Real-Time Systems

Lecture 09: PLC Automata

2013-05-29

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

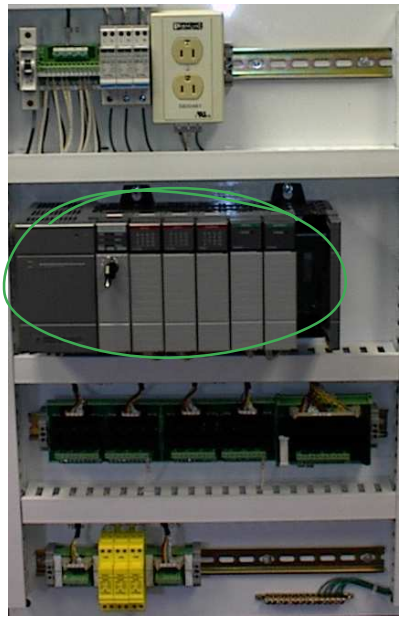
- DC Implementables.
- A controller for the gas burner.

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is the "philosophy" of PLC? What did we generalise/abstract them to?
  - What's an example for giving a DC semantics for a constructive formalism?
  - How does the proposed approach work, from requirements to a correct implementation with DC?
- **Content:**
  - Programmable Logic Controllers (PLC) ("Speicherprogrammierbare Steuerungen" (SPS))
  - PLC Automata
  - An overapproximating DC semantics for PLCA
  - An reaction time theorem for PLCA

What is a PLC?

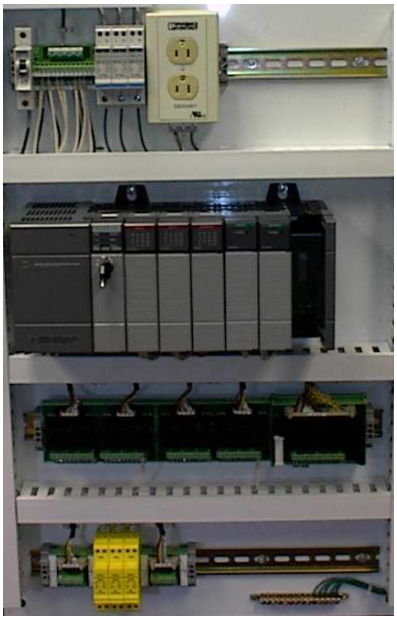
How do PLC look like?



http://wikimedia.org (public domain)

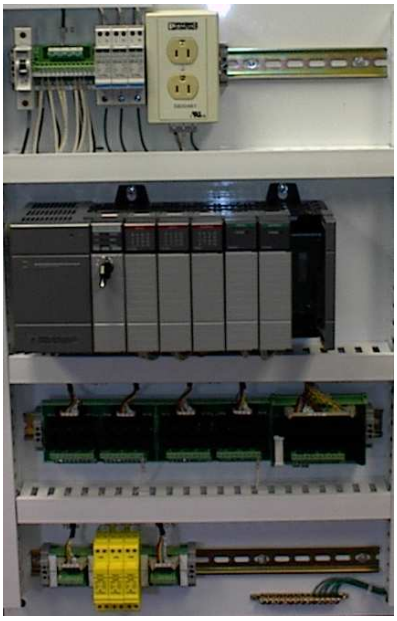
http://wikimedia.org (CC nc-sa 2.5, Ulli1105)

What's special about PLC?



- microprocessor, memory, **timers**
- digital (or analog) I/O ports
- possibly RS 232, fieldbuses, networking
- robust hardware
- reprogrammable
- **standardised programming model** (IEC 61131-3)

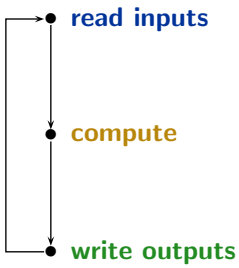
Where are PLC employed?



- mostly **process automatisation**
  - production lines
  - packaging lines
  - chemical plants
  - power plants
  - electric motors, pneumatic or hydraulic cylinders
  - ...
- not so much: **product automatisation**, there
  - tailored or OTS controller boards
  - embedded controllers
  - ...

How are PLC programmed?

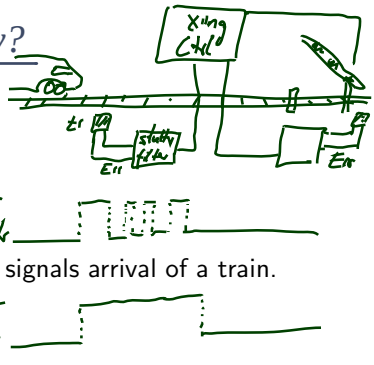
- PLC have in common that they operate in a cyclic manner:



- Cyclic operation is repeated until external interruption (such as shutdown or reset).
- Cycle time: typically a few milliseconds. [?]

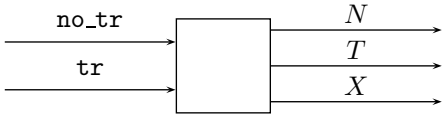
- Programming for PLC means providing the "**compute**" part.
- Input/output values are available via designated local variables.

How are PLC programmed, practically?



