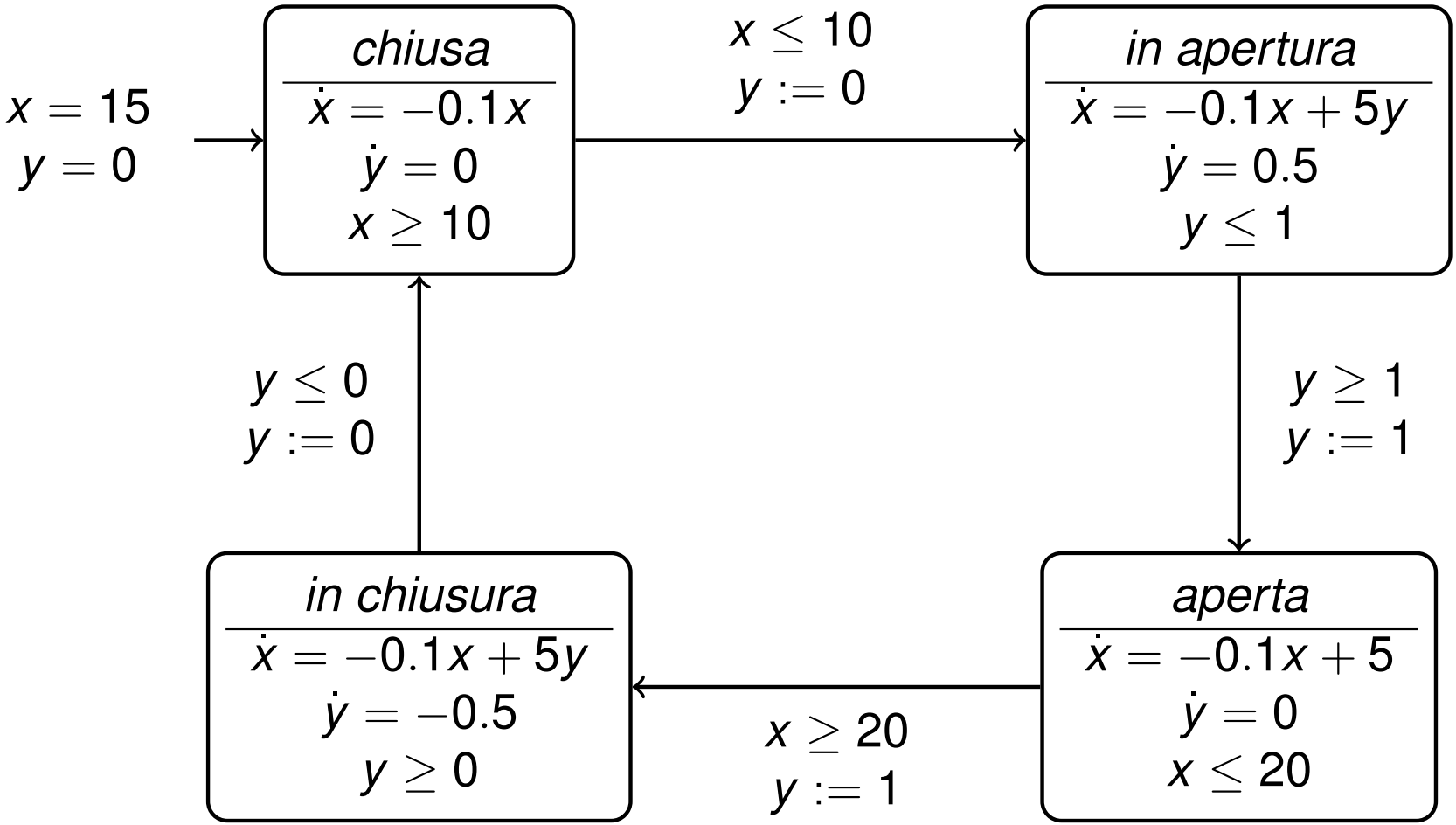
Times

Laboratorio di Sistemi in Tempo Reale

Corso di Laurea in Informatica Multimediale

29 Novembre 2007

Ariadne: un consiglio



Outline

TIMES è un pacchetto per la modellazione e l'analisi di schedulabilità di sistemi in tempo reale

- sviluppato dall'Università di Uppsala (Svezia)
- unisce due mondi: problemi di scheduling e automi temporizzati
- utilizza Uppaal come "motore"
- interfaccia grafica
- simulatore
- analisi di schedulabilità
- verifica di proprietà di sicurezza e raggiungibilità
- permette di generare codice automaticamente per BrickOS

Programma e documentazione: `www.timestool.com`

Descrivere sistemi in tempo reale in Times

Un sistema in tempo reale è descritto mediante:

- un insieme di task;
- una strategia di schedulazione (RM, DM, EDF, ...);
- un insieme di processi, definiti come automi temporizzati.

