TILCO

Sviluppo di sistemi in tempo-reale

P. Nesi, P. Bellini, A. Giotti, D. Rogai

Sommario

- Logica Temporale TILCO
- TILCO-X e C-TILCO
- Eseguibilità di TILCO
- Controllo di sistemi in tempo-reale
- Caso di studio (incrocio semaforico)

Logica temporale TILCO

- TILCO è un linguaggio logico che estende la logica del primo ordine per esprimere le relazioni tra eventi nel dominio del tempo
- I vincoli temporali possono essere espressi sia in modo quantitativo che qualitativo; è semplice definire relazioni d'ordine tra eventi, ritardi e "time-out"
- Per gli operatori il tempo è *implicito* cioè coincide con l'istante di valutazione attuale

@	Quantificatore temporale universale $\forall$	$\left(A@[t_1,t_2)\right)^{(t)} \equiv \forall x \in [t_1,t_2).A^{(x+t)}$
?	Quantificatore temporale esistenziale $\exists$	$\left(A?[t_1,t_2)\right)^{(t)} \equiv \exists x \in [t_1,t_2).A^{(x+t)}$
until	Esprime che un predicato è sempre vero in futuro o che sarà vero finché un altro predicato non diventa vero	
since	Esprime che un predicato è sempre stato vero in passato o che è stato vero da quando un altro predicato è divenuto vero	

Esempio di specifica TILCO

Sistema di allarme con auto-disattivazione

$$danger \Leftrightarrow (critical\,@[-2,0])?[-5,0]$$

$$alert \Leftrightarrow \text{since}(\partial(\neg danger) \wedge danger, \neg ack \wedge \neg shutdown)$$

$$shutdown \Leftrightarrow \text{since}((danger \wedge alert)@[-60,-1], \neg reset)$$

Individuazione di una situazione di pericolo determinata in base alla storia della variabile booleana *critical*

Attivazione di un allarme al divenire della situazione di pericolo con interruzione manuale

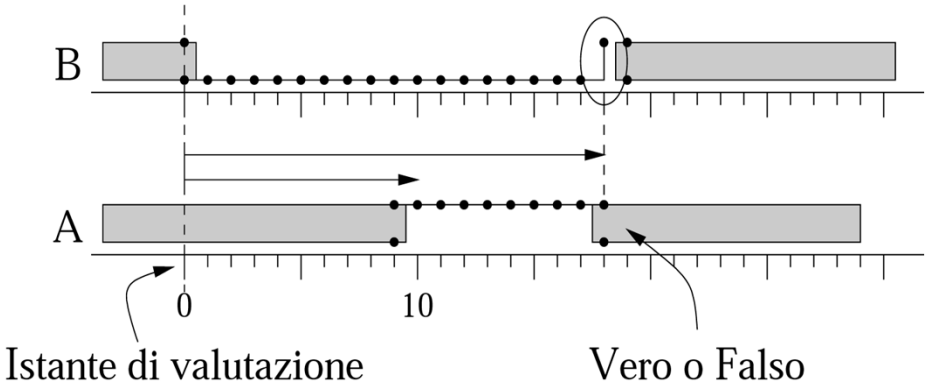
In caso di perdurante situazione di pericolo e mancata interruzione dell'allarme, auto-disattivazione del dispositivo

TILCO-X

- Per la specifica dei sistemi in tempo-reale sono molto importanti la concisione, la leggibilità e la comprensibilità della logica temporale usata
- Tali caratteristiche dipendono dall'espressività della logica dal numero degli operatori, dalla struttura delle formule…

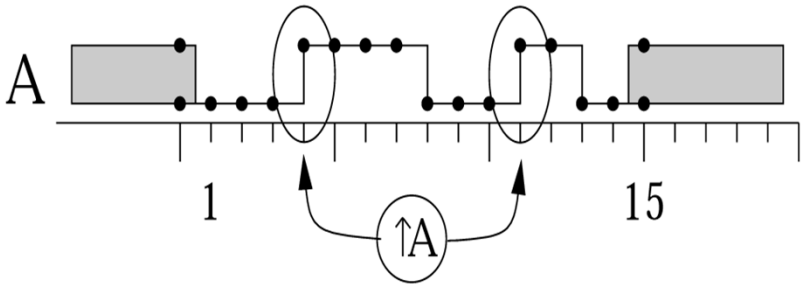
Dynamic Interval

$$A@[10,+B)$$



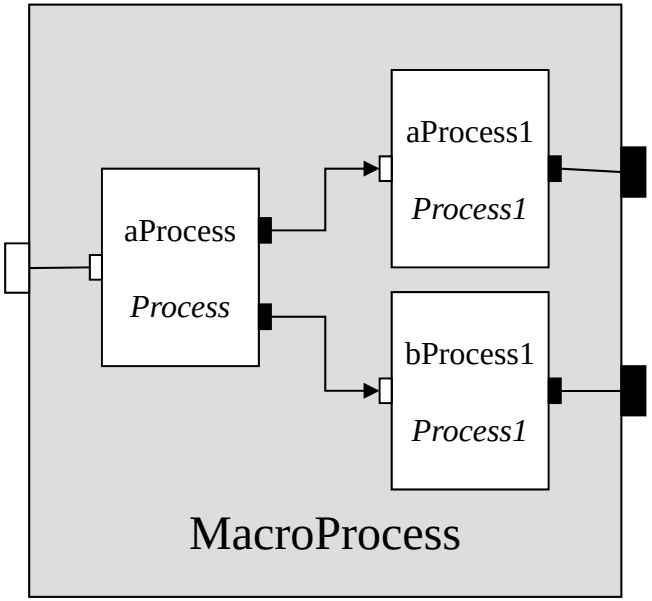
Bounded Happen

$$\left(\uparrow A\right)?_{2}^{3}[1,15)$$



C-TILCO

- Decomposizione del sistema in un insieme di processi comunicanti.
- Porte di comunicazione sincrone.
- Gerarchia di processi.
- Specifica dei processi in TILCO usando operatori specifici per la comunicazione.



