RETI DI PETRI COLORATE : REALIZZAZIONE DI UN TRADUTTORE PROLOG/PROMELA PER L'ANALISI CON IL MODEL CHECKER SPIN

Realizzato da:
Davide Galeotti
Per il corso di Linguaggi e Modelli Computazionali LM
Università degli Studi di Bologna – Seconda Facoltà di Ingegneria
A.A. 2011-2012

CPN: Definizione formale

Una rete di Petri colorata è una tupla CPN = (∑ ,P, T, A, N, C, G, E, I) in cui:

- ∑ è un insieme finito non vuoto di tipi, anche chiamato "colour set" o insieme dei colori.
- P è un insieme finito di place.
- T è un insieme finito di transition.
- A è un insieme finito di archi tale che: P ∩ T = P ∩ A = A ∩ T = ∅
- N, C, G sono funzioni denominate rispettivamente fz nodo, fz colore, fz guardia.
- E è la funzione che rappresenta l'espressione relativa ad un arco.
- I è la funzione di inizializzazione.

CPN: Proprietà

FAIRNESS

Sia $X \subseteq BE$ un set di binding elements e sia σ una sequenza infinita di stati.

I. X è imparziale per σ iff si presenta infinite volte in σ, cioè iff: $OC_X(\sigma) = \infty$.

II. X è "fair" per σ iff una sequenza infinita di attivazioni implica una sequenza infinita di occorrenze, cioè iff: $EN_X(\sigma) = \infty \Rightarrow OC_X(\sigma) = \infty$.

III. X è "just" per σ iff un'attivazione persistente implica una occorrenza, cioè iff:
$\forall i \geq 1: [EN_{X,i}(\sigma) \neq 0 \implies \exists k \geq i: [EN_{X,k}(\sigma) = 0 \vee OC_{X,k}(\sigma) \neq 0]]$

BOUNDEDNESS

Sia p un place tale che p ϵ P, n un numero intero non negativo tale che n ϵ N ed m un multi-set tale che m ϵ $C(P)_{MS}$ allora:

I. n è un bound intero per p iff:
$\forall M \in [M_0 \rangle : |M(p)| \leq n$.

II. M è un multi-set bound per p iff:
$\forall M \in [M_0 \rangle : M(p) \leq m$.

HOME STATE

Sia M un marking tale che M ϵ M e X sia un set di marking tali che X ⊆ M allora:

I. M è un home marking iff :
$\forall M' \in [M_0 \succ : M \in [M' \succ$.

II. X è un home space iff:
$\forall M' \in [M_0 \succ X \cap [M' \succ \neq \emptyset$.

LIVENESS

Sia M un marking tale che M ϵ M e X sia un set di binding element tali che X ⊆ BE allora:

I. M è "dead" iff nessun binding element è attivo, cioè iff: $\forall x \in BE: \neg M[x \succ$.

II. X è "dead" in M iff nessun elemento di X può diventare attivo, cioè iff :
$\forall M' \in [M \succ, \forall x \in X : \neg M'[x \succ$.

III. X è "live" iff non esiste un marking raggiungibile in cui X è "dead", cioè iff:
$\forall M' \in [M_0 \succ \exists M'' \in [M' \succ \exists x \in X: M''[x \succ$.

