sono state introdotte alcune estensioni, quali:

- Rete P/T con *archi inibitori* o negatori, in modo che, anche se è presente un token nel posto di ingresso collegato ad una transizione, questa risulterà comunque disabilitata a causa del tipo di arco sopracitato (realizzando una sorta di priorità tra le varie transizioni);

- Rete di Petri *colorata*, per poter distinguere i vari token in diversi tipi, tramite la specifica del «colorset», ottenendo così una maggiore espressività della rete stessa;

- Rete di Petri *estesa* con il *tempo,* dando luogo a diversi tipi di reti temporizzate, ossia le Transition-Timed Petri Nets e le Placed-Timed Petri Nets;

RETI DI PETRI TEMPORIZZATE

DESCRIZIONE DEI TIPI E DELLE CARATTERISTICHE

TIPI DI RETI DI PETRI TEMPORIZZATE

- Una rete di Petri di tipo TTPN assegna i tempi di scatto τ_i alle transizioni $t_i \in T$, per cui

$$\tau : T \to R$$

Dunque, le transizioni sono viste come delle attività, ciascuna delle quali inizia non appena la transizione corrispondente viene abilitata e termina quando si verifica lo scatto;

- Una rete PTPN, invece, assegna i tempi di conservazione τ_i ai posti $p_i \in P$, ovvero

$$\tau : P \to R$$

In questo caso i token che arrivano in un posto vi rimangono per un certo tempo di conservazione, prima di essere utilizzati per lo scatto della rispettiva transizione;

Si precisa che un modello di reti P/T con i tempi nei posti (PTPN) può essere sempre ricondotto ad un modello con i tempi nelle transizioni (TTPN).

DEFINIZIONE DI TIMED PETRI NET

Formalmente, una rete di Petri temporizzata si può descrivere come:

$$\boldsymbol{TPN = (P, T, F, W, \tau, M)}$$

Dove:

- $P = \{p_1, p_2, \dots, p_{nP}\}$ è un insieme finito di posti;
- $T = \{\tau_1, \tau_2, \dots, \tau_{nT}\}$ è un insieme finito di transizioni;
- $F \subseteq$ (P x T) U (T x P) è un insieme finito di archi, tra posti e transizioni;
- $W: F \to N$ è una funzione $w(f)$ che assegna dei pesi agli elementi $f \in F$ che denotano la molteplicità degli archi unitari tra gli elementi connessi;
- $\tau: T \to R$, che assegna ritardi di sparo τ_i agli elementi $t_i \in T$;
- $M: P \to N^{+0}$, cioè la marcatura degli elementi $p_i \in P$;

RITARDI DI SPARO

Nelle TPN si possono distinguere tre casi di *firing delays*, ossia quando:

- $\tau_i = 0$, allora si tratta di una transizione immediata, che spara consumando un tempo nullo;
- $\tau_i \in R$, cioè assume un valore deterministico (se *Deterministic Timed Transition* Petri Nets);
- τ_i è un'istanza di una variabile casuale (nelle *Stochastic Timed Transition* Petri Nets);

Per τ_i si intende la durata della transizione, ovvero il tempo che impiega a scattare.

Nel caso in cui T contenga soltanto transizioni temporizzate e stocastiche, il cui ritardo di sparo è una variabile casuale con distribuzione esponenziale, la rete TPN appartiene alla classe delle *Stochastic Petri Nets*.

Un'altra classe è quella delle *Generalized Petri Nets*, che consentono l'utilizzo di una combinazione di transizioni senza tempo o immediate (che hanno sempre precedenza sulle altre e quindi scattano per prime) e di transizioni temporizzate stocastiche.

TIPOLOGIE DI POLITICHE DI SPARO

Quando non sussistono più transizioni istantanee abilitate, allora possono scattare le transizioni abilitate temporizzate, secondo due differenti politiche di sparo.

Dunque, tenendo conto che spara per prima una transizione con ritardo di sparo minimo, si potrà adottare una politica di sparo di tipo:

- ***Race*** o *pre-emption*, ereditata dalle reti P/T classiche, che stabilisce che la transizione con ritardo di sparo minimo produce conseguenze sulle transizioni in conflitto, perciò la transizione candidata ad eseguire lo sparo è quella con il quanto di tempo minore (a parità di tempo, la scelta non è deterministica);
- **Preselection**, secondo cui si sceglie a priori una transizione temporizzata tra quelle in conflitto e se ne assicura lo sparo;

POLITICHE DI SPARO: RACE

Si precisa che il tempo che trascorre tra l'abilitazione e lo scatto di una transizione è casuale ed in questo intervallo i token risiedono nel preset.

Trattandosi di pre-emption, la transizione che sarà caratterizzata dal valore minimo di ritardo potrà pre-emptare quella attualmente abilitata.

Questo vuol dire che una transizione può perdere l'abilitazione, perciò si avrà necessità di assegnare un nuovo valore al ritardo di sparo, per l'abilitazione successiva.

Il calcolo di questo valore dipenderà dalla politica adottata, ovvero:

- *Con memoria*, se è possibile conservare l'attività svolta dalla transizione fino al momento in cui è stata interrotta, in modo tale da utilizzare il tempo residuo come valore del ritardo di sparo alla prossima abilitazione;
- Senza memoria o *re-sampling*, se è necessario calcolare ex-novo il valore del ritardo di sparo, ad ogni nuova abilitazione della transizione in questione;

POLITICHE DI SPARO: PRESELECTION

Si precisa che, seguendo la politica di preselection, una transizione temporizzata esegue lo sparo seguendo tre fasi, quali:

1) *Start firing*, durante la quale i token vengono prelevati anticipatamente dal preset;
2) *Firing in progress*, in cui i token vengono congelati per tutta la durata del tempo di sparo (rimanendo "invisibili" per le abilitazioni delle altre transizioni);
3) *End firing*, durante cui i token vengono rilasciati nei posti di output, cioè vengono generati in uscita;

Questa regola di esecuzione che definisce lo scatto di una transizione in tre tempi è detta *scatto tri-fase* della politica di sparo preselettiva o non pre-emptiva.

