L'uso delle formule logiche come formalismo descrittivo

- Logica: formalismo "universale" (molto vicino al linguaggio naturale)
- Applicabile a contesti molto vari (non solo informatici) (del resto il confine tra computer engineering e system engineering è molto labile)
- Sulla logica sono basati molti formalismi applicativi (dai linguaggi di programmazione a quelli di specifica)

1. La logica per definire linguaggi

- In realtà l'abbiamo già fatto:

  $L = \{a^n b^n \mid n \geq 1\}$

  altro non è che un'abbreviazione della formula del prim'ordine

  $x \in L \leftrightarrow \exists n(n \geq 1 \wedge x = a^n b^n)$

  dove a sua volta l'operazione x^n è definita dalla formula

  $\forall n\ ((n = 0 \to x^n = \varepsilon) \wedge (n > 0 \to x^n = x^{n-1}.x))$

  sulla base del (simbolo di) operazione elementare "."

- Similmente $L_1 = a^*b^*$ è definita da:

$$x \in L_1 \leftrightarrow (x = \varepsilon) \lor$$
$$\exists y(x = ay \land y \in L_1) \lor$$
$$\exists y(x = yb \land y \in L_1)$$

- Posto $L_2 = b^*c^*$ e definito in maniera simile
- $L_3 = a^*b^*c^*$ ($= L_1.L_2$) è definito *anche* come:

$$x \in L_3 \leftrightarrow (x \in L_1) \lor (x \in L_2) \lor$$
$$\exists y(\,(x = ay \land (y \in L_2 \lor y \in L_3)) \lor$$
$$(x = yc \land (y \in L_3 \lor y \in L_1))\,)$$

- $L_4 = \{x|\#x_a = \#x_b\}$ con $\#x_a$ definita da:

$$(x = \varepsilon \ \rightarrow \#x_a = 0) \land$$
$$(x = ay \rightarrow \#x_a = \#y_a + 1) \land$$
$$(x = by \rightarrow \#x_a = \#y_a)$$

(con quantificazione implicita $\forall x\ \forall y$)

2. La logica per definire proprietà dei programmi

- Specifica di un algoritmo di ricerca:
  La variabile logica *found* deve essere vera se e solo se esiste un elemento dell'array a, di n elementi, uguale all'elemento cercato x:

$$found \leftrightarrow \exists i(1 \leq i \leq n \land a[i] = x)$$

- Specifica di un algoritmo di inversione di un array:

$$\forall i(1 \leq i \leq n \rightarrow b[i] = a[n - i + 1])$$

Più in generale

- {Precondizione: *Pre*}
  Programma - o frammento di programma - *P*
  {Postcondizione: *Post*}
- Se vale *Pre* prima dell'esecuzione di *P* si vuole che *P* sia tale da far valere *Post* dopo la sua esecuzione:
- Ricerca in un array ordinato:
  $\{\forall i(1 \leq i < n \rightarrow a[i] \leq a[i+1])\}$
  P
  $\{found \leftrightarrow \exists i(1 \leq i \leq n \wedge a[i] = x)\}$
  NB: ciò non significa affatto che P debba essere un algoritmo di ricerca binaria. Significa solo che chi lo realizza può sfruttare il fatto che *a* sia ordinato prima dell'esecuzione di P. Un normale algoritmo di ricerca sequenziale sarebbe corretto rispetto a questa specifica; al contrario un algoritmo di ricerca binaria non sarebbe corretto rispetto ad una specifica che avesse come precondizione semplicemente *True*.

- Ordinamento di un array di *n* elementi senza ripetizioni:
  $\{\neg \exists i, j(1 \leq i \leq n \wedge 1 \leq j \leq n \wedge i \neq j \wedge a[i] = a[j])\}$
  ORD
  $\{\forall i(1 \leq i < n \rightarrow a[i] \leq a[i+1])\}$
  E' una specifica "adeguata"?
  (Pensiamo all'analogia "specifica = contratto")

$\{\neg \exists i, j(1 \leq i \leq n \wedge 1 \leq j \leq n \wedge i \neq j \wedge a[i] = a[j]) \wedge$
$\forall i(1 \leq i \leq n \rightarrow a[i] = b[i])\}$
ORD
$\{\forall i(1 \leq i < n \rightarrow a[i] \leq a[i + 1]) \wedge$
$\forall i(1 \leq i \leq n \rightarrow \exists j(1 \leq j \leq n \wedge a[i] = b[j])) \wedge$
$\forall j(1 \leq j \leq n \rightarrow \exists i(1 \leq i \leq n \wedge a[i] = b[j]))\}$