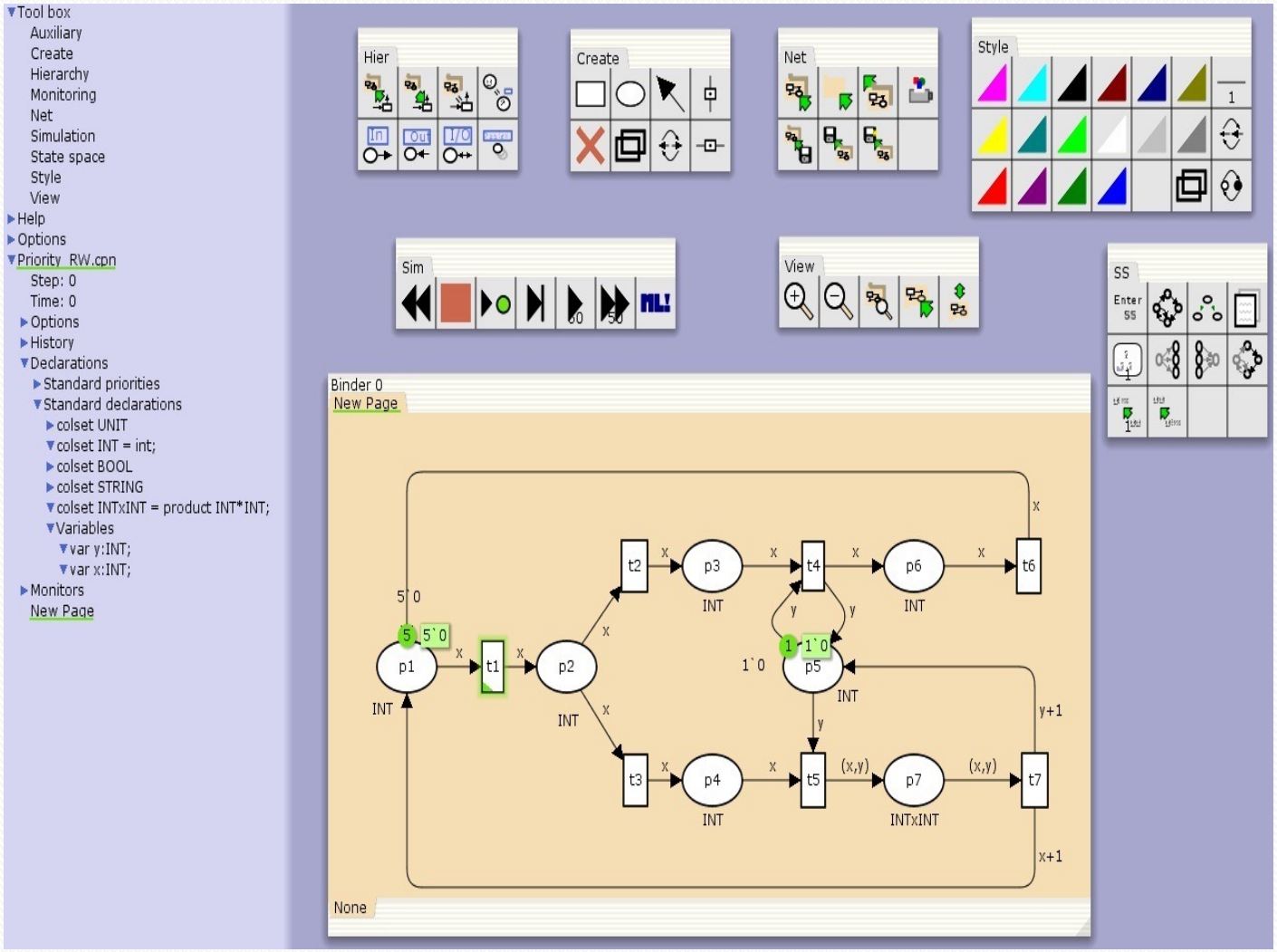
CPNTools

- Software non open source che permette di **creare**, **simulare** ed **analizzare** le reti di Petri colorate per via **grafica**, verificandone inoltre **proprietà** e **performance**.

- Disponibile per Windows e per sistemi Linux based.

- Molto utilizzato, anche perchè è l'unico tool veramente **completo**.

- Fa uso di linguaggi ad-hoc e librerie proprietarie, come ASK-CTL, per la verifica formale di proprietà.



Model checking: Spin e Promela

Spin

- Sviluppato dal 1980 ai Bell Labs
- Analisi di sistemi concorrenti e distribuiti (protocolli, OS)
- Individua deadlock, corse critiche
- Verifica rapidamente proprietà di liveness, safety, e definite dall'utente (LTL, never-claim, assert)
- Simulazione delle specifiche Promela

Promela *(PROcess MEta LAnguage)*

- Linguaggio c-like
- Esprime i concetti di: non determinismo, processo, canale di comunicazione, guardia, etc.