In questo caso, una volta che l'attività di una transizione è stata avviata, non potrà più essere interrotta sino al suo completamento.

POLITICHE DI SPARO: CASI PARTICOLARI

Se una transizione t è abilitata k volte nello stesso istante di tempo τ si possono adottare delle politiche differenti, quali:

- Semantica *Single Server*, secondo cui la transizione t può sparare una sola abilitazione per volta (perché le abilitazioni devono essere sparate in sequenza);
- Semantica *Infinite Server*, che specifica che la transizione t può sparare un qualsiasi numero di abilitazioni contemporaneamente (poiché i k spari iniziano simultaneamente, ciascuno con un valore di ritardo differente, che è ottenuto a partire dalla distribuzione di probabilità della transizione);
- Semantica *H-Server*, che prevede che la transizione t possa sparare al più h abilitazioni per volta, trattandosi di un approccio intermedio tra i primi due sopradescritti.

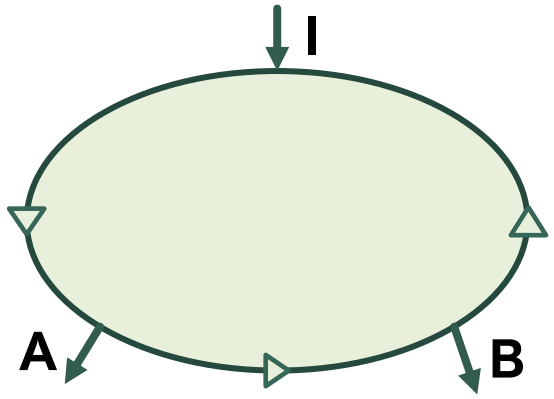
PROGETTO

MODELLAZIONE E SIMULAZIONE DI UN MAGAZZINO CIRCOLARE TRAMITE UNA RETE DI PETRI TEMPORIZZATA

INTRODUZIONE AL PROGETTO

L'obiettivo di questo elaborato è quello di descrivere e modellare il sistema di interesse tramite il formalismo definito dalle reti di Petri.

A questo scopo si è scelto di rappresentare un *magazzino a ricircolo*, costituito da una stazione di ingresso I e due stazioni di uscita, A e B, in cui vengono spostati dei pallet, carichi di merci, tramite dei trasportatori, detti conveyor, in maniera automatizzata.



Si specifica che il carico può entrare nel magazzino soltanto se è disponibile un pallet vuoto nella stazione di ingresso e che i trasportatori dei pallet continueranno a circolare, in senso antiorario, a partire dalla stazione I, a prescindere dalla presenza o meno delle merci sul pallet.

Nel caso di studio scelto, si supporrà, per semplicità, che esista un solo pallet.

FUNZIONAMENTO DEL MAGAZZINO

Quando il carrello trasportatore si trova presso la stazione di ingresso I si avrà che:

- Se il pallet che viene trasportato dal conveyor è vuoto ed è disponibile un nuovo carico nella stazione di ingresso, questo verrà prelevato e condotto verso la stazione A;
- Se il pallet è vuoto e non è disponibile un nuovo carico in I, il conveyor si dirige verso A;
- Se il pallet è carico allora verrà trasportato direttamente verso la stazione A, senza che vi vengano caricate altre merci;

Quando il conveyor giunge presso una stazione di uscita, che sia A oppure B, si presenta uno di questi scenari:

- Se il pallet è carico, bisogna verificare se è sopraggiunta una richiesta di merci dalla stazione di uscita e nel caso, effettuare la consegna, per poi proseguire, vuoto, verso la stazione successiva;
- Se il pallet è vuoto, non verrà interrogata nessuna delle due stazioni di uscita, ma il conveyor si muoverà direttamente verso la stazione successiva;