I Task in Times

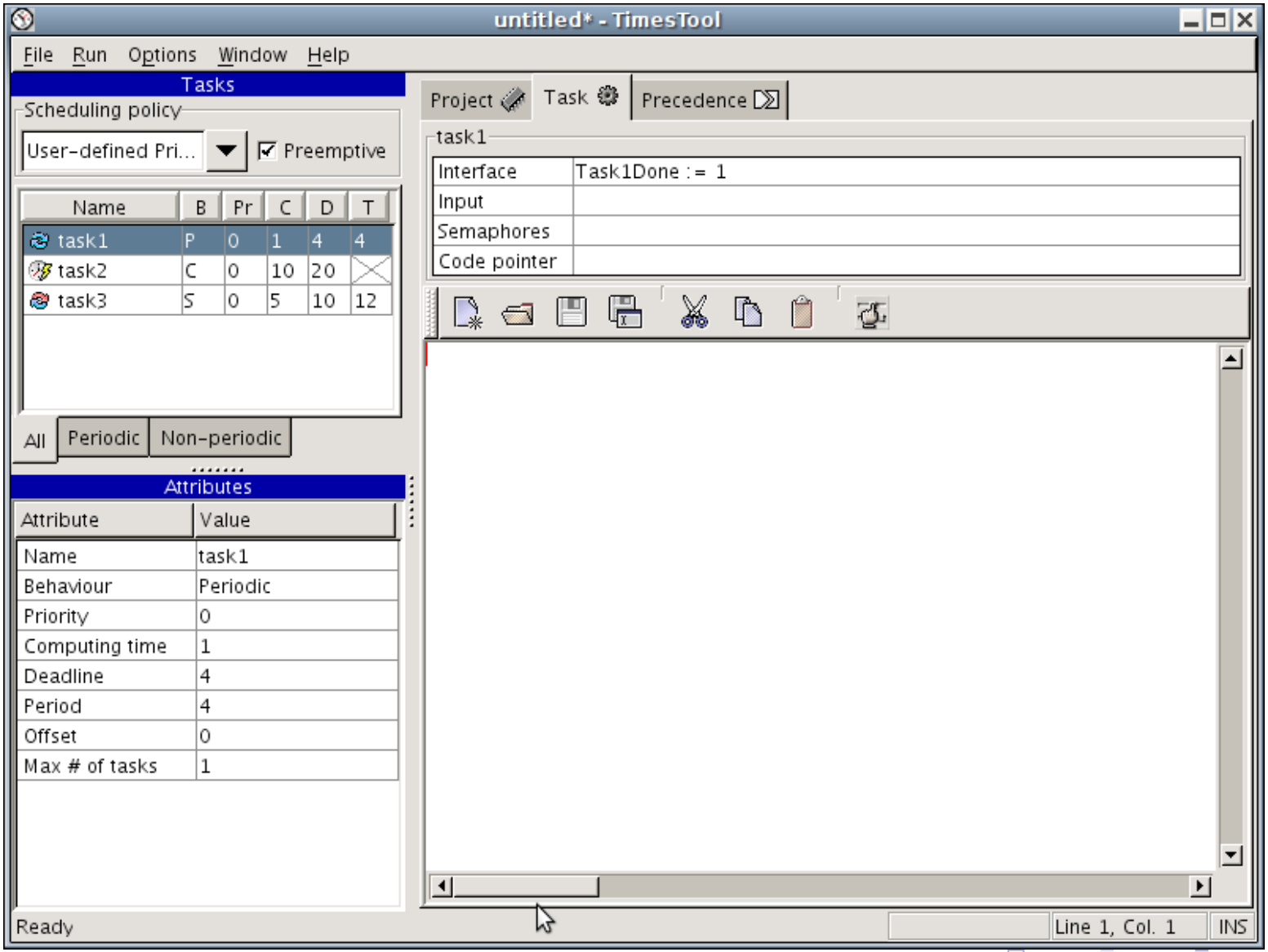
Un task in Times è caratterizzato da:

- un tempo di esecuzione C;
- una scadenza D;
- un'interfaccia costituita da una serie di assegnamenti di variabili discrete $x_1 := e_1, \ldots, x_n := e_n$.

Può essere di tre tipi:

- periodico: si attiva ad ogni un periodo P;
- sporadico: la periodicità non è nota a priori, viene trattato imponendo un periodo minimo P;
- controllato: l'attivazione del task dipende dal sistema di automi temporizzati dato dall'utente.