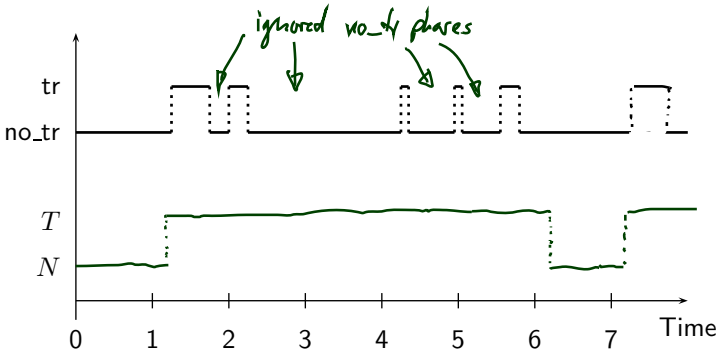
- **Example:** reliable, stutter-free train sensor.
  - Assume a track-side sensor with outputs:
    - `no_tr` — "no passing train"
    - `tr` — "a train is passing"
  - Assume that a change from "`no_tr`" to "`tr`" signals arrival of a train. (No spurious sensor values.)

  - **Problem:** the sensor may **stutter**, i.e. oscillate between "`no_tr`" and "`tr`" multiple times.
  - **Idea:** a stutter **filter** with outputs N and T, for "no train" and "train passing" (and possibly X, for error).
    After arrival of a train, ignore "`no_tr`" for 5 seconds.



Example: Stutter Filter

- **Idea:** After arrival of a train, ignore "no_tr" for 5 seconds.



How are PLC programmed, practically?

read inputs
compute
write outputs

```
 1: PROGRAM PLC_PRG_FILTER
 2: VAR
 3:   state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
 4:   tmr   : TP;
 5: ENDVAR
 6:
 7: IF state = 0 THEN
 8:     %output := N;
 9:     IF %input = tr THEN
10:         state := 1;
11:         %output := T;
12:     ELSIF %input = Error THEN
13:         state := 2;
14:         %output := X;
15:     ENDIF
16: ELSIF state = 1 THEN
17:     tmr( IN := TRUE, PT := t#5.0s );
18:     IF (%input = no_tr AND NOT tmr.Q) THEN
19:         state := 0;
20:         %output := N;
21:         tmr( IN := FALSE, PT := t#0.0s );
22:     ELSIF %input = Error THEN
23:         state := 2;
24:         %output := X;
25:         tmr( IN := FALSE, PT := t#0.0s );
26:     ENDIF
27: ENDIF
```

time

could more loose and have same effect, decreases ignore time & bug

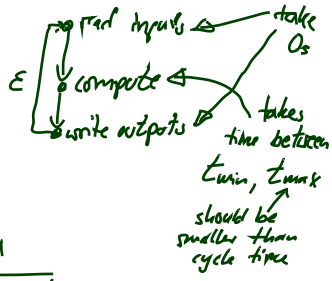
start/set time

duration

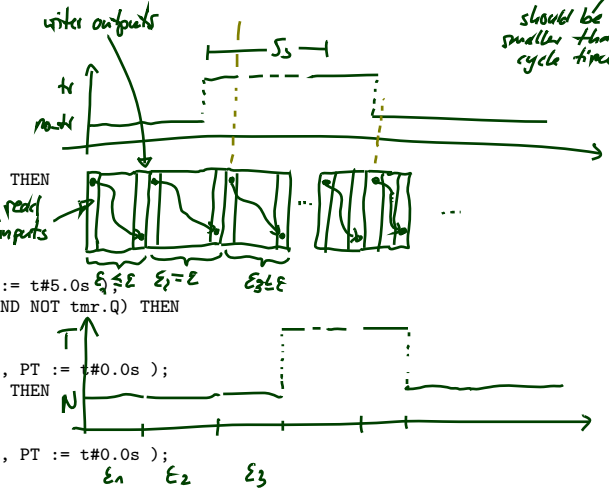
intuitive semantics:
- do assignment
- if assignment changed IN from FALSE to TRUE ("rising edge on IN") then set tmr to given duration (IN is initially FALSE)

TRUE iff tmr is still running (here: if 5s not yet elapsed)

How are PLC programmed, practically?




```
 1: PROGRAM PLC_PRG_FILTER
 2: VAR
 3:   state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
 4:   tmr   : TP;
 5: ENDVAR
 6:
 7: IF state = 0 THEN
 8:     %output := N;
 9:     IF %input = tr THEN
10:         state := 1;
11:         %output := T;
12:     ELSIF %input = Error THEN
13:         state := 2;
14:         %output := X;
15:     ENDIF
16: ELSIF state = 1 THEN
17:     tmr( IN := TRUE, PT := t#5.0s );
18:     IF (%input = no_tr AND NOT tmr.Q) THEN
19:         state := 0;
20:         %output := N;
21:         tmr( IN := FALSE, PT := t#0.0s );
22:     ELSIF %input = Error THEN
23:         state := 2;
24:         %output := X;
25:         tmr( IN := FALSE, PT := t#0.0s );
26:     ENDIF
27: ENDIF
```



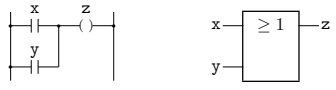
Alternative Programming Languages by IEC 61131-3

```
LD  x
OR  y
ST  z
```

Instruction List

```
z := x OR y
```

Structured Text



(Relay) Ladder Diagram

Function Block Diagram

Figure 2.2: Implementations of the operation "x becomes $y \vee z$" [?]

Tied together by

- Sequential Function Charts (SFC)

Unfortunate: deviations in semantics... [?]

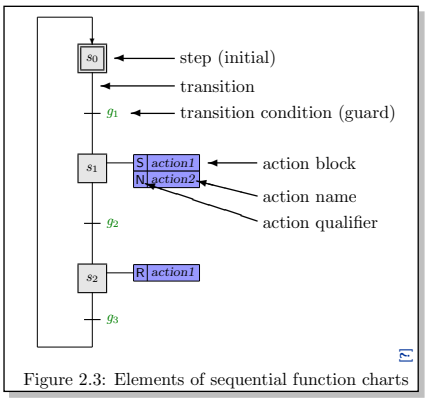


Figure 2.3: Elements of sequential function charts [?]