- E se eliminiamo la prima riga della precondizione la specifica è ancora valida?
- Siamo sicuri che il problema dell'ordinamento venga sempre inteso alla stessa maniera, sia che si tratti di ordinare un array o una lista o un file?
- In realtà anche un concetto ben noto come l'ordinamento è esposto a imprecisioni ed equivoci nell'uso informale del termine
- Pensiamo a requisiti del tipo "vogliamo automatizzare il rilascio di certificati, o la gestione dei cc bancari"

3. La logica per la specifica di proprietà di sistemi

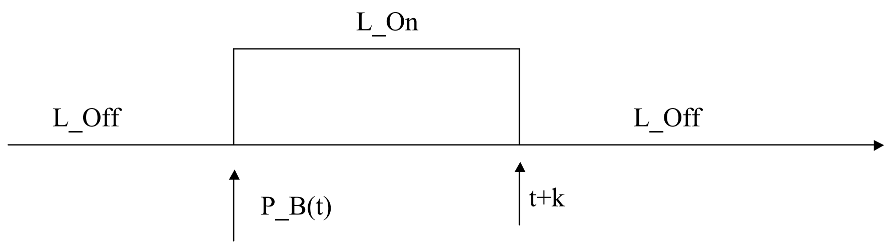
- "Se premo il pulsante si accende la luce entro Δ istanti":
  - P_B(t): Predicato che indica la pressione del pulsante all'istante t
  - L_On(t): Predicato che indica che all'istante t la luce è accesa

$$\forall t(P\_B(t) \rightarrow \exists t_1((t \leq t_1 \leq t + \Delta) \wedge L\_On(t_1)))$$

In realtà una specifica siffatta lascia molto a desiderare rispetto a quanto normalmente si chiede ad un pulsante di accensione della luce.

Scendiamo in qualche dettaglio

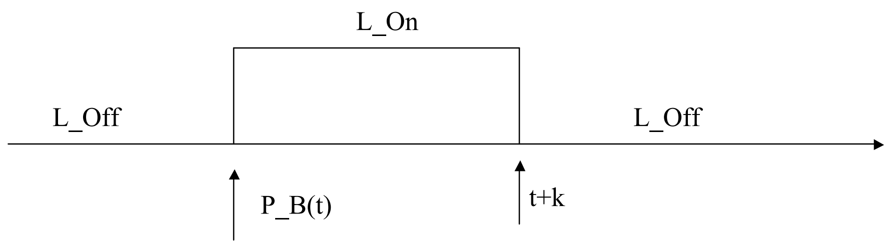
- Focalizziamo l'attenzione su un classico "pulsante a tempo" per la luce (se eccita di più la fantasia: un allarme e/o chiusura a tempo per cassaforti)



$$\forall t(P\_B(t) \rightarrow \forall t_1((t \leq t_1 < t+k) \rightarrow L\_On(t_1)) \wedge \forall t_2((t+k \leq t_2) \rightarrow L\_Off(t_2)))$$

- In realtà ci sono ancora molte cose che non vanno …
  un po' di caccia all'errore ...

Proviamo questa:



$$\forall t((P\_B(t) \wedge L\_Off(t)) \rightarrow \forall t_1((t \leq t_1 < t+k) \rightarrow L\_On(t_1)) \wedge L\_Off(t+k))$$

$$\wedge$$

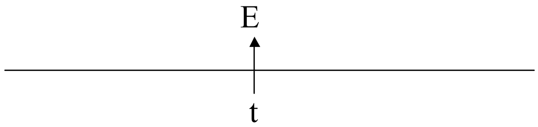
$$\forall t_3, t_4(L\_Off(t_3) \wedge \forall t_5((t_3 \leq t_5 \leq t_4) \rightarrow \neg P\_B(t_5)) \rightarrow L\_Off(t_4))$$

11

Un approccio un po' più sistematico (utile soprattutto per modelli a tempo continuo)

- Event_E: notazione abbreviata per l'assioma seguente:

$$\forall t(E(t) \rightarrow \exists\delta(\forall t_1((t-\delta < t_1 < t \vee t < t_1 < t+\delta) \rightarrow \neg E(t_1))))$$



$\delta > 0$

- Up_to_now_P(t): notazione abbreviata per l'assioma seguente:

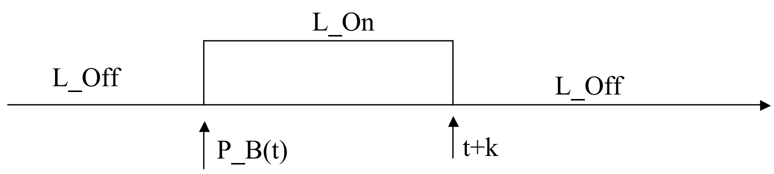
$$\exists\delta(\forall t_1(t-\delta < t_1 < t \rightarrow P(t_1)))$$