I Task in Times



Automi Temporizzati estesi con Task

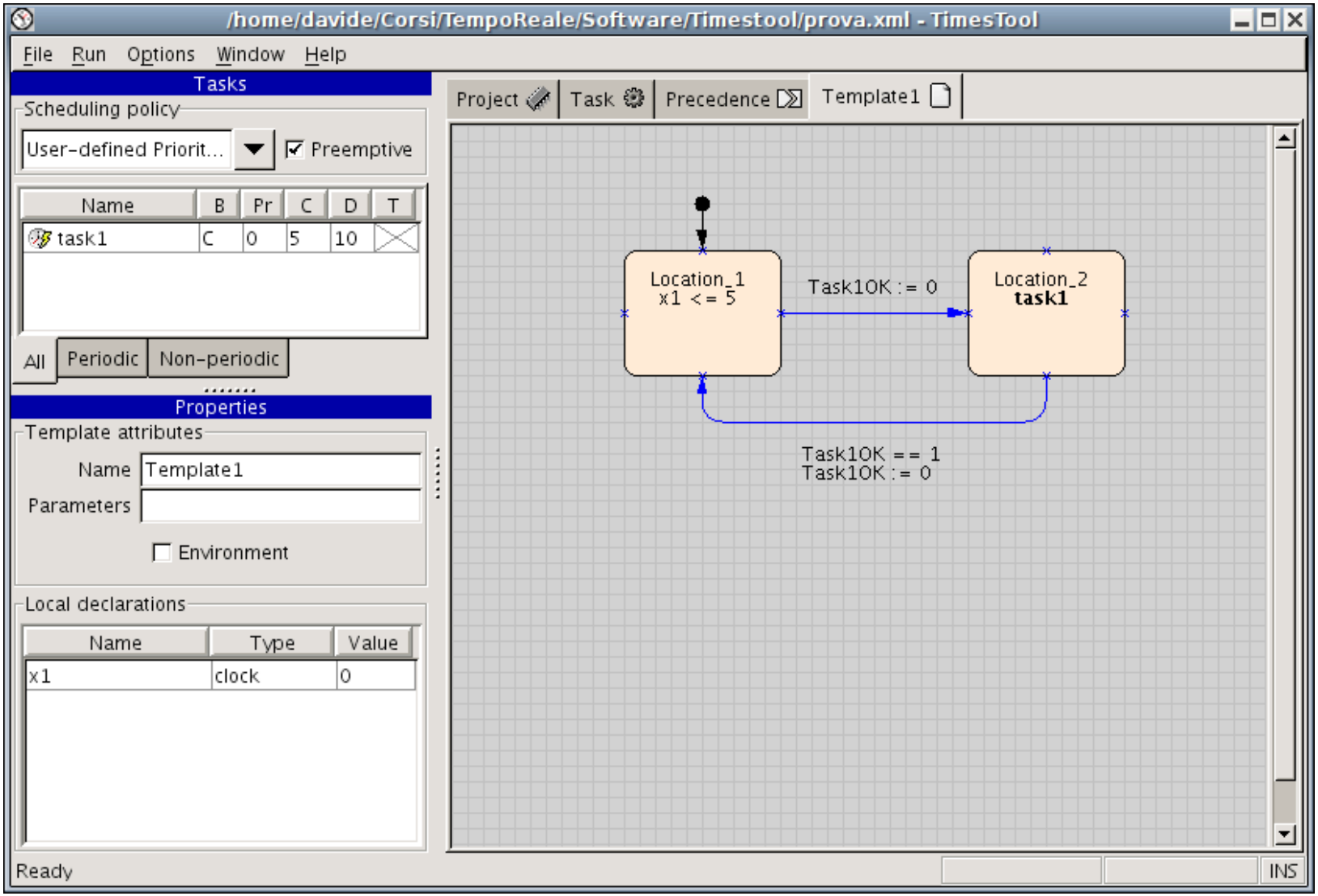
I processi sono rappresentati da Automi temporizzati estesi:

- orologi;
- variabili discrete;
- canali di comunicazione `chan!`, `chan?`;

Le locazioni possono essere etichettate con un task controllato T:

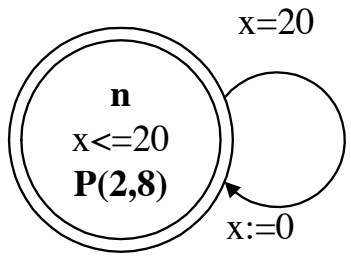
- quando si entra nella locazione, il task T viene attivato e inserito nella coda dei pronti;
- il momento di esecuzione del task dipende dalla politica di scheduling.