TILCO eseguibile

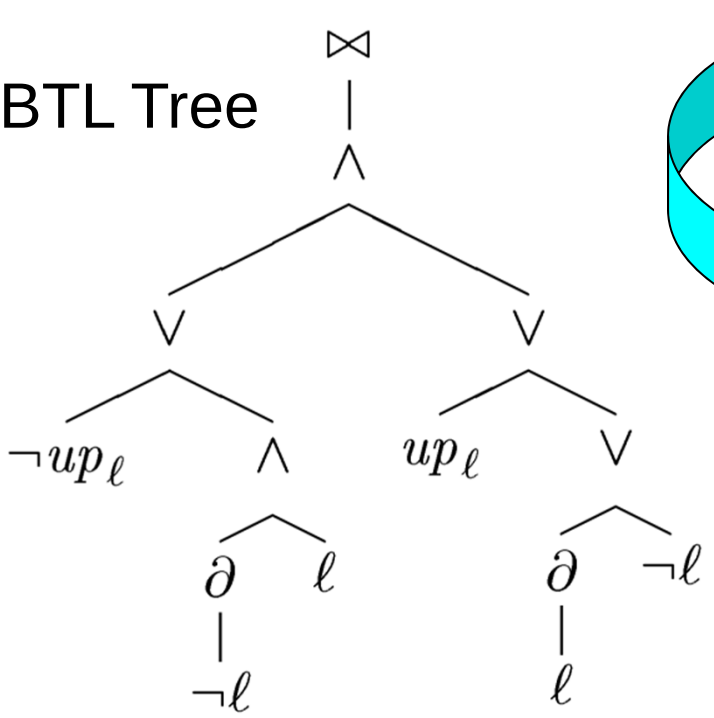
Le specifiche scritte in TILCO-X non risultano sempre eseguibili.

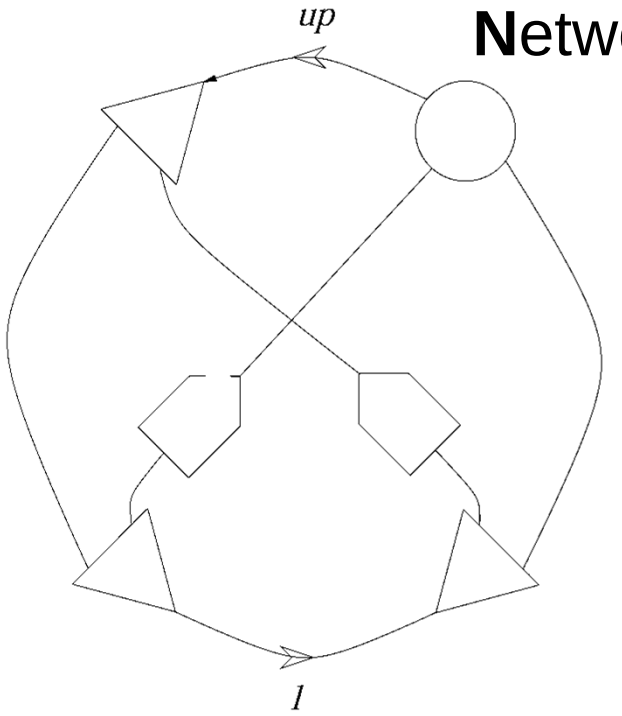
È necessario trasformare la specifica in una rappresentazione più conveniente per la sua esecuzione in tempo-reale.

$$up_\ell \Leftrightarrow \partial(\neg\ell) \wedge \ell$$

Temporal
Inference
Network

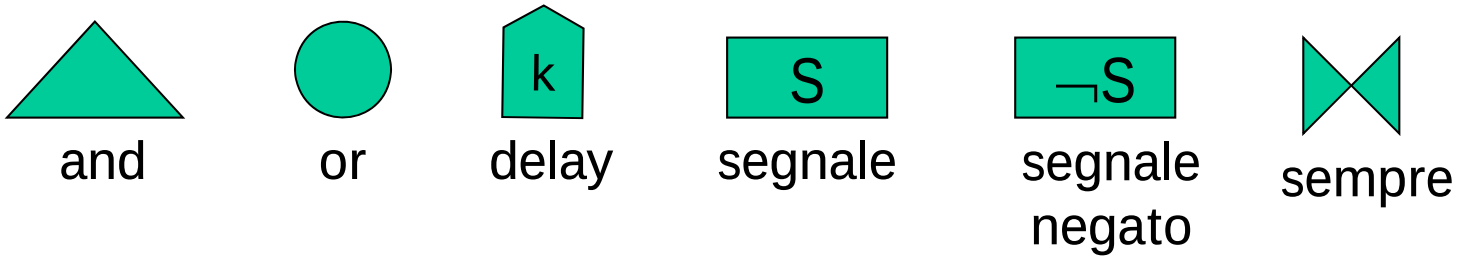
BTL Tree





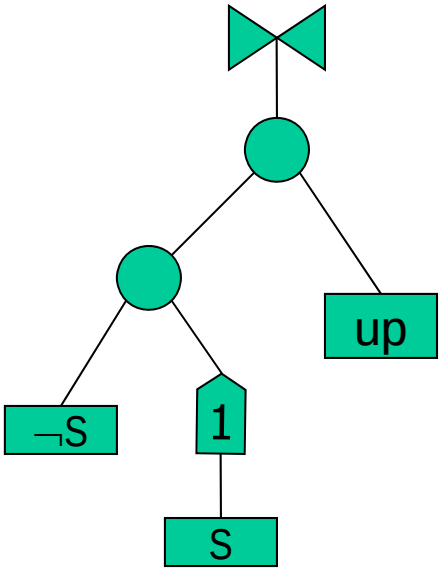
Elementi della rete TIN

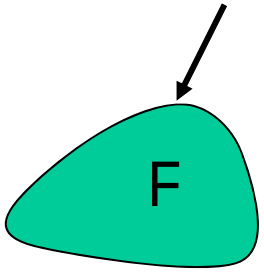
- Elementi di base:



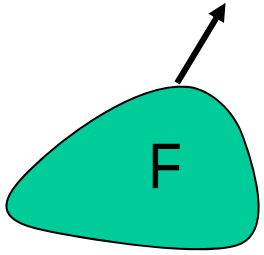
Albero sintattico

$S \wedge \partial\neg S \rightarrow up$





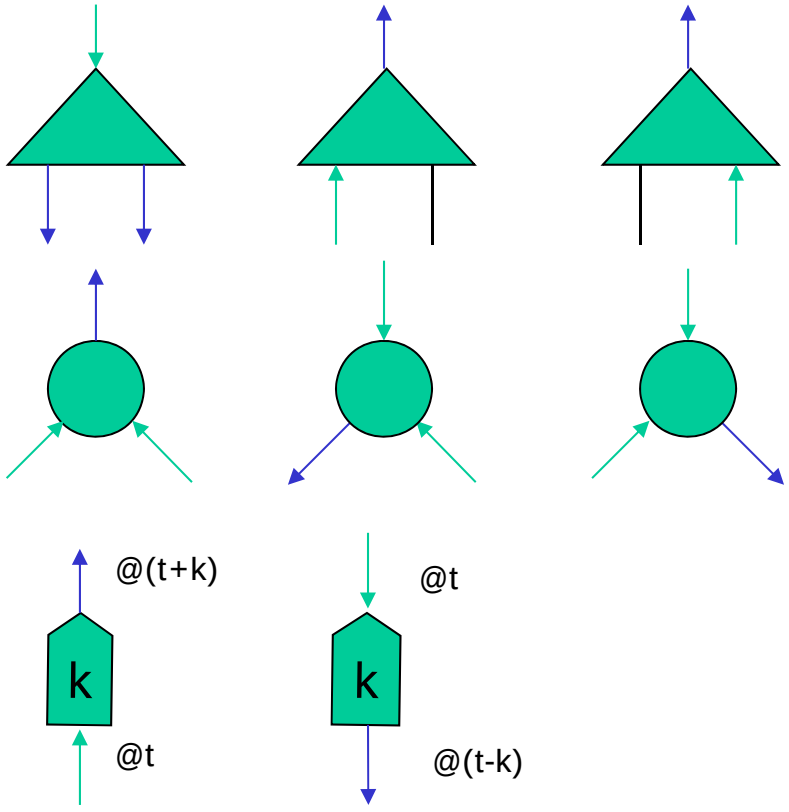
Formula F vera

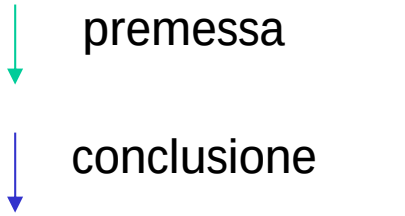


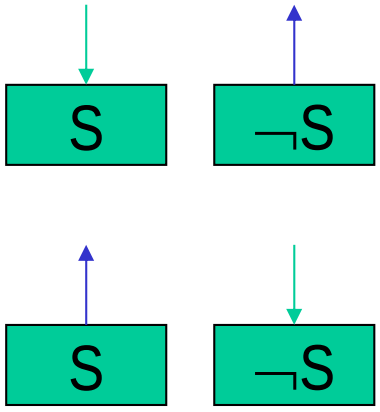
Formula F falsa

Reti di inferenza temporale

- Regole di inferenza

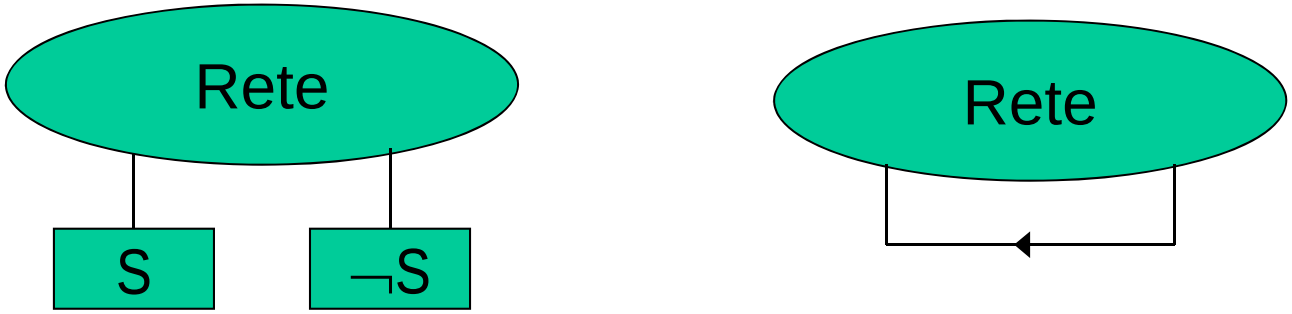




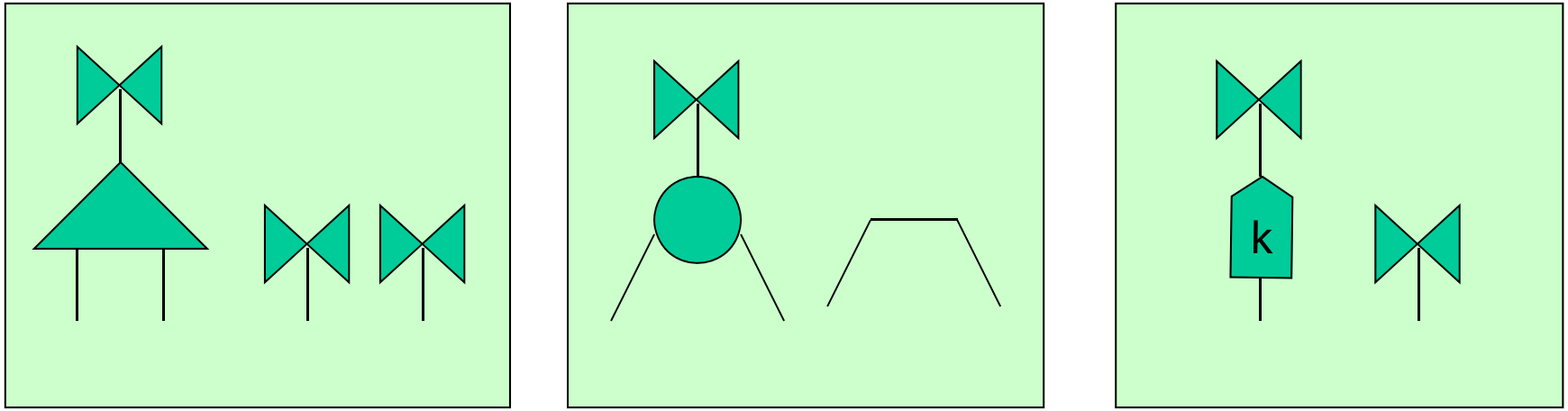


Semplificazioni della rete

- Eliminazione dei letterali di I/O



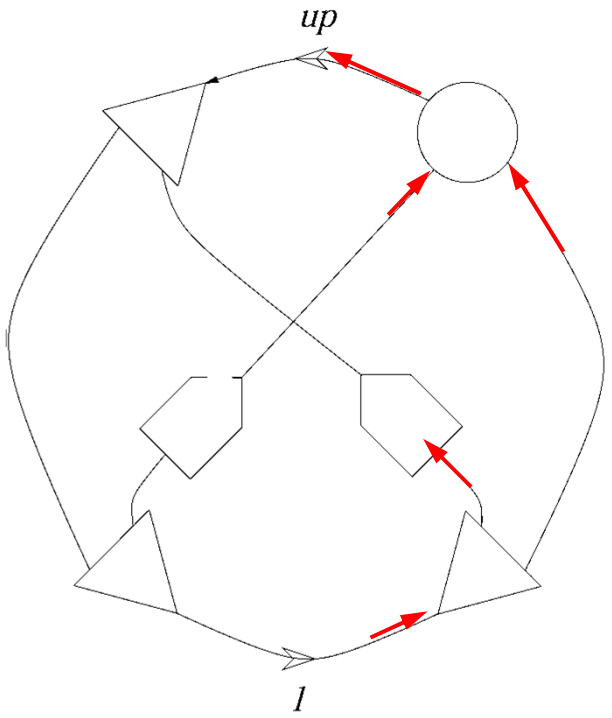
- Eliminazione degli elementi “sempre vero”



Esempio di Inferenza

Evoluzione della conoscenza all'interno della rete che a partire dalla storia del segnale *l* produce il segnale *up* che è vero in corrispondenza di ogni transizione falso-vero di *l*