MODELLAZIONE DEL SISTEMA

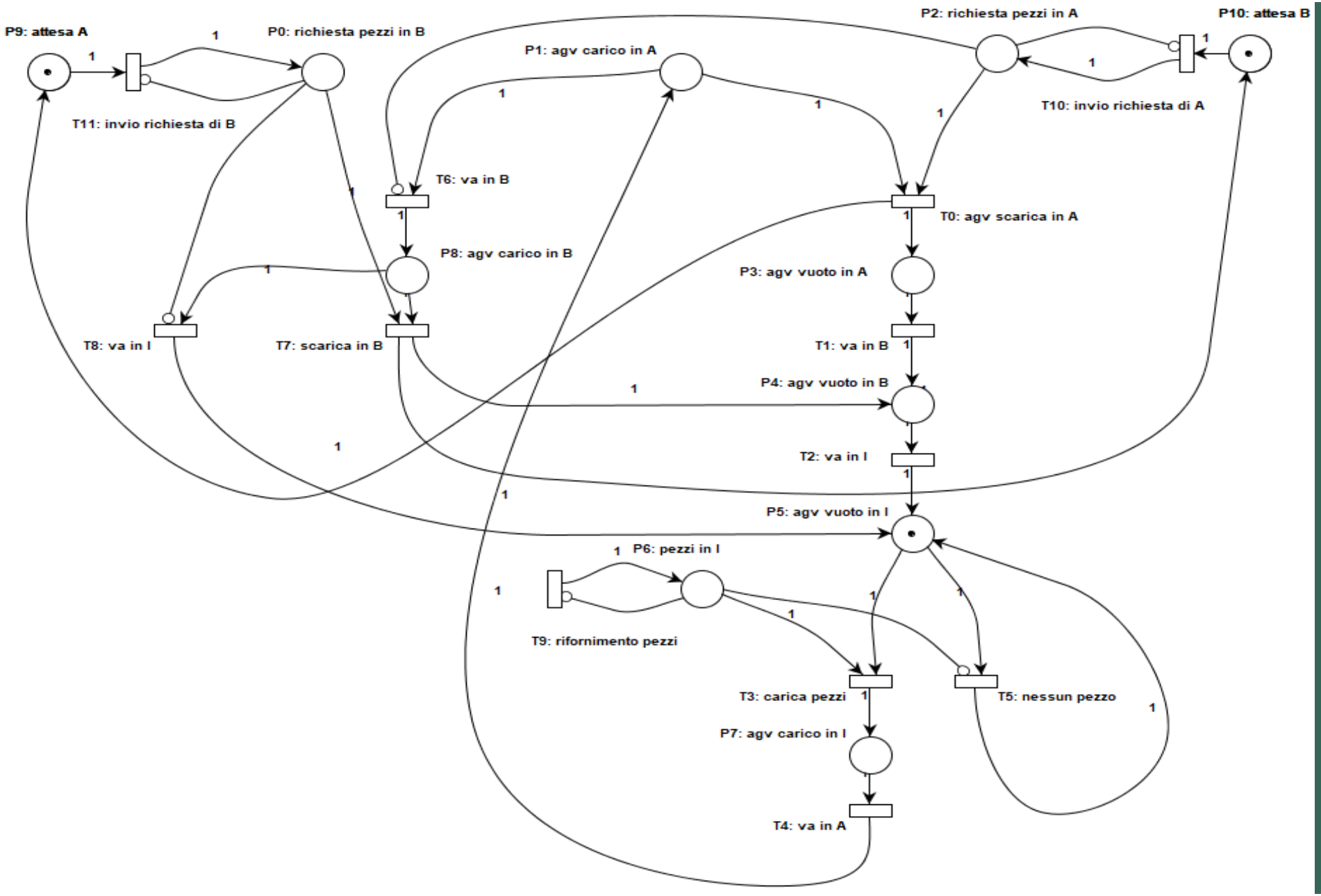
Per la simulazione e l'analisi della rete di Petri che rappresenta il sistema in esame è stato utilizzato il software **PIPE editor**.

La rete di Petri ottenuta, quindi, descrive il comportamento del magazzino a ricircolo, tenendo conto che ci sia un pallet soltanto ad essere trasportato da una stazione alla successiva.

Inoltre, la richiesta di merci da parte delle stazioni di uscita viene descritta con dei posti esogeni (non determinati all'interno del modello) e lo stato del pallet, vuoto o con carico, viene indicato, rispettivamente, dall'assenza o presenza del token.

In conclusione, il modello sarà costituito da 11 posti, 12 transizioni e 34 archi (anche inibitori).

La costruzione del grafo di raggiungibilità associato alla rete ha impiegato, nel complesso, 0.54 secondi.



MAGAZZINO CIRCOLARE

modellato tramite una rete di Petri P/T con PIPE Editor

RISULTATI DELLA SIMULAZIONE DELLA RETE SU PIPE EDITOR

Place	Average number of tokens	95% confidence interval (+/-)
P0: richiesta pezzi in B	0,78218	0,09315
P1: agv carico in A	0,16832	0,02126
P10: attesa B	1	0
P10: attesa B	0,28713	0,10867
P2: richiesta pezzi in A	0,34653	0,06677
P3: agv vuoto in A	0,07921	0,01227
P4: agv vuoto in B	0,19802	0,00951
P5: agv vuoto in I	0,20792	0,02904
P6: pezzi in I	0,77228	0,13809
P7: agv carico in I	0,24752	0,0469
P8: agv carico in B	0,09901	0,02632
P9: attesa A	1	0
P9: attesa A	0,58416	0,14656

Dai risultati si evince che la rete di Petri è estesa, ma limitata e senza deadlock.

State Machine	false
Marked Graph	false
Free Choice Net	false
Extended Free Choice Net	false
Simple Net	true
Extended Simple Net	true

Bounded	true
Safe	false
Deadlock	false

ANALISI DEGLI INVARIANTI

A riprova del fatto che la rete sia limitata e viva si riportano i valori dei T-invarianti, che risultano positivi:

T-Invariants

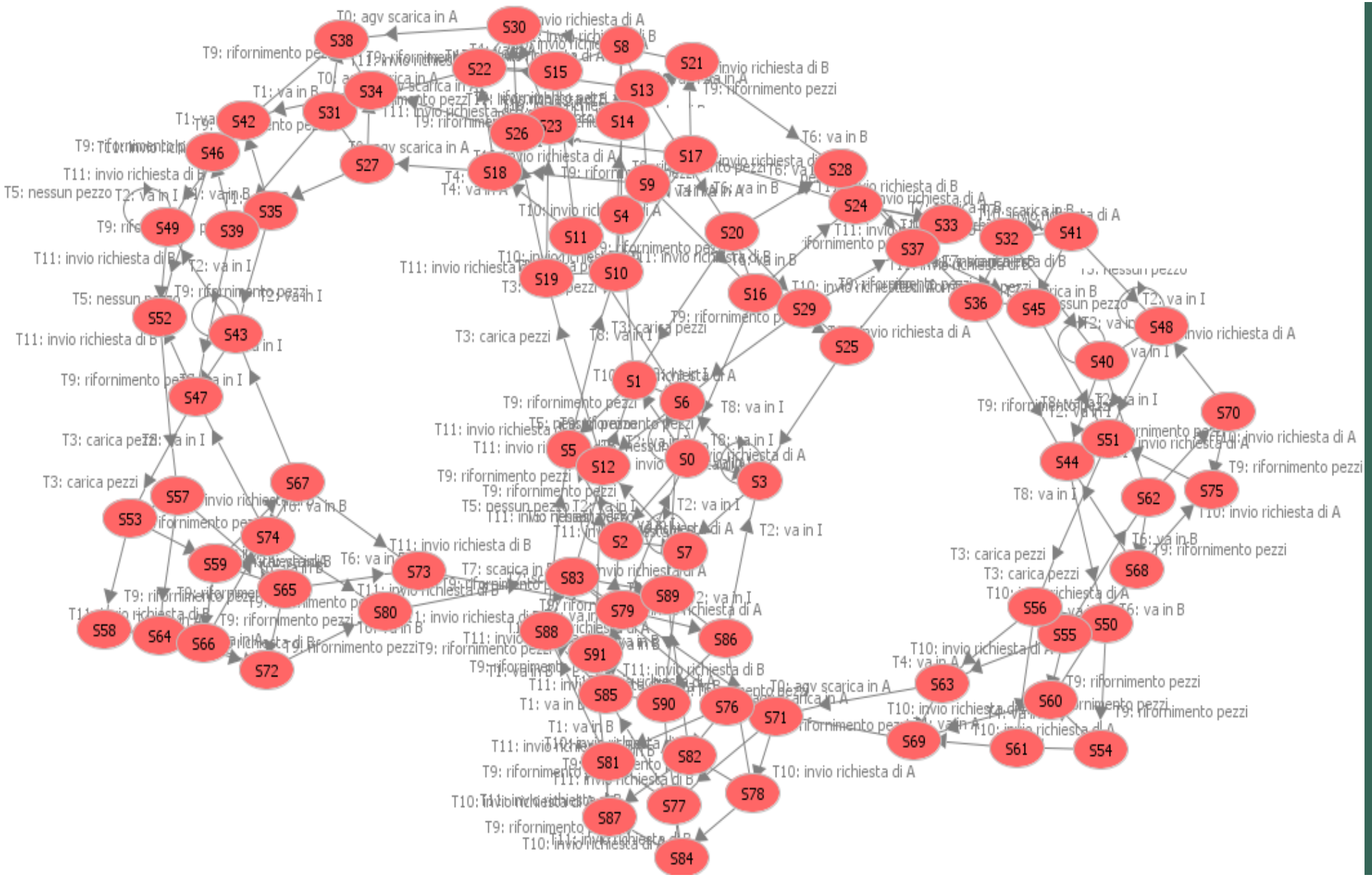
T0: agv scarica in A	T1: va in B	T10: invio richiesta di A	T11: invio richiesta di B	T2: va in I	T3: carica pezzi	T4: va in A	T5: nessun pezzo	T6: va in B	T7: scarica in B	T8: va in I	T9: rifornimento pezzi
1	1	1	1	2	2	2	0	1	1	0	2
0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	1	0	1	0	1	1

Per quanto riguarda i P-invarianti, cioè le proprietà della rete riferite ai posti, si sono ottenuti i seguenti risultati:

P-Invariants

P0: richiesta pezzi in B	P1: agv carico in A	P10: attesa B	P10: attesa B	P2: richiesta pezzi in A	P3: agv vuoto in A	P4: agv vuoto in B	P5: agv vuoto in I	P6: pezzi in I	P7: agv carico in I	P8: agv carico in B	P9: attesa A	P9: attesa A
0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	1
0	1	0	0	0	1	1	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0

GRAFO DI RAGGIUNGIBILITÀ



ANALISI DEL SISTEMA SU TPN DESIGNER

Si è scelto di modellare e simulare il sistema anche su **TPN Designer**, un ambiente integrato di progettazione grafica e simulazione di reti TPN.

La rete di Petri modellata su PIPE Editor è stata quindi ricreata nella sezione del software dedita alla costruzione di modelli di reti, il sotto-ambiente **TPN CAD**.

Volendo poi esportare questa rete in una struttura che possa essere analizzata tramite il software Uppaal, si è scelto di trasformare la rete in una rete di Petri con Tempo.

Questo è stato possibile modificando la funzione caratteristica di ogni transizione.
La temporizzazione di ciascuna, infatti, non sarà più immediata, ma di tipo Merlin & Faber.

Dopo aver fatto queste modifiche, si è potuti passare al sotto-ambiente **TPN Engine**, per compilare e simulare la rete e poi tradurla in codice eseguibile con Uppaal.

RETE DI PETRI CON TEMPO DI MERLIN E FABER

È un tipo di rete che si definisce "time", anziché "timed", poiché non si tratta di una rete di Petri temporizzata, ma di una rete di Petri con tempo.

Questo formalismo prende il nome da *Merlin*, che è lo studente che lo ha proposto nella sua tesi di dottorato e *Faber*, che è il professore che lo ha seguito.