Why study PLC?

- **Note:**
  the discussion here is **not limited** to PLC and IEC 61131-3 languages.
- Any programming language on an operating system with **at least one** real-time clock will do.
  (Where a **real-time clock** is a piece of hardware such that,
  - we can program it to wait for t time units,
  - we can query whether the set time has elapsed,
  - if we program it to wait for t time units,
    it does so with negligible deviation.)
- And strictly speaking, we don't even need "full blown" operating systems.
- PLC are just a formalisation on a good level of abstraction:
  - there are inputs **somehow** available as local variables,
  - there are outputs **somehow** available as local variables,
  - **somehow**, inputs are polled and outputs updated atomically,
  - there is **some** interface to a real-time clock.

PLC Automata

PLC Automata

Definition 5.2. A **PLC-Automaton** is a structure

$$\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$$

where

- $(q \in)\ Q$ is a finite set of **states**, $q_0 \in Q$ is the **initial state**,
- $(\sigma \in)\ \Sigma$ is a finite set of **inputs**,
- $\delta : Q \times \Sigma \to Q$ is the **transition function** (!),
- $S_t : Q \to \mathbb{R}_0^+$ assigns a **delay time** to each state,
- $S_e : Q \to 2^\Sigma$ assigns a set of **delayed inputs** to each state,
- Ω is a finite, non-empty set of **outputs**,
- $\omega : Q \to \Omega$ assigns an **output** to each state,
- ε is an **upper time bound** for the execution cycle.

PLC Automata Example: Stuttering Filter

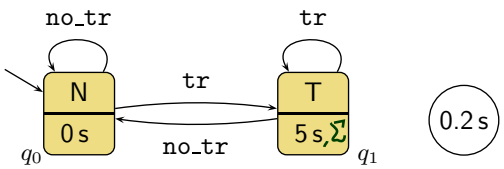
$$\begin{aligned}\mathcal{A} = (&Q = \{q_0, q_1\},\\ &\Sigma = \{\texttt{tr}, \texttt{no\_tr}\},\\ &\delta = \{(q_0, \texttt{tr}) \mapsto q_1, (q_0, \texttt{no\_tr}) \mapsto q_0, (q_1, \texttt{tr}) \mapsto q_1, (q_1, \texttt{no\_tr}) \mapsto q_0\},\\ &q_0 = q_0,\\ &\varepsilon = 0.2,\\ &S_t = \{q_0 \mapsto 0, q_1 \mapsto 5\},\\ &S_e = \{q_0 \mapsto \emptyset, q_1 \mapsto \Sigma\},\\ &\Omega = \{N, T\},\\ &\omega = \{q_0 \mapsto N, q_1 \mapsto T\})\end{aligned}$$

delay time

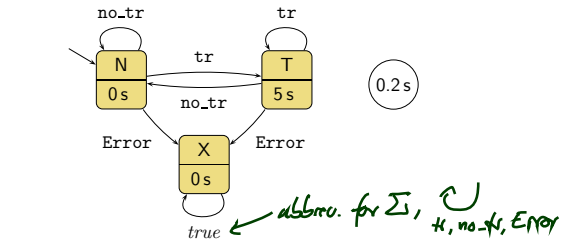
delayed inputs

"in state $q \in Q$, for time $S_t(q)$ after entering q ignore $S_e(q)$"

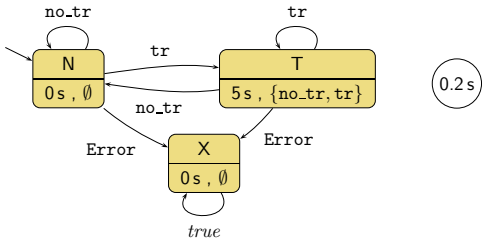
don't take any transition on $S_e(q)$



PLC Automata Example: Stuttering Filter with Exception



abbrev. for Σ, $\circlearrowright$ tr, no_tr, Error

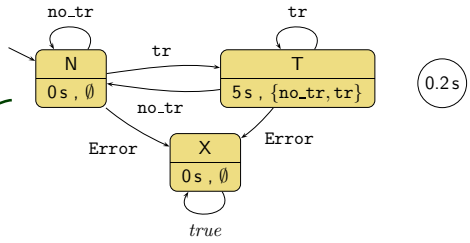


PLC Automaton Semantics

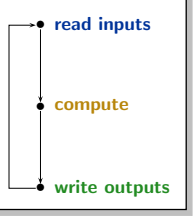
translation to 61131 (?)

```
 1: PROGRAM PLC_PRG_FILTER
 2: VAR
 3:   state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
 4:   tmr   : TP;
 5: ENDVAR
 6:
 7: IF state = 0 THEN
 8:     %output := N;
 9:     IF %input = tr THEN
10:         state := 1;
11:         %output := T;
12:     ELSIF %input = Error THEN
13:         state := 2;
14:         %output := X;
15:     ENDIF
16: ELSIF state = 1 THEN
17:     tmr( IN := TRUE, PT := t#5.0s );
18:     IF (%input = no_tr AND NOT tmr.Q) THEN
19:         state := 0;
20:         %output := N;
21:         tmr( IN := FALSE, PT := t#0.0s );
22:     ELSIF %input = Error THEN
23:         state := 2;
24:         %output := X;
25:         tmr( IN := FALSE, PT := t#0.0s );
26:     ENDIF
27: ENDIF
```