$$up \Leftrightarrow \partial(\neg \ell) \wedge \ell$$



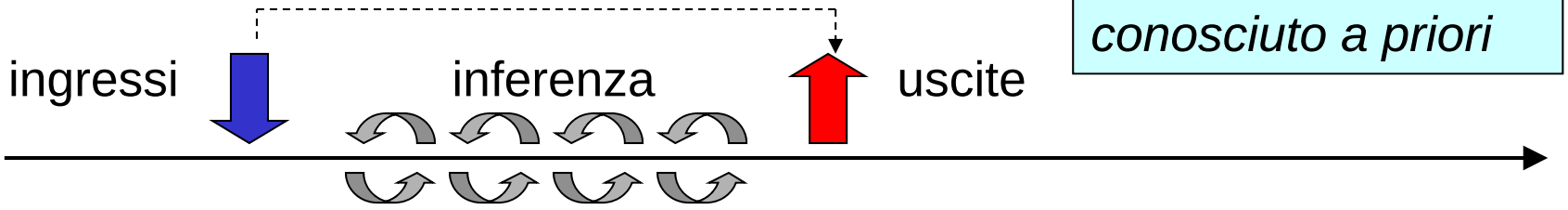
@t+1

l	vero
up	vero

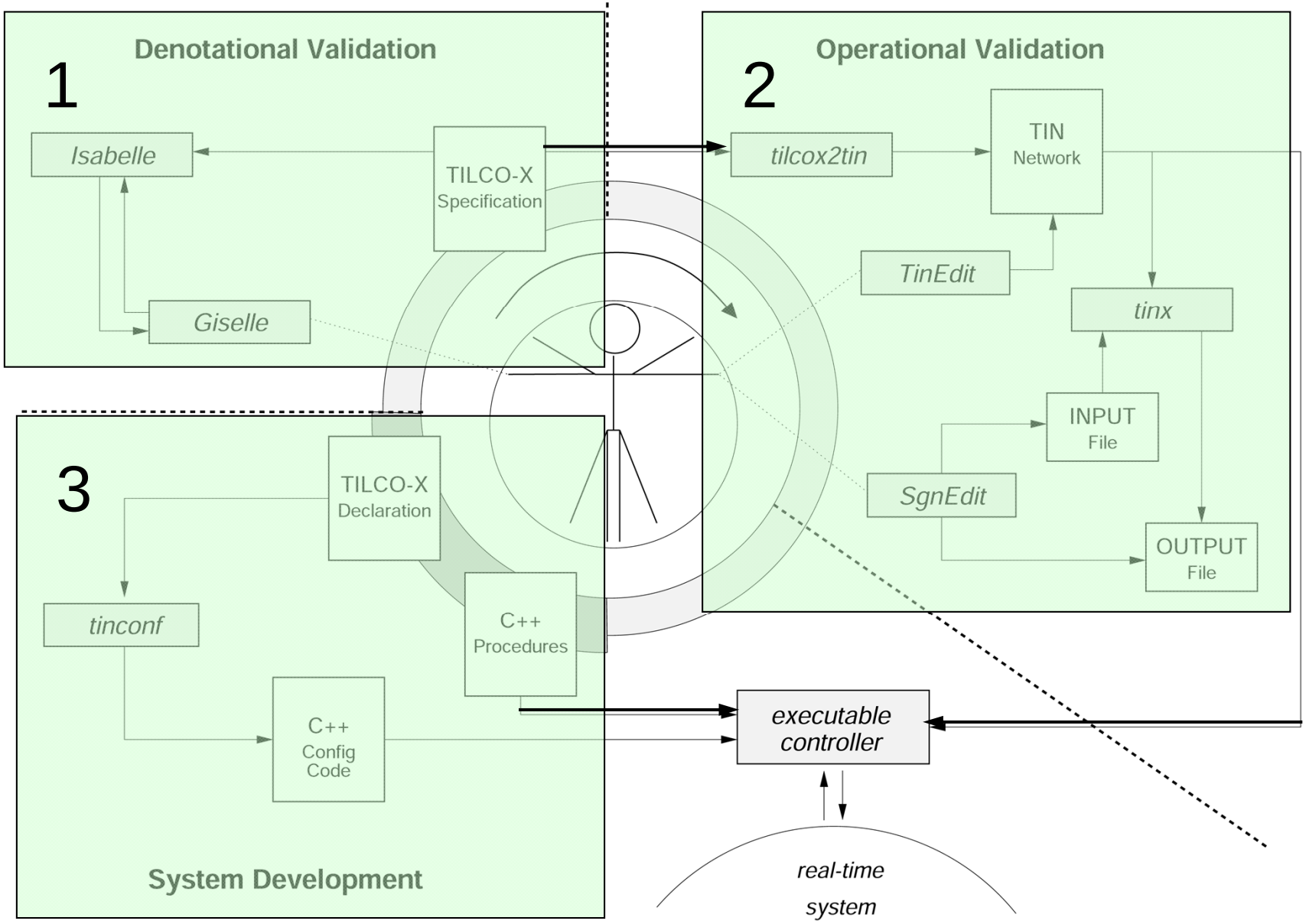
Algoritmo di inferenza

- **Procedura di chiusura progressiva**:
  - Gli ingressi sono trattati come se fossero prodotti dal processo di inferenza
  - Le regole di inferenza sono applicate solamente ai nuovi eventi (ciò che è stato processato interessa i passi successivi solo se interagente con nuovi eventi)
  - I nuovi eventi vengono smaltiti a partire da quello che si riferisce all'istante di tempo più antico
- Complessità del processo di inferenza proporzionale a parametri statici della rete
  - dimensione della rete (numero dei nodi o degli archi)
  - massimo valore all'interno dei blocchi "ritardo"