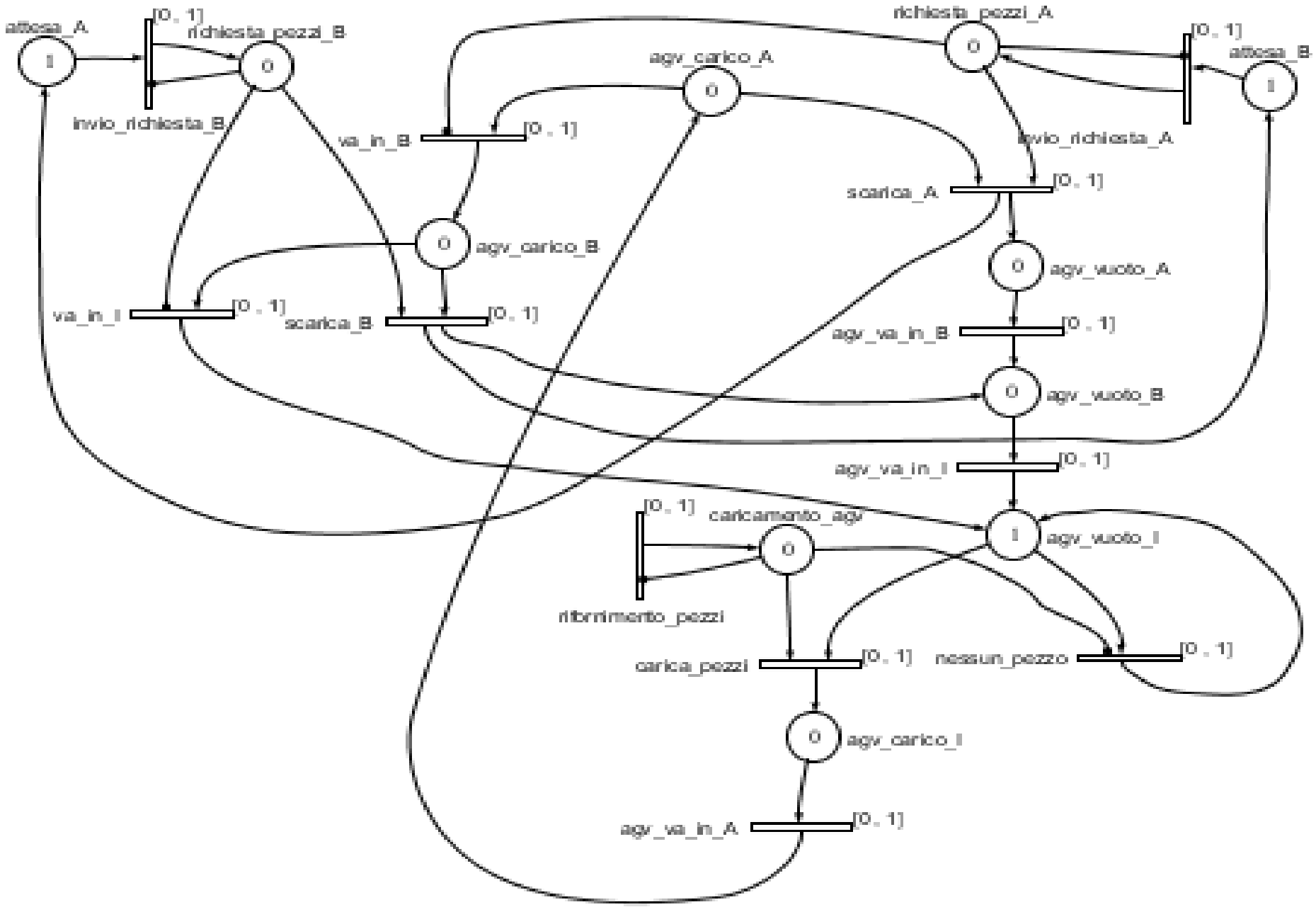
In una *Time Petri Net*, ogni transizione è caratterizzata da un *intervallo di tempo*, in cui ogni istante che lo compone potrebbe essere, potenzialmente, quello in cui la transizione sparerà.

Allora, per *finestra di sparo* di una transizione si intende l'intervallo denso $[a, b]$ in cui:

- "a" è detto Earliest Firing Time (EFT);
- "b" è detto Latest Firing Time (LFT);

Inoltre deve valere la condizione $0 \leq a \leq b$.

Ciò vuol dire che una transizione abilitata non può più sparare se il tempo corrente eccede la sua finestra di transizione. Quindi, qualora questa risultasse costantemente abilitata, dovrà necessariamente sparare per evitare il deadlock dell'intera rete.



MAGAZZINO CIRCOLARE

modellato tramite una rete Time Petri Net di Merlin & Faber con TPN Designer

ANALISI DEL SISTEMA SU UPPAAL

Il sistema trattato è stato analizzato anche tramite **Uppaal**, un software utilizzato per la modellazione, la validazione e la verifica di sistemi a tempo reale, formalizzati come reti di automi temporizzati.

Questo software, che supporta la sintassi C, è costituito da tre sotto-ambienti, ovvero l'Editor grafico (la cui GUI è realizzata in Java), il Simulator ed il Verifier (o model checker).

Un automa temporizzato, su Uppaal, è caratterizzato da:

- Tempo denso, attraverso variabili di clock che possono essere resettate e che avanzano allo stesso rate con cui scorre il tempo globale della rete;
- Dichiarazioni globali di variabili intere, booleane e di array;
- Sincronizzazione del tipo "rendezvous" (CSP), basata su canali unicast e/o broadcast e tramite dati globali;
- Processi (automi) tramite template istanziabili, parametrici e con eventuali dichiarazioni locali;
- Il sistema è visto come una composizione parallela di istanze di Timed Automata;

TIMED AUTOMATA

Un *automa temporizzato* è un formalismo utilizzato per la modellazione e la verifica esaustiva (model checking) di sistemi concorrenti e tempo-dipendenti, ottenuto sovrapponendo ad un automa la struttura dell'orologio (clock) che temporizza i suoi eventi. In questo modo si crea una sequenza di stati oppure eventi del sistema (execution trace) associata ad una specifica temporizzazione.

Formalmente, un automa temporizzato si descrive come $G = (L, I_0, C, A, E, I)$, dove:

- L è un insieme di locazioni (gli stati) ed $I_0 \in L$ è la locazione iniziale;
- C è l'insieme dei clock utilizzati;
- A è un insieme di azioni (operazioni di input/output nel sistema);
- $E \subseteq L \; x \; B(C) \; x \; A \; x \; 2^x \; x \; L$, che è un insieme di archi (transizioni) tra coppie di locazioni;
- $I\colon \; L \rightarrow B(C)$ è la funzione che assegna gli invarianti alle locazioni;
- $B(C)$ rappresenta le congiunzioni che si possono formare tramite confronti fra i clock;

TRADUZIONE DA TIME PETRI NET A UPPAAL

Per effettuare questo passaggio da un software all'altro, si dovranno considerare le seguenti strutture:

- Matrice di *Backward*, definita come $B: |T| x\ Pre \rightarrow Info$;
- Matrice di *Forward*, ovvero $F: |T| xPost \rightarrow Info$;
- Vettore di *Marking*, cioè $M: |P| \rightarrow N$;
- Matrice di *Intervals*, indicata come $I: |T| x\ 2\ \rightarrow N\ \cup \{-1\}$, che associa ad ogni transizione il suo intervallo di sparo;
- Struttura *Info*, contenente un indice di un posto in M ed il peso dell'arco entrante/uscente alla/dalla transizione, a seconda che si tratti di B o di F.