- From_now_on_P(t): notazione abbreviata per l'assioma seguente:

$$\exists\delta(\forall t_1(t < t_1 < t+\delta \rightarrow P(t_1)))$$

12

Tornando al nostro timer:



$$\forall t(L\_On(t) \leftrightarrow \neg L\_Off(t)) \qquad \text{(discutibile)} \qquad \wedge$$

$$Event\_P\_B \qquad \wedge$$

$$\forall t((P\_B(t) \wedge Up\_to\_now\_L\_Off(t))$$
$$\rightarrow \forall t_1(t \le t_1 < t+k \rightarrow L\_On(t_1)) \wedge L\_Off(t+k)) \qquad \wedge$$

$$\forall t_3,t_4(L\_Off(t_3) \wedge \forall t_5(t_3 \le t_5 \le t_4 \rightarrow \neg P\_B(t_5))$$
$$\rightarrow L\_Off(t_4))$$

13

Variazioni sul tema:

- Pulsante di spegnimento
- Mantenimento luce accesa mediante pressione durante l'accensione
- Apertura e chiusura tende/paratoie/finestrini auto, …
  - Mantenimento pressione pulsanti
  - Interruzione o non interruzione movimento in corso
  - …..
- Generalità e sistematicità dell'approccio
- Verso *metodi e linguaggi di specifica*

14

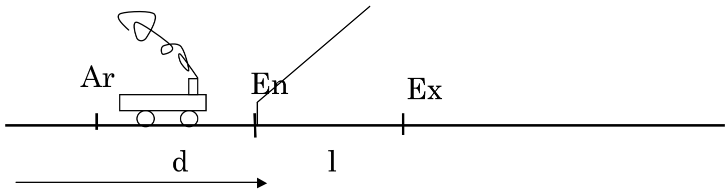
Dalla specifica alla prova: un cenno

- Dopo aver *specificato* i requisiti di un algoritmo (e.g., di ordinamento) e dopo aver *costruito* un tale algoritmo, occorre *verificare* la correttezza del medesimo:

  se ho a disposizione un modello matematico (e.g., *un'assiomatizzazione*) dell'implementazione costruita, in linea di principio posso ottenere la prova di correttezza come una *dimostrazione di teorema*.
- Similmente a livello di sistema ...

15

Consideriamo un passaggio a livello (ultra-semplificato)



- ➢Un solo binario e una sola direzione.
- ➢Formalizzazione relativa al passaggio di un solo treno.
- ➢Abbassamento ed innalzamento delle sbarre istantanei.

- •Event_Ar
- •Event_En
- •Event_Ex

16

Dinamica del treno:

$$(\delta_1 = d/V_{\min} \wedge \delta_2 = \frac{d+l}{V_{\max}}) \wedge (\delta_3 = l/V_{\min}) \wedge (\delta_4 = l/V_{\max}) \wedge (\delta_{\min} = d/V_{\max}) \wedge (\delta_{\max} = \frac{d+l}{V_{\min}}) \wedge$$
$$\forall t(Ar(t) \rightarrow \exists t_1(En(t_1) \wedge (t + \delta_{\min} \leq t_1 \leq t + \delta_1)) \wedge \exists t_2(Ex(t_2) \wedge (t + \delta_2 \leq t_2 \leq t + \delta_{\max}))) \wedge$$
$$\forall t(En(t) \rightarrow \exists t_1(Ar(t_1) \wedge (t - \delta_1 \leq t_1 \leq t - \delta_{\min}))) \wedge$$
$$\forall t(Ex(t) \rightarrow \exists t_1(Ar(t_1) \wedge (t - \delta_{\max} \leq t_1 \leq t - \delta_2))) \wedge$$
$$\forall t(In(t) \leftrightarrow \exists t_1(En(t_1) \wedge (t_1 \leq t)) \wedge \neg \exists t_2(Ex(t_2) \wedge (t_2 \leq t)))$$

“Progetto” del passaggio a livello

$$\forall t(Ar(t) \rightarrow \forall t_1((t + \delta_{\min} \leq t_1 \leq t + \delta_{\max}) \rightarrow Down(t_1))) \wedge$$
$$\forall t(Down(t) \rightarrow \exists t_1((t - \delta_{\max} \leq t_1 \leq t - \delta_{\min}) \wedge Ar(t_1))) \wedge$$
$$\forall t(Down(t) \leftrightarrow \neg Up(t))$$

Specifica del requisito di sicurezza:

$$\forall t(In(t) \rightarrow Down(t))$$

A questo punto il requisito può (e dovrebbe) essere *dimostrato* come *teorema* derivato dalla formalizzazione del sistema (controllore/controllato) … in corsi successivi