Worst-case
conosciuto a priori



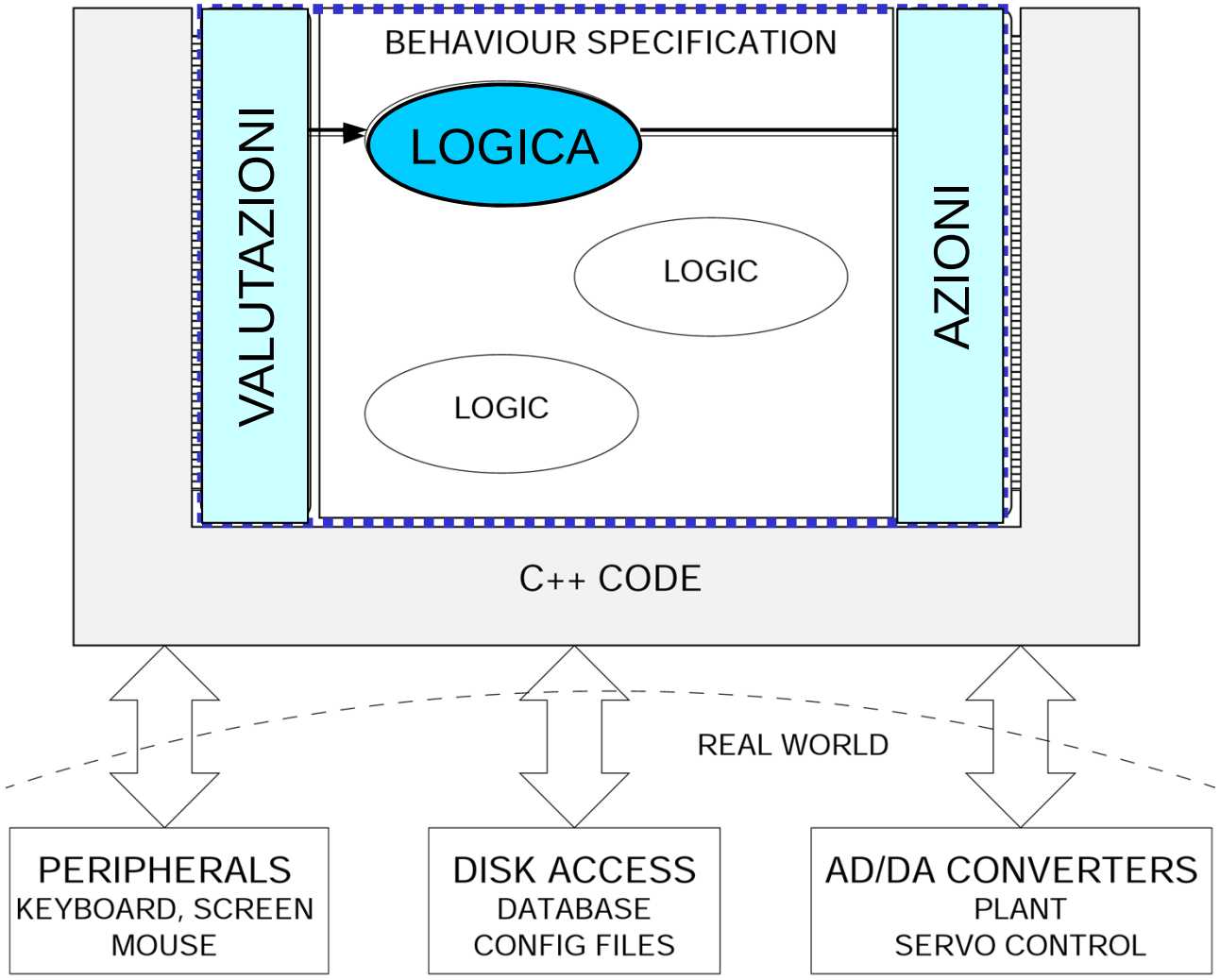
TOTS – strumenti di sviluppo



Modello di controllo 1

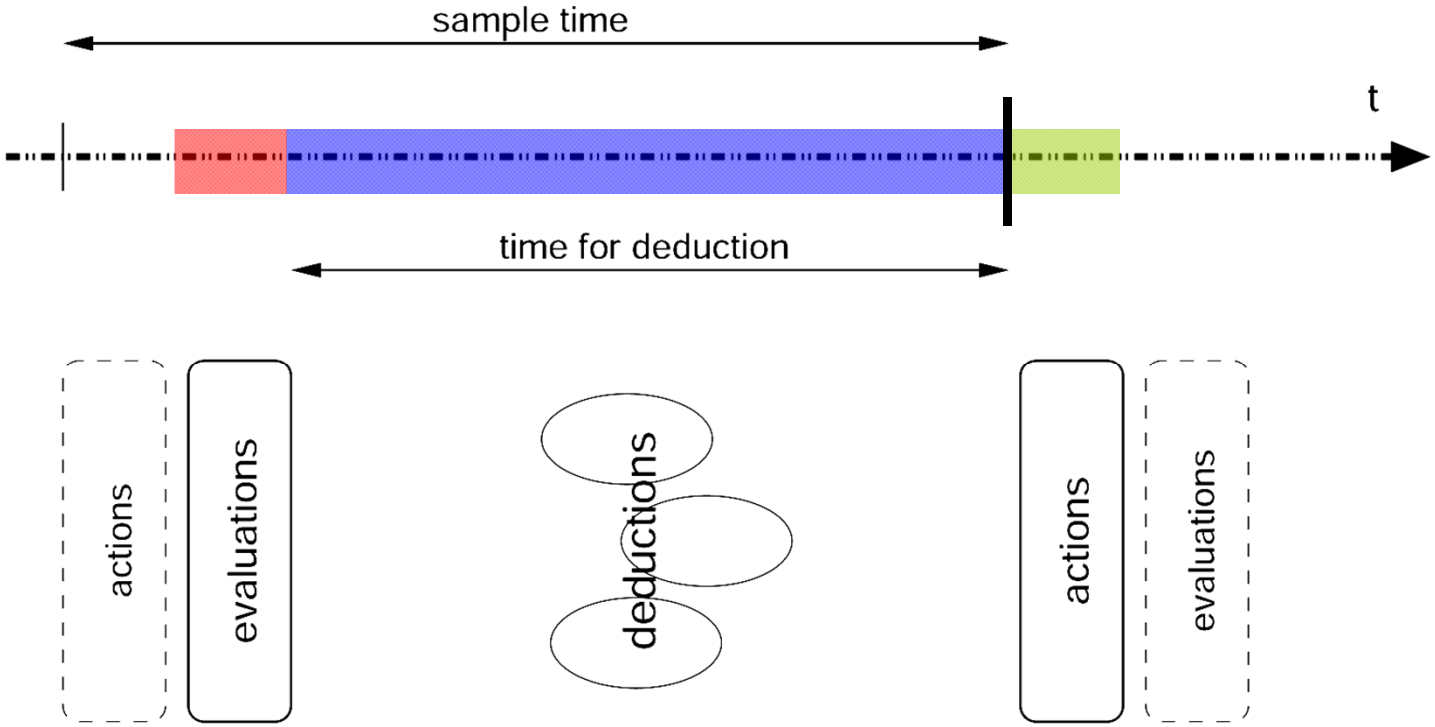
Caratteristiche del controllore

- Cervello logico
- Cuore tempo-reale
- Sensi ed arti procedurali



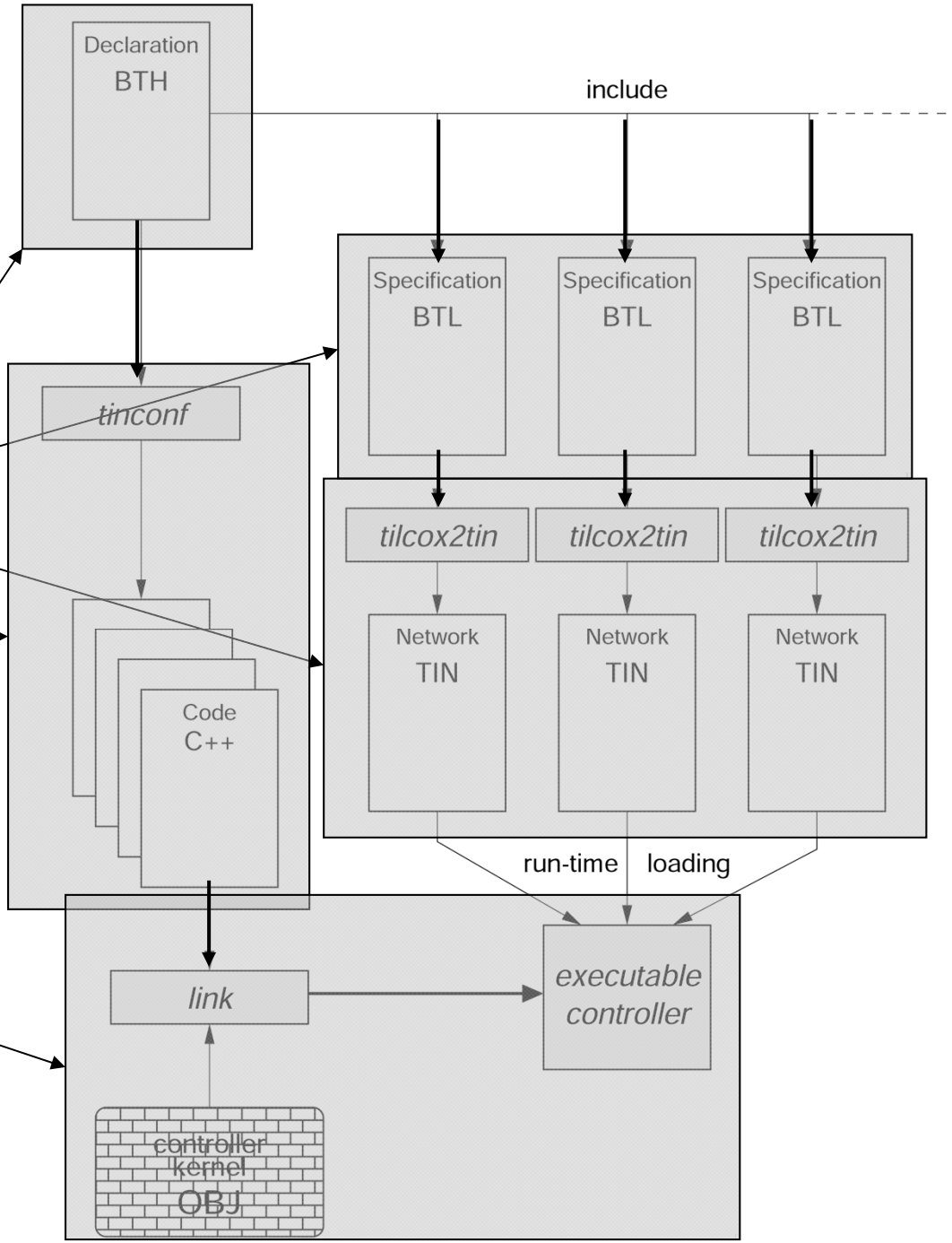
Modello di controllo 2

- Valutazione: Apposite procedure nell'interfaccia di ingresso campionano lo stato del sistema da controllare
- Deduzione: A partire dagli eventi che si sono verificati si determinano, in base alla specifica del comportamento, quali sono quelli conseguenti
- Azione: Le azioni conseguenti vengono eseguite dall'interfaccia di uscita al nuovo istante di campionamento