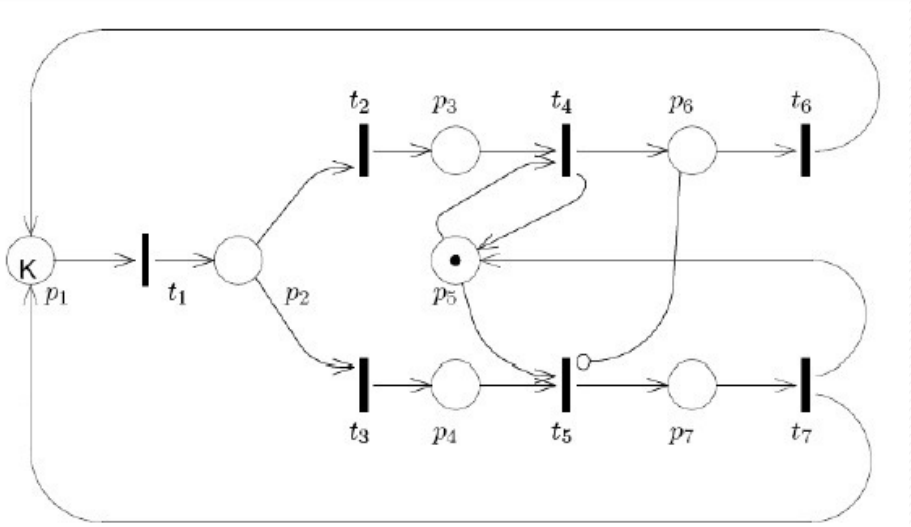
Sviluppo del progetto

- **Reti Place Transition**
  1. Modello in Promela
  2. Rappresentazione in predicati Prolog
  3. Sviluppo del traduttore

- **Reti di Petri colorate**
  4. Estensione del modello Promela
  5. Estensione rappresentazione Prolog
  6. Sviluppo del traduttore
  7. Nuove funzionalità per le CPN

Un modello in Promela per le PTN

- Problema di riferimento: Readers – Writer



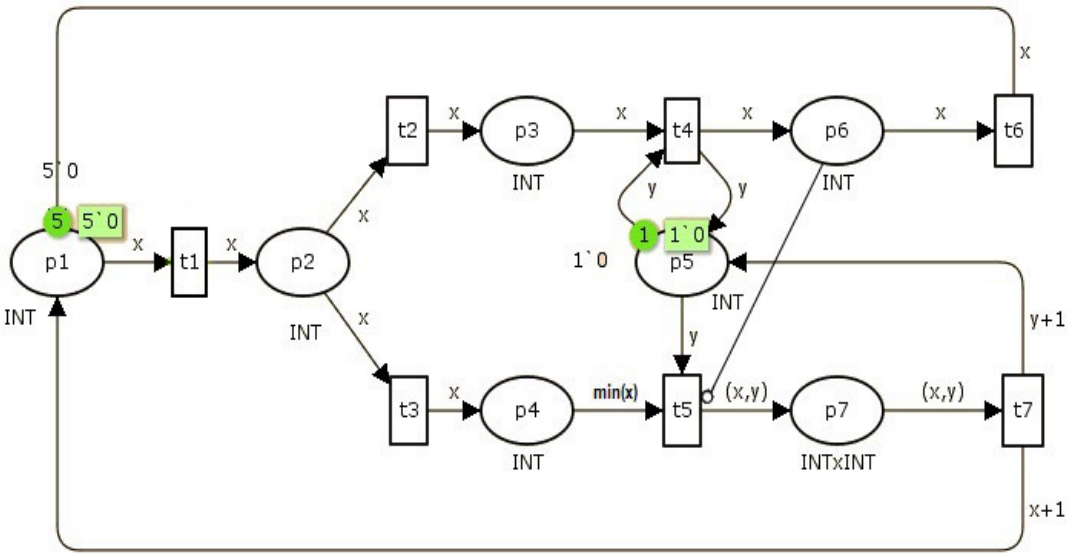
- Place: byte p1,p2...
- Main process
  - Marking iniziale: p5++;
  - Loop infinito: do{ } ,ad ogni ciclo sceglie non deterministicamente la transizione da eseguire.
- Transition:

```
:: (p6==0 && p4>0 && p5>0 && p7<Max) -> atomic{
                    p4--;
                    p5--;
                    p7++;
                }
```

Estensione del modello per le CPN

- Problema di riferimento: RW con priorità

- Funzioni:
  - setToken
  - getRandToken
  - getMinToken
  - setToken2



- Place: int p7_0[Max],p7_1[Max]; byte p7_len

- Transition:

```
:: (p6_len==0 && p4_len>0 && p5_len>0  && p7_len<Max) -> atomic{/* t5 */
                                        getMinToken(p4,p4_len,x);
                                        getRandToken(p5,p5_len,y);
                                        setToken2(p7_0,p7_1,p7_len,x,y);
                                    }
```