Si precisa che $|T|$ e $|P|$ si riferiscono, rispettivamente, alla cardinalità dell'insieme delle transizioni e dell'insieme dei posti, presenti nel modello descritto su TPN Designer.

Dunque, per la traduzione in Uppaal, ad ogni transizione si dovrà associare un template *Process* e poi istanziare un processo per ogni transizione presente.

ELEMENTI STRUTTURALI DEL MODELLO

Locazioni (stati)

- const int agv_carico_A = 0;
- const int richiesta_pezzi_B = 1;
- const int richiesta_pezzi_A = 2;
- const int attesa_B = 3;
- const int attesa_A = 4;
- const int agv_vuoto_A = 5;
- const int agv_vuoto_B = 6;
- const int agv_vuoto_I = 7;
- const int agv_carico_I = 8;
- const int caricamento_agv = 9;
- const int agv_carico_B = 10;

Transizioni

- const int invio_richiesta_B = 0;
- const int invio_richiesta_A = 1;
- const int va_in_B = 2;
- const int scarica_A = 3;
- const int agv_va_in_B = 4;
- const int agv_va_in_I = 5;
- const int agv_va_in_A = 6;
- const int carica_pezzi = 7;
- const int nessun_pezzo = 8;
- const int rifornimento_pezzi = 9;
- const int scarica_B = 10;
- const int va_in_I = 11;

ELEMENTI STRUTTURALI DELL'AUTOMA

Transizioni per cui istanziare i processi

- e_invio_richiesta_B = Transition(invio_richiesta_B, dummy_task);
- e_invio_richiesta_A = Transition(invio_richiesta_A, dummy_task);
- e_va_in_B = Transition(va_in_B, dummy_task);
- e_scarica_A = Transition(scarica_A, dummy_task);
- e_agv_va_in_B = Transition(agv_va_in_B, dummy_task);
- e_agv_va_in_I = Transition(agv_va_in_I, dummy_task);
- e_agv_va_in_A = Transition(agv_va_in_A, dummy_task);
- e_carica_pezzi = Transition(carica_pezzi, dummy_task);
- e_nessun_pezzo = Transition(nessun_pezzo, dummy_task);
- e_rifornimento_pezzi = Transition(rifornimento_pezzi, dummy_task);
- e_scarica_B = Transition(scarica_B, dummy_task);
- e_va_in_I = Transition(va_in_I, dummy_task);

Processi che costituiscono il sistema

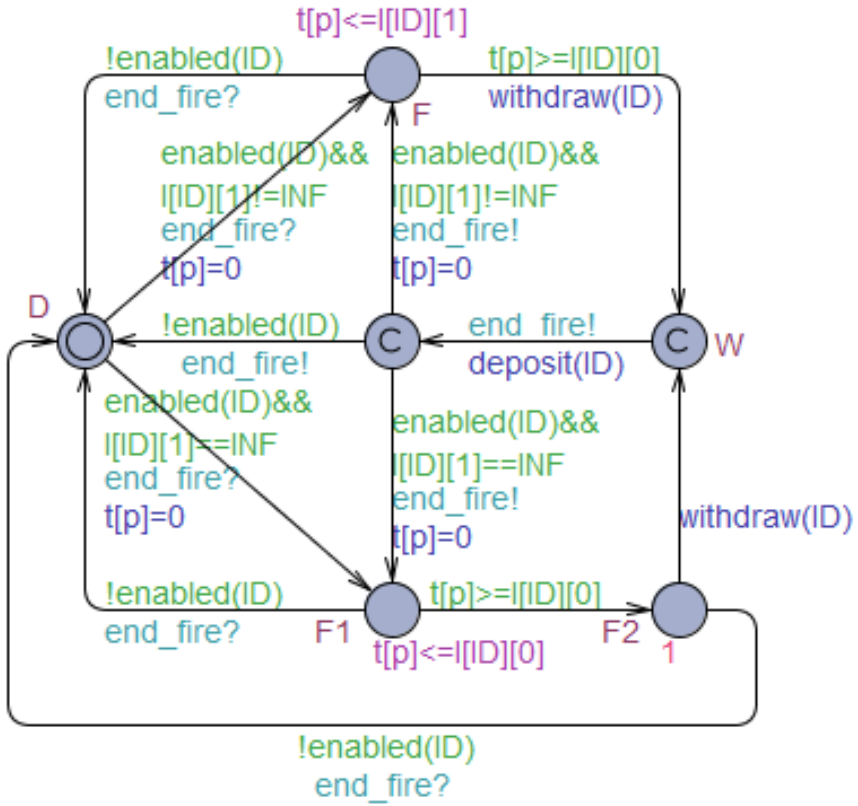
- Starter;
- e_invio_richiesta_B, e_va_in_B;
- e_agv_va_in_B, e_scarica_B;
- e_invio_richiesta_A;
- e_agv_va_in_A;
- e_scarica_A;
- e_agv_va_in_I;
- e_carica_pezzi;
- e_nessun_pezzo;
- e_rifornimento_pezzi;
- e_va_in_I;