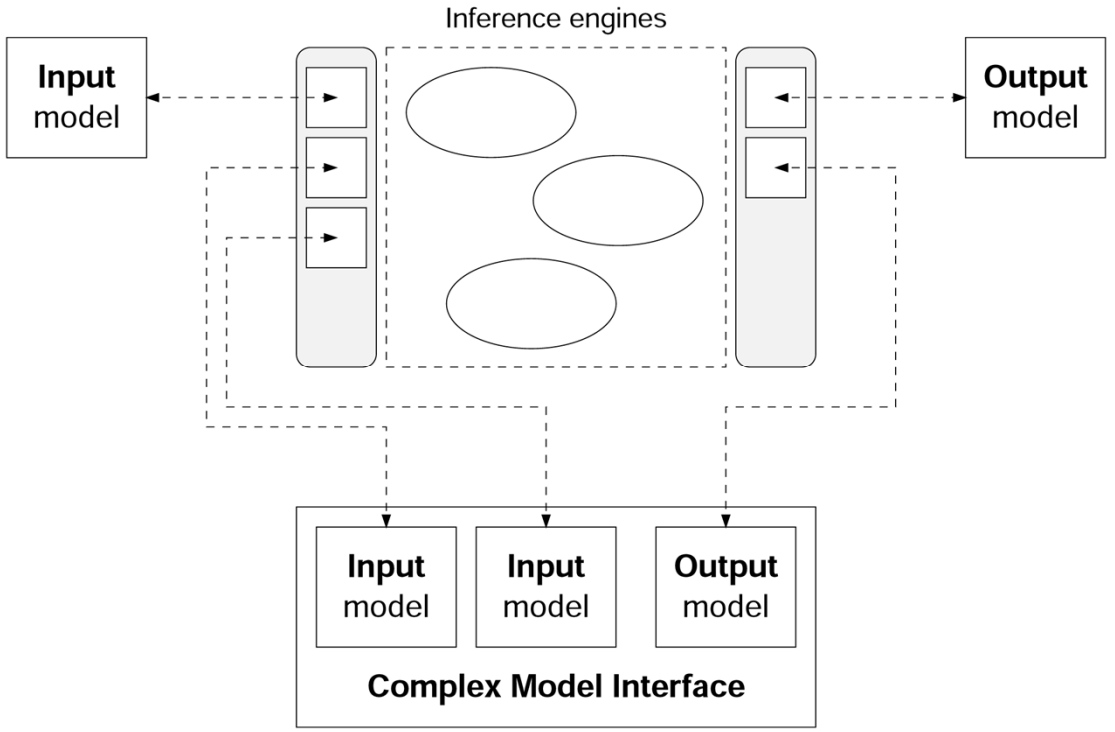
Sistema di sviluppo

- Dichiarazione entità
- Scrittura specifiche
- Compilazione
- Generazione del codice di **configurazione automatica** e *edit* delle procedure
- Link delle sorgenti C++ con la libreria



Controllo dei processi

- Estensione dei modelli di I/O per rappresentare entità il cui stato può essere letto da dispositivi di ingresso e modificato da azioni in uscita dalla rete di inferenza



Processi		Semafori
start, kill	→	*signal*
running, succeeded	←	*waiting*

Caso di studio (descrizione)

Controllo delle lanterne semaforiche
presso un incrocio stradale

- Sequenza delle luci per il flusso alternato dei veicoli
- Interruttore manuale di disabilitazione (lampeggiamento)
- Accettazione di richiesta di cambio di flusso stradale da parte di veicoli di emergenza
- Controllo di pannelli informativi per pubblicità e viabilità integrato al flusso stradale in condizioni di sicurezza
  - Consente la visualizzazione di spot pubblicitari solo dove i veicoli sono fermi in attesa di ripartire
  - Mostra indicazioni utili alla viabilità in caso di veicoli in movimento

Caso di studio (specifica) 1

Sequenza nominale degli stati

$$go_A \Leftrightarrow since\left({}^{up}go_A, {}^{up}warning_A\right)$$
$$warning_A \Leftrightarrow since\left({}^{up}warning_A, {}^{up}alert_A\right)$$
$$alert_A \Leftrightarrow since\left({}^{up}alert_A, {}^{up}stop_A\right)$$
$$stop_A \Leftrightarrow since\left({}^{up}stop_A, {}^{up}go_B\right)$$
$$\ldots$$

Ordine

Tempi

$${}^{up}go_A \Leftrightarrow stop_B @\left[-\mathrm{STOP\_TIME}, 0\right)$$
$${}^{up}warning_A \Leftrightarrow go_A @\left[-\mathrm{GO\_TIME\_A}, 0\right)$$
$$\ldots$$

$$green_A \Leftrightarrow go_A$$
$$yellow_A \Leftrightarrow warning_A \vee alert_A$$
$$red_A \Leftrightarrow stop_A \vee go_B \vee warning_B$$
$$red\_yellow_A \Leftrightarrow alert_B \vee stop_B$$

Stati <> Preset luci

Segnali luci

$$green\_signal_A \Leftrightarrow green_A$$
$$yellow\_signal_A \Leftrightarrow yellow_A \vee red\_yellow_A \vee yellow\_blink_A$$
$$red\_signal_A \Leftrightarrow red_A \vee red\_yellow_A$$

Caso di studio (specifica) 2

Lampeggiamento, richieste di emergenza, gestione pannelli

$$^{up}warning_A \Leftrightarrow (go_A @[-\text{GO\_TIME\_A}, 0) \wedge \neg emergency\_req_A) \vee$$
$$(go_A @[-\text{EMERGENCY\_TIME}, 0) \wedge \neg emergency\_req_A \wedge emergency\_req_B)$$

Emergenza

$$blink\_tick \Leftrightarrow (disabled \wedge \neg blink\_tick @[-\text{BLINK\_RATE}, 0)) \vee {}^{up}disabled$$
$$yellow\_blink \Leftrightarrow disabled \wedge blink\_tick\ ?(-\text{BLINK\_TIME}, 0]$$