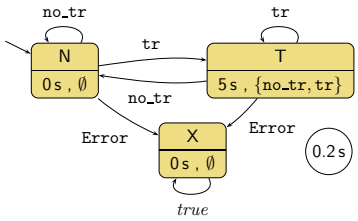
$[\![\cdot]\!]$



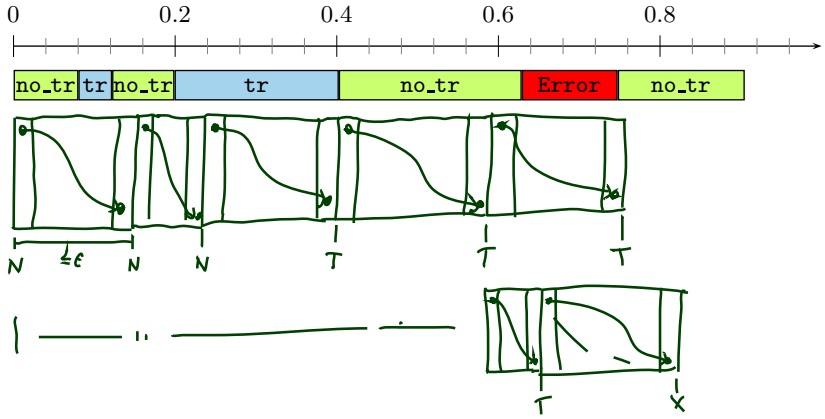
Recall:



PLCA Semantics: Examples




```
1: PROGRAM PLC_PRG_FILTER
2: VAR
3:    state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
4:    tmr   : TP;
5: ENDVAR
6:
7: IF state = 0 THEN
8:     %output := N;
9:     IF %input = tr THEN
10:        state := 1;
11:        %output := T;
12:    ELSIF %input = Error THEN
13:        state := 2;
14:        %output := X;
15:    ENDIF
16: ELSIF state = 1 THEN
17:    tmr( IN := TRUE, PT := t#5.0s );
18:    IF (%input = no_tr AND NOT tmr.Q) THEN
19:        state := 0;
20:        %output := N;
21:        tmr( IN := FALSE, PT := t#0.0s );
22:    ELSIF %input = Error THEN
23:        state := 2;
24:        %output := X;
25:        tmr( IN := FALSE, PT := t#0.0s );
26:    ENDIF
27: ENDIF
```



We assess correctness in terms of cycle time...

...but where does the cycle time come from?

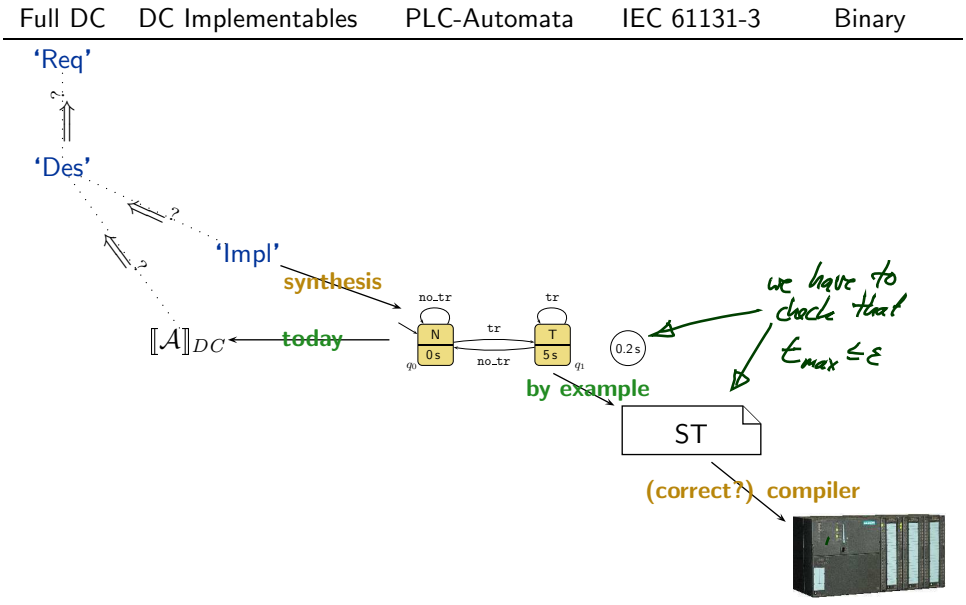
- First of all, ST on the hardware **has** a cycle time
  - so we can **measure** it — if it is larger than ε, don't use this program on this controller
  - we can **estimate** (approximate) the **WCET** (worst case execution time) — if it's larger than ε, don't use it, if it's smaller we're safe

    (Major obstacle: caches, out-of-order execution.)

- Some PLC have a **watchdog**:
  - set it to ε,
  - if the current "computing" cycle takes longer,
  - then the watchdog forces the PLC into an error state and signals the error condition

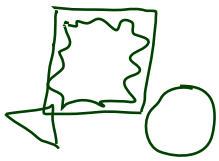
And what does this have to with DC?

Wait, what is the Plan?



An Overapproximating DC Semantics for PLC Automata

Interesting Overall Approach



- Define PLC Automaton syntax (abstract and concrete).
- Define PLC Automaton semantics by translation to ST (structured text).
- Give DC **over-approximation** of PLC Automaton semantics.
- Assess correctness of over-approximation against DC **requirements**.

- **In other words:** we'll define $[\![\mathcal{A}]\!]_{DC}$ such that

$$\text{“}\mathcal{I} \in [\![\mathcal{A}]\!]\text{”} \implies \mathcal{I} \models [\![\mathcal{A}]\!]_{DC}$$

  but not necessarily the other way round.
- **In even other words:** $\text{“}[\![\mathcal{A}]\!]\text{”} \subseteq \{\mathcal{I} \mid \mathcal{I} \models [\![\mathcal{A}]\!]_{DC}\}$.