Automi Temporizzati estesi con Task



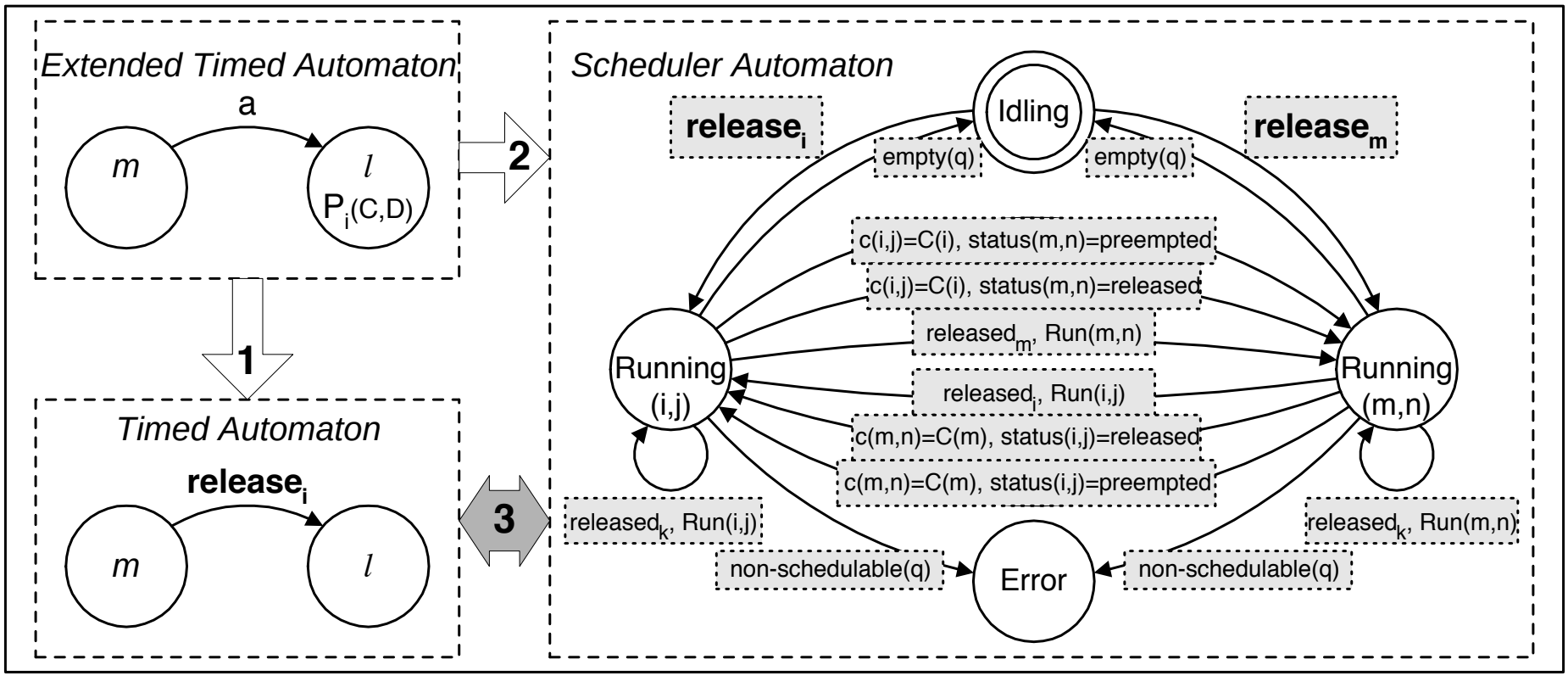
Tradurre i problemi di scheduling in automi temporizzati (1)

1. I task periodici e sporadici possono essere rappresentati da automi:



2. Dati i task e la politica di scheduling, viene generato un automa temporizzato che la implementa;
3. I task che etichettano le locazioni diventano canali di sincronizzazione con l'automa dello scheduler.

Tradurre i problemi di scheduling in automi temporizzati (2)



Tradurre i problemi di scheduling in automi temporizzati (3)

Questo approccio presenta molti vantaggi:

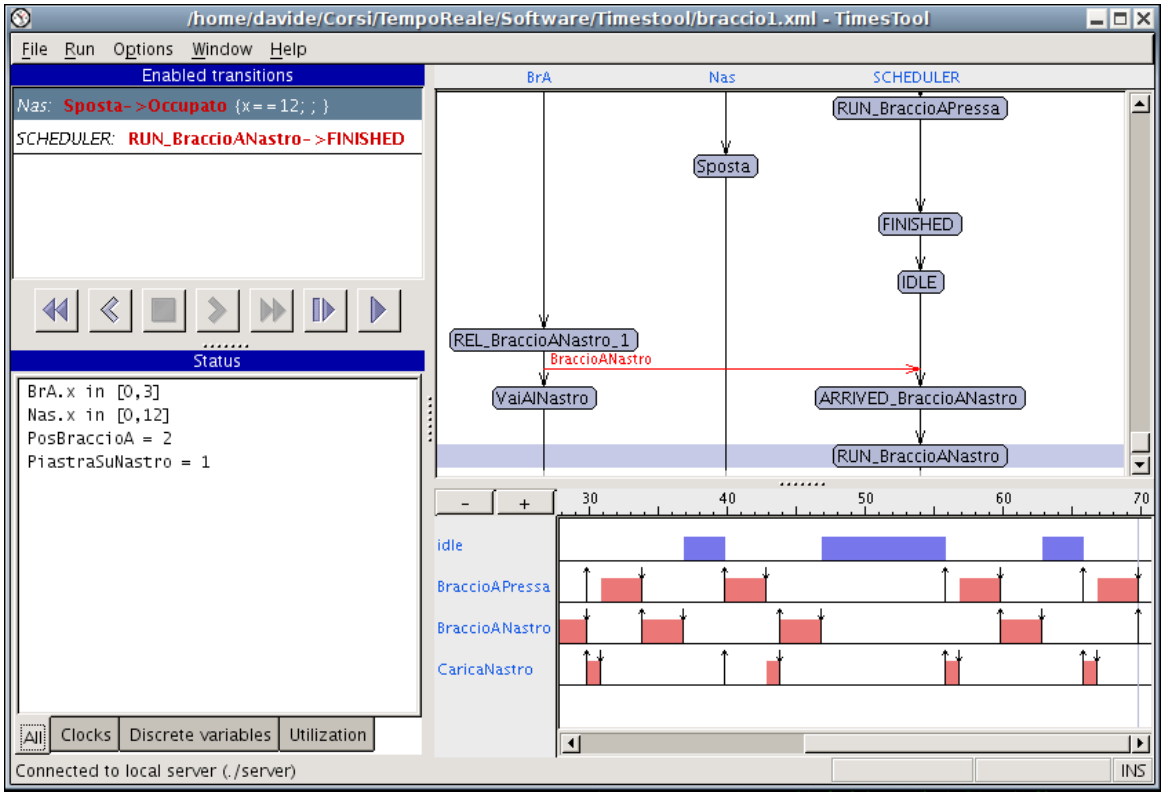
- l'analisi di schedulabilità si riduce ad un problema di raggiungiblità:
  - il sistema è schedulabile se e solo se non si raggiunge mai lo stato di errore
- task con periodicità complesse o variabili possono essere descritti ed implementati in modo semplice
- i task possono essere integrati in sistemi complessi

Funzionalità di Times (1)

- Analisi di schedulabilità:
  - stabilisce se la politica di schedulazione scelta soddisfa sempre le scadenze;
  - in caso positivo, visualizza i tempi di risposta massimi (Worst Case Response Time, WCRT) per ogni task;
  - in caso negativo, ritorna un controesempio non schedulabile.
- Verifica di proprietà formali di sicurezza e raggiungibiità:
  - usa lo stesso linguaggio di query di Uppaal;
  - `A[] not deadlock;`
  - `E<> x > 10;`
  - ...

Funzionalità di Times (2)

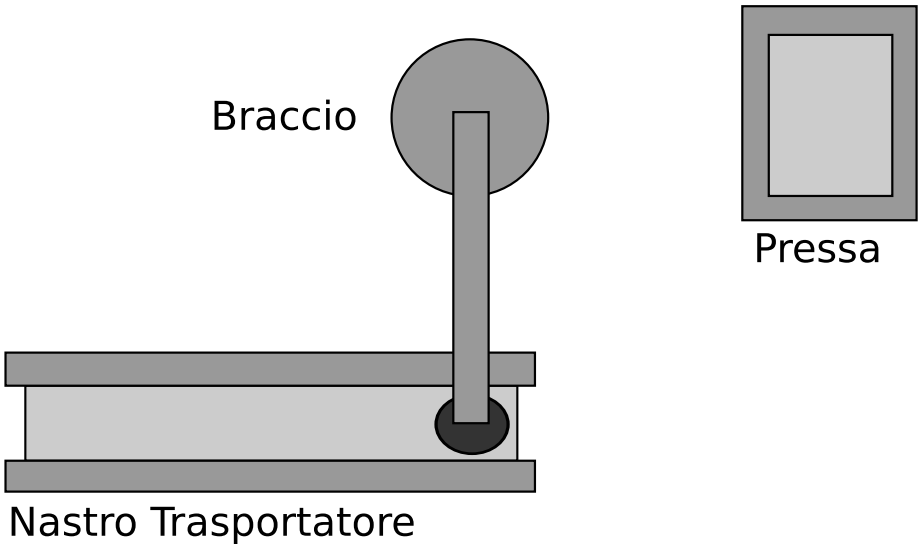
- Simulazione del sistema:



- Generazione di codice per BrickOS
  - interfaccia con i robot della Lego.