Lampeggiamento

$$^{safe}kill(info_A) \Leftrightarrow \text{since}(^{up}stop_A \vee {}^{up}disabled, running(info_A))$$
$$^{safe}start(commerce_A) \Leftrightarrow \neg disabled \wedge \text{since}(\text{down}(^{safe}kill(info_A)), {}^{safe}kill(commerce_A))$$

Pannelli Informativi

Caso di studio (sviluppo)

- **Dichiarazione** per la generazione automatica del codice
  - ← 1 richiesta disabilitazione
  - ← 2 richieste di emergenza
  - → 6 lanterne semaforiche
  - → 4 processi per il controllo dei pannelli

Questo file dichiarativo è incluso nella specifica per garantire l'utilizzo corretto dei nomi (**consistenza**)

C:\...\src\traffic_light.bth

```
input   disable_request;

input   emergency_request_1,
        emergency_request_2;

output  green_signal_1,
        yellow_signal_1,
        red_signal_1;

output  green_signal_2,
        yellow_signal_2,
        red_signal_2;

thread  info_1,
        info_2;

thread  commerce_1,
        commerce_2;

#ifndef __LOGIC
logic traffic(50);
#endif
```

- **Personalizzazione** degli I/O
  - editing dei metodi "vuoti" generati dal sistema di sviluppo

C:\...\traffic_light_custom.h

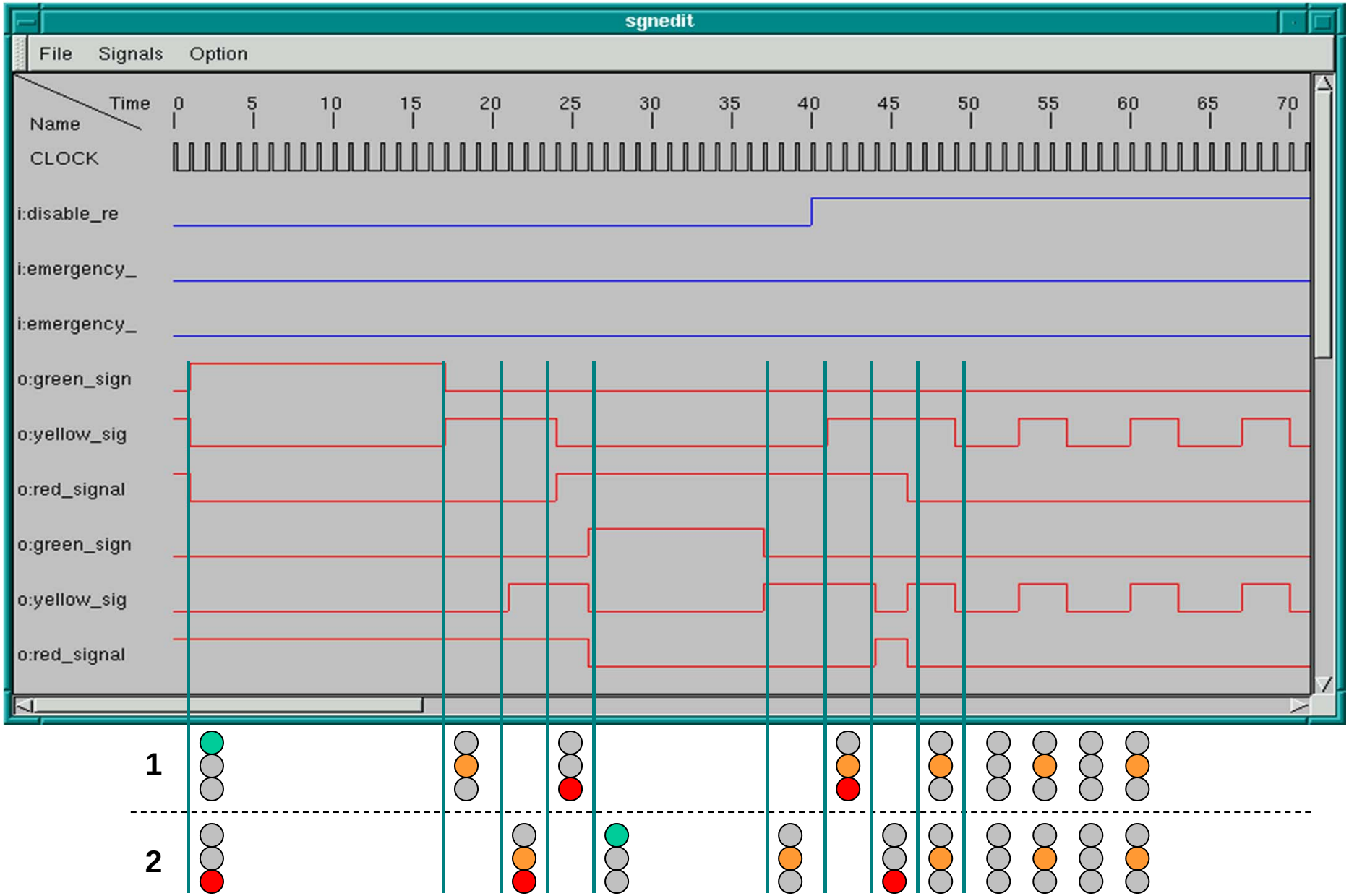
```
class Output_green_signal_1 : public Output
{
private
    LightDevice Light;
public:
    Output_green_signal_1() :
        Output("green_signal_1") :
        Light(GREEN_COLOR) {}
    void action(bool val);
};
```

traffic_light_custom.cpp

```
#include "traffic_light.h"

void Output_green_signal_1::action(bool val)
{
    if(val)
        Light.SwitchOn();
    else
        Light.SwitchOff();
}
```

Caso di studio (esecuzione)



Conclusioni

- L’ambiente di sviluppo integrato per sistemi in tempo-reale mantiene separati i due contesti di programmazione, ma assicura un legame consistente tra le due diverse realtà
- La logica temporale TILCO risulta efficace per scrivere vincoli temporali di precedenza e time-out, ma può essere ulteriormente arricchita con librerie di costrutti predefiniti (macro) per modellare situazioni utili al contesto di sviluppo come periodicità e mutua esclusione
- L’ambiente di sviluppo appare ancora poco amichevole, perciò stiamo creando un nuovo strumento che aiuti visualmente il programmatore in tutti i passi dello sviluppo, ad esempio una visione di insieme dei processi in gioco o delle specifiche