Observables

- Consider
$$\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega).$$

- The DC formula $[\![\mathcal{A}]\!]_{DC}$ we construct ranges over the observables
  - $\mathsf{In}_{\mathcal{A}}$, $\mathcal{D}(\mathsf{In}_{\mathcal{A}}) = \Sigma$ — values of the **inputs**
  - $\mathsf{St}_{\mathcal{A}}$, $\mathcal{D}(\mathsf{St}_{\mathcal{A}}) = Q$ — current **local state**
  - $\mathsf{Out}_{\mathcal{A}}$, $\mathcal{D}(\mathsf{Out}_{\mathcal{A}}) = \Omega$ — values of the **outputs**

Overview

$$\boxed{\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)}$$

- A arbitrary with $\emptyset \neq A \subseteq \Sigma$,
- $\lceil q \wedge A \rceil$ abbreviates $\lceil \mathsf{St}_{\mathcal{A}} = q \wedge \mathsf{In}_{\mathcal{A}} \in A \rceil$,
- $\delta(q, A)$ abbreviates $\mathsf{St}_{\mathcal{A}} \in \{\delta(q, a) \mid a \in A\}$.

- **Initial State**:
$$\lceil \rceil \vee \lceil q_0 \rceil \,;\, true \qquad \text{(DC-1)}$$

- **Effect of Transitions, untimed**:
$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

- **Cycle time**:
$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

- **Delays**:
$$S_t(q) > 0 \implies \lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \xrightarrow{\leq S_t(q)} \lceil q \vee \delta(q, A \setminus S_e(q)) \rceil \qquad \text{(DC-4)}$$
$$S_t(q) > 0 \implies \lceil \neg q \rceil \,;\, \lceil q \rceil \,;\, \lceil q \wedge A \rceil^{\varepsilon} \xrightarrow{\leq S_t(q)} \lceil q \vee \delta(q, A \setminus S_e(q)) \rceil \qquad \text{(DC-5)}$$

Overview

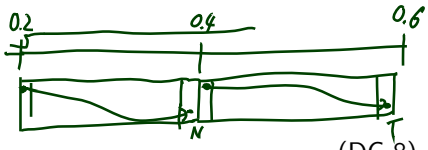
$$\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$$

- A arbitrary with $\emptyset \neq A \subseteq \Sigma$,
- $\lceil q \wedge A \rceil$ abbreviates $\lceil \mathsf{St}_{\mathcal{A}} = q \wedge \mathsf{In}_{\mathcal{A}} \in A \rceil$,
- $\delta(q, A)$ abbreviates $\mathsf{St}_{\mathcal{A}} \in \{\delta(q, a) \mid a \in A\}$.

- **Progress from non-delayed inputs**:

$$S_t(q) = 0 \wedge q \notin \delta(q, A) \implies \Box(\lceil q \wedge A \rceil \implies \ell < 2\varepsilon) \qquad \text{(DC-6)}$$

$$S_t(q) = 0 \wedge q \notin \delta(q, A) \implies \lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil^{\varepsilon} \longrightarrow \lceil \neg q \rceil \qquad \text{(DC-7)}$$



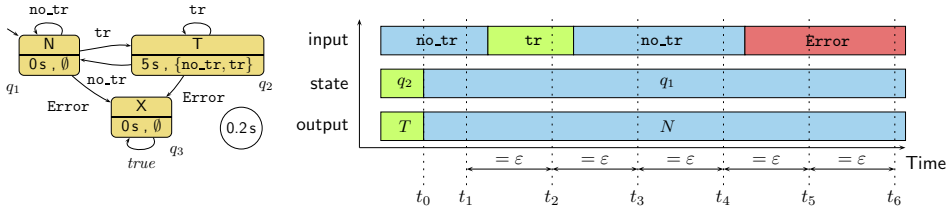
- **Progress from delayed inputs**:

$$S_t(q) > 0 \wedge q \notin \delta(q, A) \implies \Box(\lceil q \rceil^{S_t(q)} \,;\, \lceil q \wedge A \rceil \implies \ell < S_t(q) + 2\varepsilon) \qquad \text{(DC-8)}$$

$$S_t(q) > 0 \wedge A \cap S_e(q) = \emptyset \wedge q \notin \delta(q, A) \implies \Box(\lceil q \wedge A \rceil \implies \ell < 2\varepsilon) \qquad \text{(DC-9)}$$

$$S_t(q) > 0 \wedge A \cap S_e(q) = \emptyset \wedge q \notin \delta(q, A) \implies \lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil^{\varepsilon} \longrightarrow \lceil \neg q \rceil \qquad \text{(DC-10)}$$

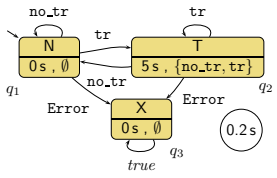
Effect of Transitions, untimed

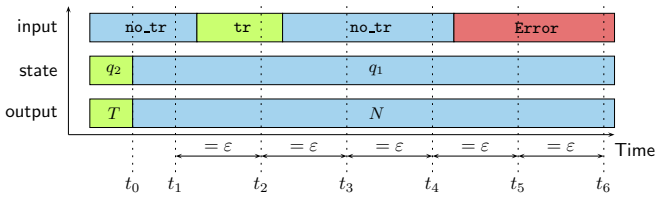


$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

$\lceil q_1 \wedge A \rceil$ holds in	with input	After	state	output
$[t_0, t_1]$	$A = \{\texttt{no\_tr}\}$	t_1	$\{q_1\}$	$\{N\}$
$[t_0, t_2]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_2	$\{q_1\}$	$\{N, T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_3	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_4]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_4	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_5]$	$A = \{\texttt{no\_tr}, \texttt{tr}, \texttt{Error}\}$	t_5	$\{q_1, q_2, q_3\}$	$\{N, T, X\}$
$[t_0, t_6]$	$A = \{\texttt{no\_tr}, \texttt{tr}, \texttt{Error}\}$	t_6	$\{q_1, q_2, q_3\}$	$\{N, T, X\}$