Outline

Esempio: un impianto industriale

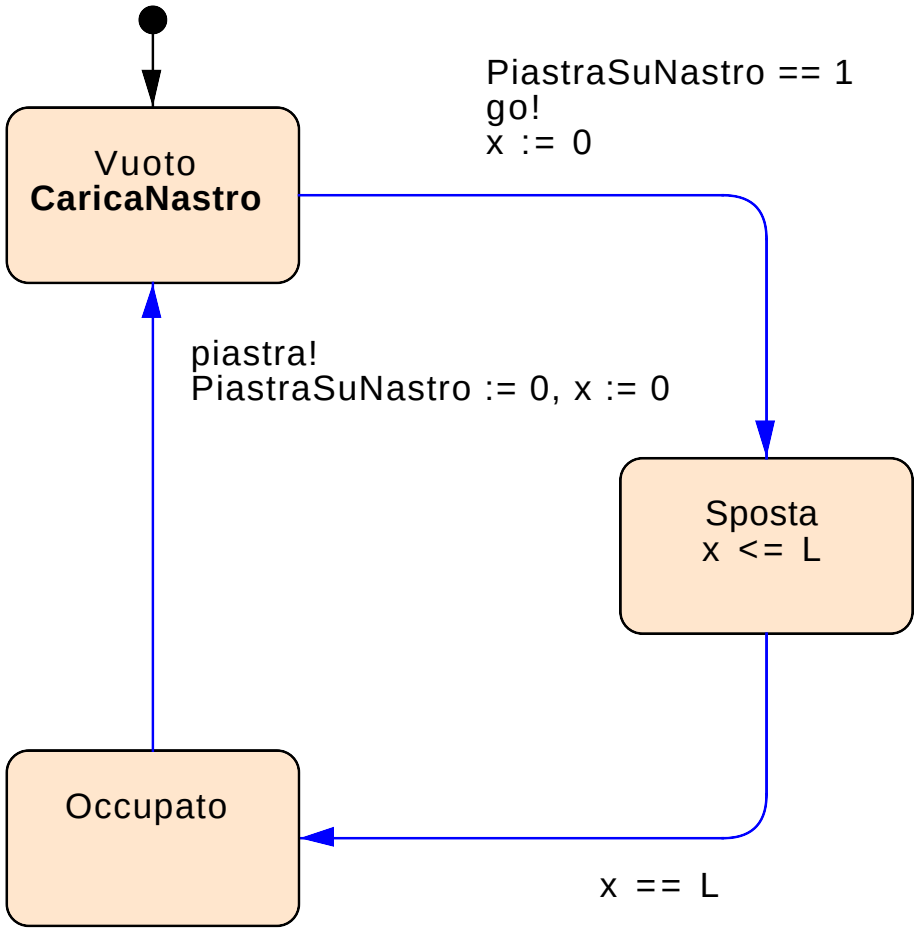


- il pezzo viene caricato sul nastro e spostato verso il braccio
- il braccio preleva il pezzo e lo sposta nella pressa
- la pressa effettua la lavorazione
- il controllore è unico e con un solo processore

I Task dell'esempio

- `CaricaNastro`:
  - Tempo di esecuzione: 1;
  - Deadline: 5;
  - Interfaccia: `PiastraSuNastro := 1`
- `BraccioANastro`:
  - Tempo di esecuzione: 3;
  - Deadline: 10;
  - Interfaccia: `PosBraccio := 1`
- `BraccioAPressa`:
  - Tempo di esecuzione: 3;
  - Deadline: 10;
  - Interfaccia: `PosBraccio := 2`
- `CaricaPressa`:
  - Tempo di esecuzione: 1;
  - Deadline: 5;
  - Interfaccia: `PressaCarica := 1`