MAGAZZINO CIRCOLARE

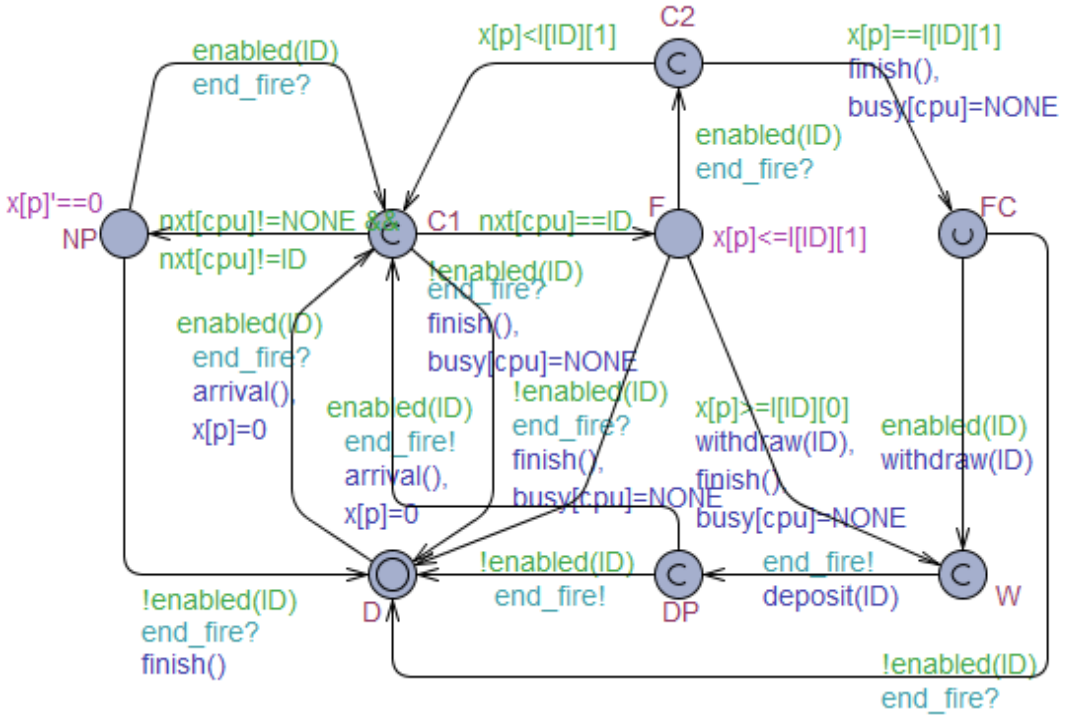
Modellato come Timed Automata con UPPAAL