Cycle Time

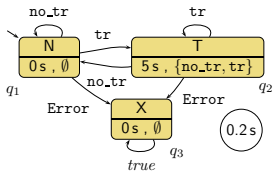


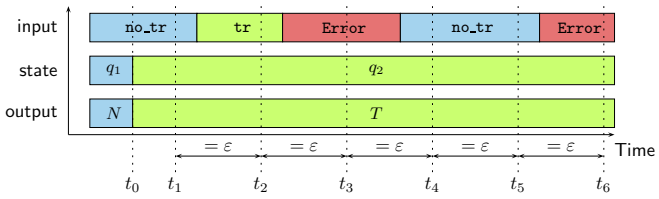


$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

$\lceil q_1 \wedge A \rceil$ holds in	with input	After	state	output
$[t_1, t_2]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_2	$\{q_1, q_2\}$	$\{N, T\}$
$[t_2, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_3	$\{q_1, q_2\}$	$\{N, T\}$
$[t_3, t_4]$	$A = \{\texttt{no\_tr}\}$	t_4	$\{q_1\}$	$\{N\}$
$[t_4, t_5]$	$A = \{\texttt{no\_tr}, \texttt{Error}\}$	t_5	$\{q_1, q_3\}$	$\{N, X\}$
$[t_5, t_6]$	$A = \{\texttt{Error}\}$	t_6	$\{q_1, q_3\}$	$\{N, X\}$

Delays

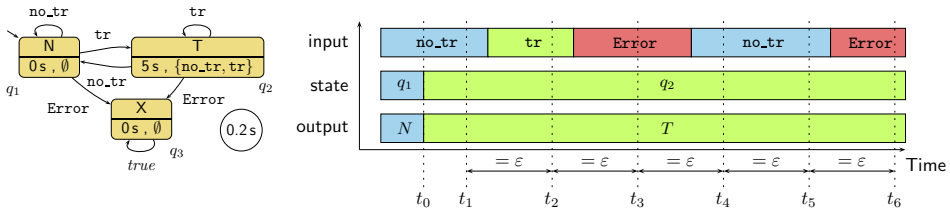




$$S_t(q) > 0 \implies \lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \xrightarrow{\leq S_t(q)} \lceil q \vee \delta(q, A \setminus S_e(q)) \rceil \qquad \text{(DC-4)}$$

$\lceil q_1 \wedge A \rceil$ holds in	with input	After	state	output
$[t_0, t_1]$	$A = \{\texttt{no\_tr}\}$	t_1	$\{q_2\}$	$\{T\}$
$[t_0, t_2]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_2	$\{q_2\}$	$\{T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}, \texttt{Error}\}$	t_3	$\{q_2, q_3\}$	$\{T, X\}$
$[t_0, t_4]$	$A = \{\texttt{no\_tr}, \texttt{tr}, \texttt{Error}\}$	t_4	$\{q_2, q_3\}$	$\{T, X\}$
$[t_0, t_5]$	$A = \{\texttt{no\_tr}, \texttt{tr}, \texttt{Error}\}$	t_5	$\{q_2, q_3\}$	$\{T, X\}$
$[t_0, t_6]$	$A = \{\texttt{no\_tr}, \texttt{tr}, \texttt{Error}\}$	t_6	$\{q_2, q_3\}$	$\{T, X\}$

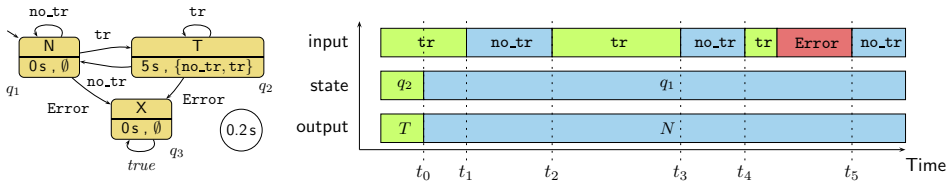
Delays



$$S_t(q) > 0 \implies \lceil \neg q \rceil \,;\, \lceil q \rceil \,;\, \lceil q \wedge A \rceil^{\varepsilon} \xrightarrow{\leq S_t(q)} \lceil q \vee \delta(q, A \setminus S_e(q)) \rceil \quad \text{(DC-5)}$$

$\lceil q_1 \wedge A \rceil$ holds in	with input	After	state	output
$[t_1, t_2]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_2	$\{q_2\}$	$\{T\}$
$[t_2, t_3]$	$A = \{\texttt{tr}, \texttt{Error}\}$	t_3	$\{q_2, q_3\}$	$\{T, X\}$
$[t_3, t_4]$	$A = \{\texttt{no\_tr}, \texttt{Error}\}$	t_4	$\{q_2, q_3\}$	$\{T, X\}$
$[t_4, t_5]$	$A = \{\texttt{no\_tr}\}$	t_5	$\{q_2\}$	$\{T\}$
$[t_5, t_6]$	$A = \{\texttt{no\_tr}, \texttt{Error}\}$	t_6	$\{q_2, q_3\}$	$\{T, X\}$