L'automa del nastro



Nome del Template: `Nastro`

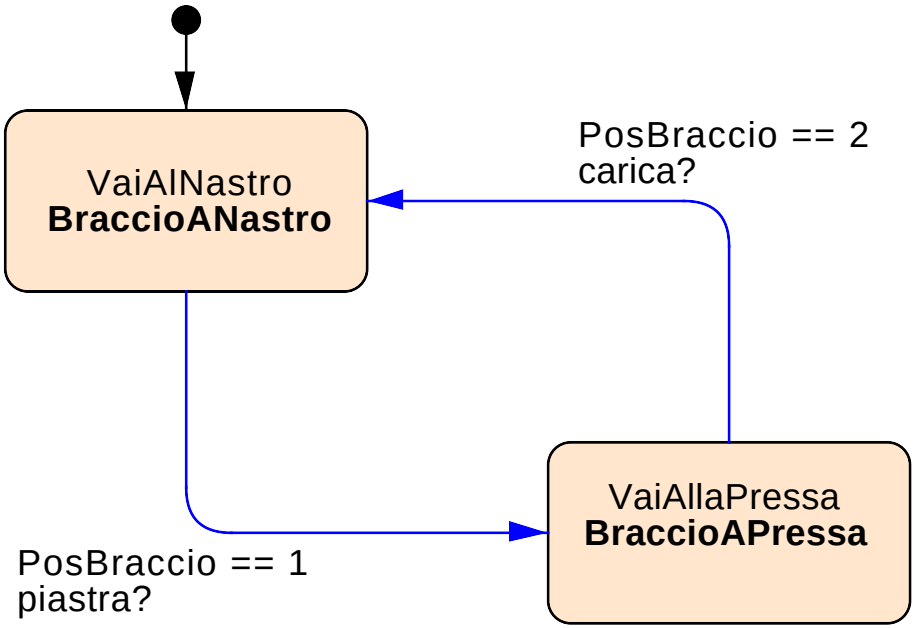
Parametri: `const L`

Variabili locali: `clock x`

Variabili globali:

```
int PiastraSuNastro
urgent chan piastra
urgent chan go
```

L'automa del braccio



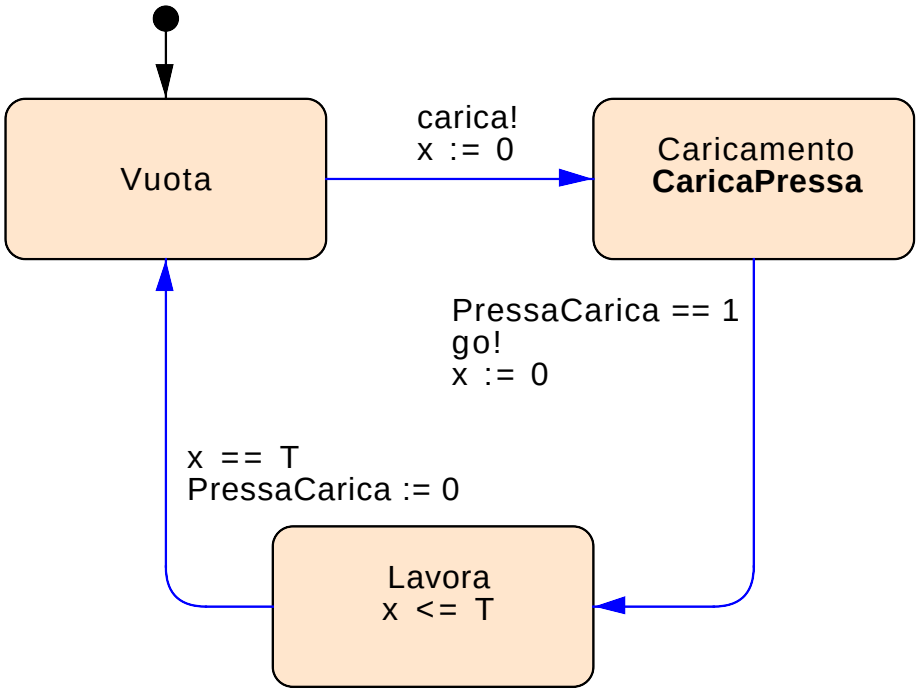
Nome del Template:

`Braccio`

Variabili globali:

```
int PosBraccio
urgent chan piastra
urgent chan carica
```

L'automa della pressa



Nome del Template:

```
Pressa
```

Parametri: `const T`

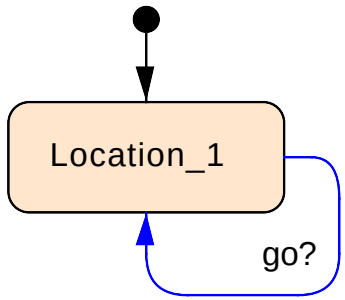
Variabili locali: `clock x`

Variabili globali:

```
int PressaCarica
urgent chan carica
urgent chan go
```

Un trucco per avere transizioni urgenti

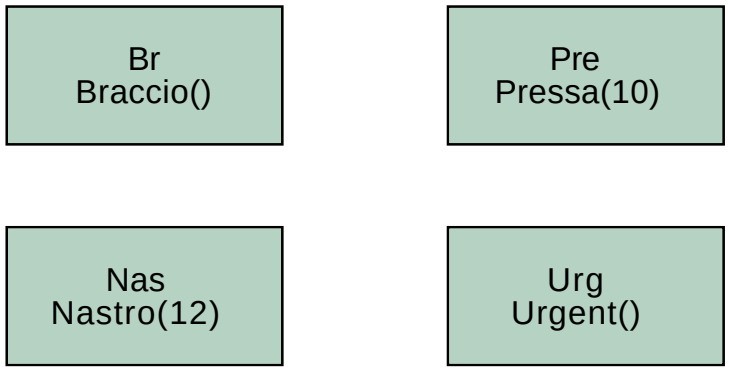
Questo automa permette di avere transizioni urgenti:



- `go` è un `urgent chan` globale;
- ogni transizione che deve essere urgente va etichettata con `go!`;
- in questo modo la transizione viene eseguita appena possibile.