CPN:Definizione in Prolog

place/3

- l'identificativo del place;
- una lista di tipi che rappresentano il colore del place;
- una lista di token che rappresentano il marking iniziale;

variable/2

- Tipo della variabile;
- Nome della variabile;

transition/7

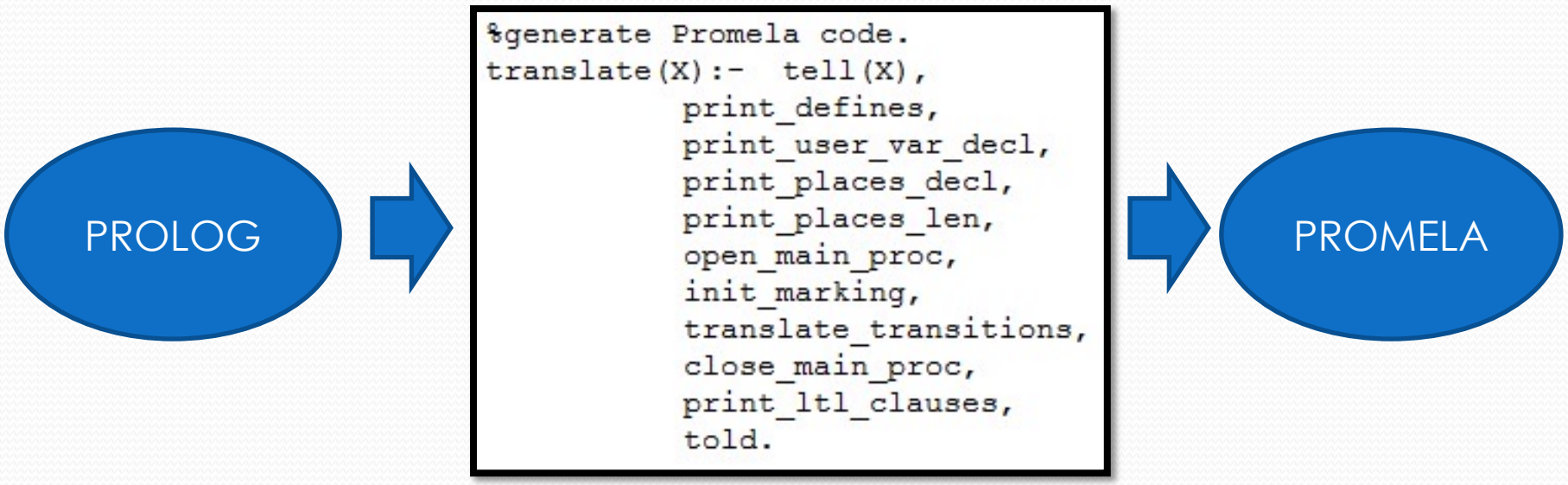
- un identificativo della transizione;
- una lista di place da cui vengono consumati i token ;
- una lista di variabili in cui vengono memorizzati temporaneamente i valori dei token;
- una lista di place in cui i token vengono inseriti;
- una lista di precondizioni che i token devono soddisfare;
- una lista di elaborazioni fatte sulle variabili;
- una lista di place che inibiscono la transizione;

Traduttore da Prolog a Promela

Lo sviluppo traduttore è stato impostato con un approccio "*Divide et impera*"

Sono state individuate due categorie:

- Codice invariante per qualsiasi rete
- Codice dipendente dalla rete specifica



Verifica formale

Proprietà formulate con la logica LTL ➡ Predicato ltl/2

Sono state verificate singolarmente per il problema Readers-Writer con priorità le seguenti:

- Safety: [] !(p6_len>0 && p7_len >0)
- Liveness: []<> (p6_len>0 || p7_len >0)
- Priority: [] true U (p4_len>0 && p7_len>0 &&
  p7_0[0]<=p4[0] && p7_0[0]<=p4[1] &&
  p7_0[0]<=p4[2] && p7_0[0]<=p4[3] &&
  p7_0[0]<=p4[4])

Funzionalità e limiti

- Controllo errori: uso di variabili riservate, match dei tipi tra variabili e token, errori sintattici.
- Precondizioni : supporto delle funzioni min, max, e di tutti gli operatori di confronto.
- Supporto ai colori multipli per i place.

- Capienza massima dei place limitata a qualche decina di token.
- Colori supportati: byte, int, bool.
- Massimo numero di colori per place limitato a due.
- Mancanza di un'interfaccia grafica.