Progress from non-delayed inputs

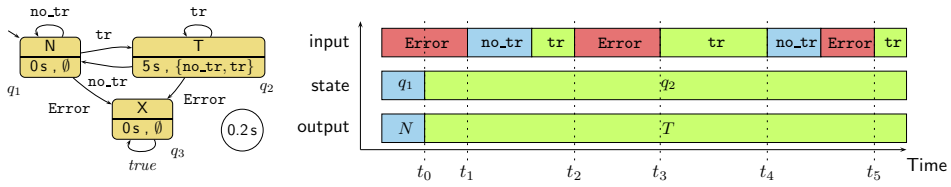


$$S_t(q) = 0 \wedge q \notin \delta(q, A) \implies \Box(\lceil q \wedge A \rceil \implies \ell < 2\varepsilon) \quad \text{(DC-6)}$$

$$S_t(q) = 0 \wedge q \notin \delta(q, A) \implies \lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil^{\varepsilon} \longrightarrow \lceil \neg q \rceil \quad \text{(DC-7)}$$

- Due to (DC-6):
  - $t_5 - t_4 < 2\varepsilon$
  - $t_3 - t_2 < 2\varepsilon$
- Due to (DC-7):
  - $t_1 - t_0 < \varepsilon$

Progress from delayed inputs



$$S_t(q) > 0 \wedge q \notin \delta(q, A) \implies \Box(\lceil q \rceil^{S_t(q)} \,;\, \lceil q \wedge A \rceil \implies \ell < S_t(q) + 2\varepsilon) \quad \text{(DC-8)}$$

$$S_t(q) > 0 \wedge A \cap S_e(q) = \emptyset \wedge q \notin \delta(q, A) \implies \Box(\lceil q \wedge A \rceil \implies \ell < 2\varepsilon) \quad \text{(DC-9)}$$

$$S_t(q) > 0 \wedge A \cap S_e(q) = \emptyset \wedge q \notin \delta(q, A) \implies \lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil^{\varepsilon} \longrightarrow \lceil \neg q \rceil \quad \text{(DC-10)}$$

- Due to (DC-8):
  - $t_5 - t_4 < 2\varepsilon$
- Due to (DC-9):
  - $t_3 - t_2 < 2\varepsilon$
- Due to (DC-10):
  - $t_1 - t_0 < \varepsilon$

Behaviour of the Output and System Start

$$\Box(\lceil q \rceil \implies \lceil \omega(q) \rceil) \quad \text{(DC-11)}$$

$$\lceil q_0 \wedge A \rceil \longrightarrow_0 \lceil q_0 \vee \delta(q_0, A) \rceil \quad \text{(DC-2')}$$

$$S_t(q_0) > 0 \implies \lceil q_0 \wedge A \rceil \xrightarrow{\leq S_t(q_0)}_0 \lceil q_0 \vee \delta(q_0, A \setminus S_e(q_0)) \rceil \quad \text{(DC-4')}$$

$$S_t(q_0) > 0 \implies \lceil q_0 \rceil \,;\, \lceil q_0 \wedge A \rceil^{\varepsilon} \xrightarrow{\leq S_t(q_0)}_0 \lceil q_0 \vee \delta(q_0, A \setminus S_e(q_0)) \rceil \quad \text{(DC-5')}$$

$$S_t(q_0) = 0 \wedge q_0 \notin \delta(q_0, A) \implies \lceil q_0 \wedge A \rceil^{\varepsilon} \longrightarrow_0 \lceil \neg q_0 \rceil \quad \text{(DC-7')}$$

$$S_t(q_0) > 0 \wedge A \cap S_e(q_0) = \emptyset \wedge q_0 \notin \delta(q_0, A) \implies \lceil q_0 \wedge A \rceil^{\varepsilon} \longrightarrow_0 \lceil \neg q_0 \rceil \quad \text{(DC-10')}$$

DC Semantics of PLC Automata

Definition 5.3.
The **Duration Calculus semantics** of a PLC Automaton $\mathcal{A}$ is

$$[\![\mathcal{A}]\!]_{DC} := \bigwedge_{\substack{q \in Q,\\ \emptyset \neq A \subseteq \Sigma}} \text{DC-1} \wedge \cdots \wedge \text{DC-11} \wedge \text{DC-2}' \wedge \text{DC-4}' \wedge \text{DC-5}' \wedge \text{DC-7}' \wedge \text{DC-10}'.$$

Claim:

- Let $P_{\mathcal{A}}$ be the ST program semantics of $\mathcal{A}$.
- Let π be a recording over time of then inputs, local states, and outputs of a PLC device running $P_{\mathcal{A}}$.
- Let $\mathcal{I}_\pi$ be an encoding of π as an interpretation of $\text{In}_{\mathcal{A}}$, $\text{St}_{\mathcal{A}}$, and $\text{Out}_{\mathcal{A}}$.
- Then $\mathcal{I}_\pi \models [\![\mathcal{A}]\!]_{DC}$.
- But not necessarily the other way round.

One Application: Reaction Times

One Application: Reaction Times

- Given a PLC-Automaton, one often wants to know whether it guarantees properties of the form

$$\lceil \mathsf{St}_{\mathcal{A}} \in Q \wedge \mathsf{In}_{\mathcal{A}} = \mathit{emergency\_signal} \rceil \xrightarrow{0.1} \lceil \mathsf{St}_{\mathcal{A}} = \mathit{motor\_off} \rceil$$

("**whenever** the **emergency signal** is observed,
the PLC Automaton switches the **motor off** **within at most** 0.1 seconds")

- Which is (**why?**) for from obvious from the PLC Automaton in general.
- We will give a theorem, that allows us to compute an upper bound on such reaction times.
- Then in the above example, we could simply compare this upper bound one against the required 0.1 seconds.