Il sistema completo

Nella scheda `Project` vanno inseriti i seguenti processi:



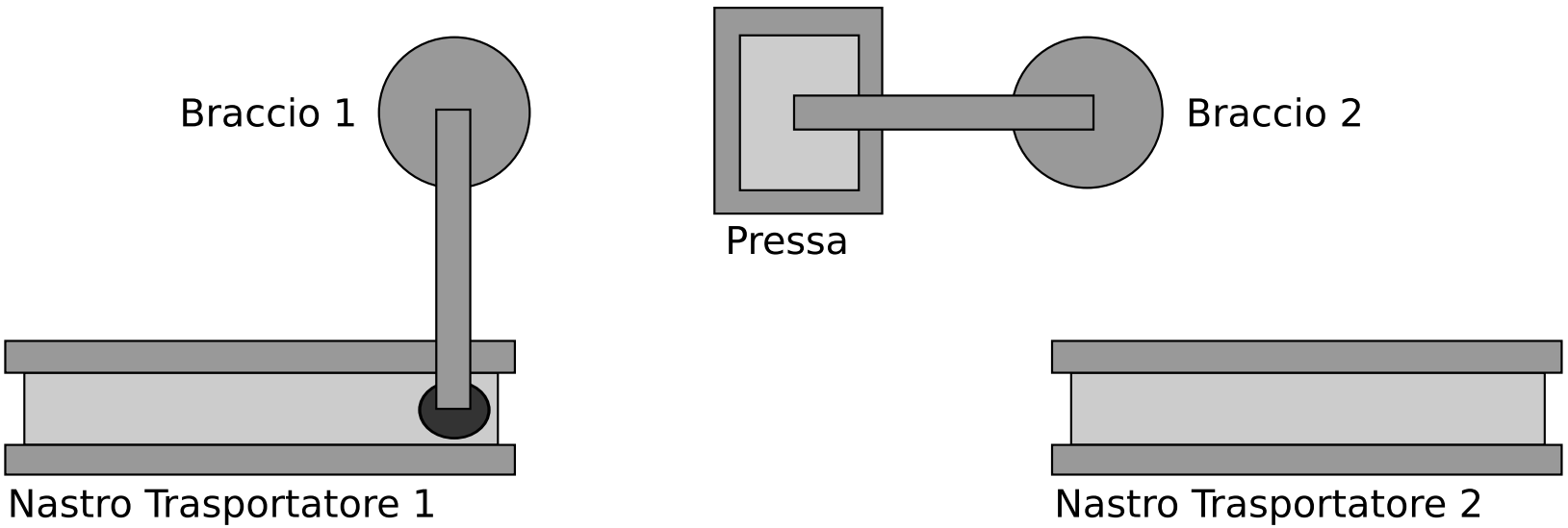
- I processi sono istanze dei template definiti in precedenza
- Il nastro impiega 12 unità di tempo per trasportare i pezzi
- La pressa impiega 10 unità di tempo per la lavorazione

Primo esercizio

- Implementare il sistema nastro/braccio/pressa in Times
- Effettuare l'analisi di schedulabilità con le varie politiche disponibili:
  - per quali politiche il sistema è schedulabile?
- Verificare se il sistema rispetta le seguenti proprietà:
  - Non va mai in deadlock:
    `A[] not deadlock`
  - Ogni pezzo rimane sul nastro al massimo 15 unità di tempo:
    `A[] not (Nas.x > 15)`

Outline

Estensione dell'impianto industriale (1)



- Aggiungere un secondo braccio e un secondo nastro all'impianto
- Braccio 2 preleva il pezzo dalla pressa e lo sposta sul Nastro 2
- Nastro 2 trasporta il pezzo lavorato

Estensione dell'impianto industriale (2)

- Implementare il sistema in Times
- Effettuare l'analisi di schedulabilità con le varie politiche disponibili:
  - per quali politiche il sistema è schedulabile?
- Verificare se il sistema rispetta le seguenti proprietà:
  - Non va mai in deadlock:
    `A[] not deadlock`
  - Ogni pezzo rimane sul nastro al massimo 15 unità di tempo:
    `A[] not (Nas.x > 15)`

http://profs.sci.univr.it/~bresolin/lab05.pdf