Starter

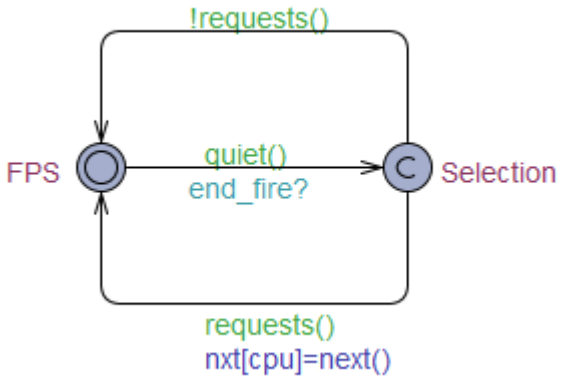
Transition



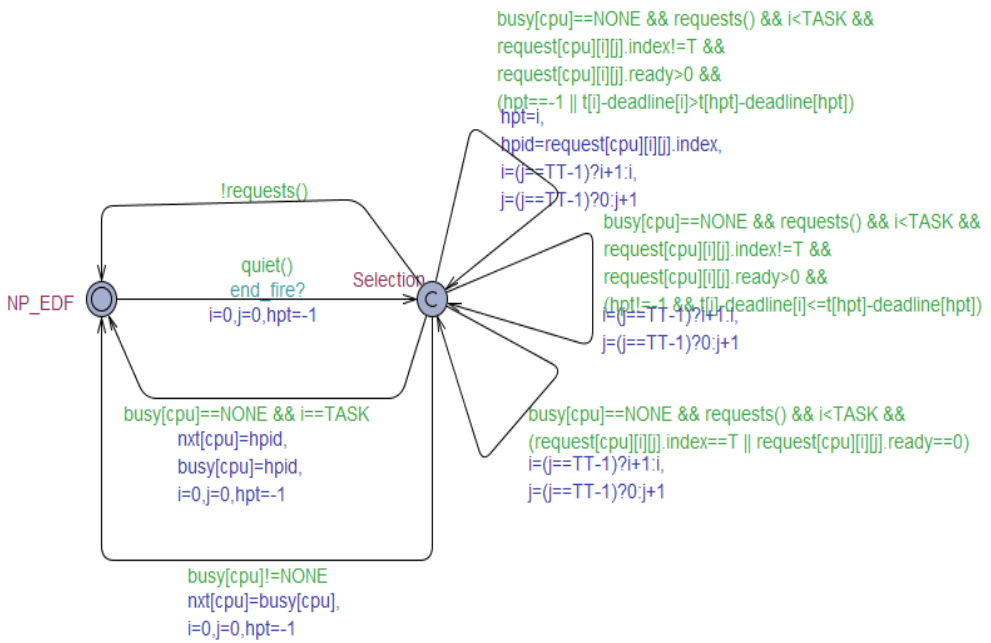
pTransition



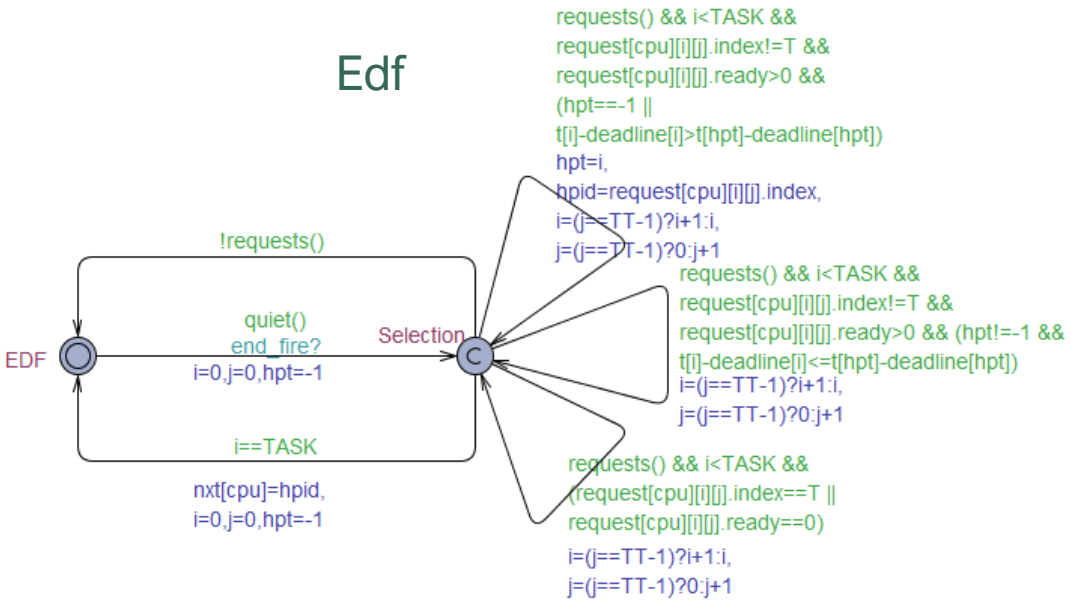
Fps



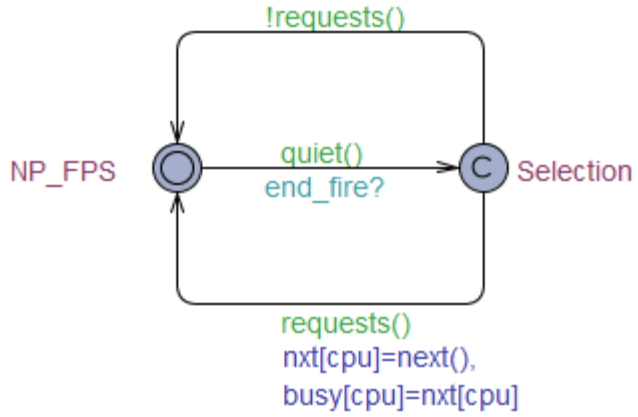
npEdf



Edf



npFps



MAGAZZINO CIRCOLARE

Modellato come Timed Automata con UPPAAL

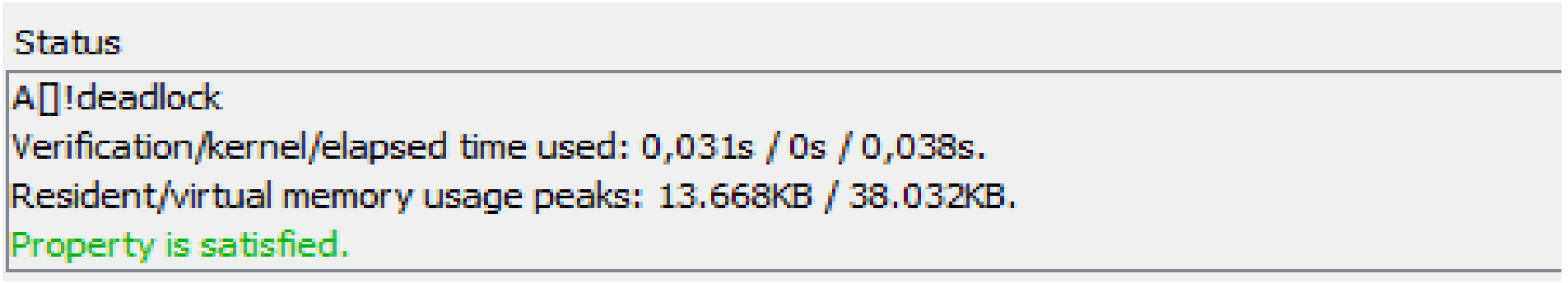
VERIFICA DEL MODELLO

L'automa temporizzato di Uppaal è stato verificato tramite il sotto-ambiente **Verifier**, inserendo delle query da processare.

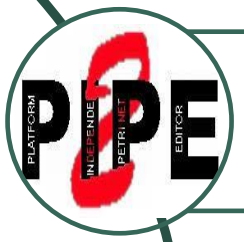
Ad esempio, è stato possibile verificare se il sistema andasse in deadlock, scrivendo $A[\ \ ]\,!\,deadlock$, dove A sta ad indicare tutti i possibili *path* nell'automa.

Il vantaggio di testare questa proprietà su Uppaal è che, a differenza di altri software, come TPN-Designer, questo strumento consente di visualizzare ugualmente i risultati ottenuti durante la simulazione, pur verificandosi un blocco nel sistema.

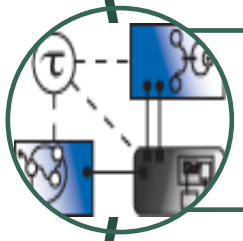
Nel caso in esame non è stato riscontrato alcun deadlock, infatti il risultato è il seguente:



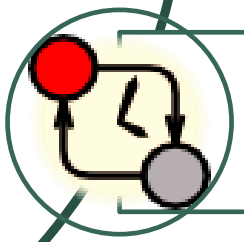
STRUMENTI UTILIZZATI



Platform Independent Petri net Editor 2, uno strumento open-source multi-piattaforma per la creazione e l'analisi di reti di Petri.



TPN Designer, un ambiente integrato di progettazione grafica e simulazione di reti TPN, scritto in Java e sviluppato nel Laboratorio di Ingegneria del Software dell'UNICAL.



Uppaal, uno strumento software per la verifica di sistemi real-time, modellati sotto forma di reti di automi a tempo.

CONTATTI

IVONNE RIZZUTO

Email: ivonne.rizzuto@gmail.com

Github: ivochan

GRAZIE PER L'ATTENZIONE!