The Reaction Time Problem in General

- Let
  - $\Pi \subseteq Q$ be a set of **start states**,
  - $A \subseteq \Sigma$ be a set of **inputs**,
  - $c \in \mathsf{Time}$ be a **time bound**, and
  - $\Pi_{target} \subseteq Q$ be a set of **target states**.
- Then we seek to establish properties of the form

$$\lceil \mathsf{St}_{\mathcal{A}} \in \Pi \wedge \mathsf{In}_{\mathcal{A}} \in A \rceil \xrightarrow{c} \lceil \mathsf{St}_{\mathcal{A}} \in \Pi_{target} \rceil,$$

abbreviated as

$$\lceil \Pi \wedge A \rceil \xrightarrow{c} \lceil \Pi_{target} \rceil.$$

Reaction Time Theorem Premises

- Actually, the reaction time theorem addresses **only** the **special case**

$$\lceil \Pi \wedge A \rceil \xrightarrow{c_n} \lceil \underbrace{\delta^n(\Pi, A)}_{=\Pi_{target}} \rceil$$

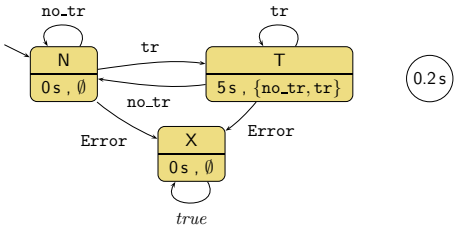
for PLC Automata with

$$\delta(\Pi, A) \subseteq \Pi.$$

- Where the transition function is canonically **extended** to **sets** of start states and inputs:

$$\delta(\Pi, A) := \{\delta(q, a) \mid q \in \Pi \wedge a \in A\}.$$

Premise Examples



Examples:

- $\Pi = \{N, T\},\ A = \{\texttt{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \subseteq \Pi$
- $\Pi = \{N, T, X\},\ A = \{\texttt{Error}\}$
  - $\delta(\Pi, A) = \{X\} \subseteq \Pi$
- $\Pi = \{T\},\ A = \{\texttt{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \not\subseteq \Pi$

Reaction Time Theorem (Special Case $n = 1$)

Theorem 5.6. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$, $\Pi \subseteq Q$, and $A \subseteq \Sigma$ with

$$\delta(\Pi, A) \subseteq \Pi.$$

Then

$$\lceil \Pi \wedge A \rceil \xrightarrow{c} \lceil \underbrace{\delta(\Pi, A)}_{=\Pi_{target}} \rceil$$

where

$$c := \varepsilon + \max(\{0\} \cup \{s(\pi, A) \mid \pi \in \Pi \setminus \delta(\Pi, A)\})$$

and

$$s(\pi, A) := \begin{cases} S_t(\pi) + 2\varepsilon & \text{, if } S_t(\pi) > 0 \text{ and } A \cap S_e(\pi) \neq \emptyset \\ \varepsilon & \text{, otherwise.} \end{cases}$$

Reaction Time Theorem: Example 1

(1) $\lceil \{N, T\} \wedge \{\texttt{no\_tr}\} \rceil \xrightarrow{5+3\varepsilon} \lceil N \rceil$:

Reaction Time Theorem: Example 2

(2) $\lceil \{N, T, X\} \wedge \{\texttt{Error}\} \rceil \xrightarrow{2\varepsilon} \lceil X \rceil$:

Monotonicity of Generalised Transition Function

- Define
$$\delta^0(\Pi, A) := \Pi, \qquad \delta^{n+1}(\Pi, A) := \delta(\delta^n(\Pi, A), A).$$

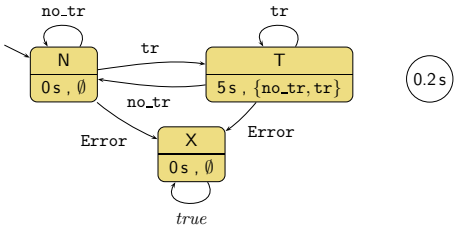
- **If** we have $\delta(\Pi, A) \subseteq \Pi$, then we have
$$\delta^{n+1}(\Pi, A) \subseteq \delta^n(\Pi, A) \subseteq \cdots \subseteq \underbrace{\delta(\delta(\Pi, A), A)}_{=\delta^2(\Pi, A)} \subseteq \delta(\Pi, A) \subseteq \Pi$$
i.e. the sequence is a **contraction**.

- (Because the extended transition function has the following (not so surprising) **monotonicity** property:

Proposition 5.4.

$\Pi \subseteq \Pi' \subseteq Q$ and $A \subseteq A' \subseteq \Sigma$ implies $\delta(\Pi, A) \subseteq \delta(\Pi', A').$)

Contraction Examples



Examples:

- $\Pi = \{N, T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{N\}$
- $\Pi = \{N, T, X\}$, $A = \{\texttt{Error}\}$
  - $\delta^0(\Pi, A) = \{N, T, X\}$
  - $\delta(\delta^0(\Pi, A), A) = \{X\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{X\}$
- $\Pi = \{T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \not\subseteq \Pi$

Reaction Time Theorem (General Case)

Theorem 5.8. Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$, $\Pi \subseteq Q$, and $A \subseteq \Sigma$ with

$$\delta(\Pi, A) \subseteq \Pi.$$

Then for all $n \in \mathbb{N}_0$,

$$\lceil \Pi \wedge A \rceil \xrightarrow{c_n} \lceil \underbrace{\delta^n(\Pi, A)}_{=\Pi_{target}} \rceil$$

where

$$c_n := \varepsilon + \max\left(\{0\} \cup \left\{ \sum_{i=1}^{k} s(\pi_i, A) \;\middle|\; \begin{array}{l} 1 \le k \le n \wedge \\ \exists\, \pi_1, \ldots, \pi_k \in \Pi \setminus \delta^n(\Pi, A) \\ \quad \forall\, j \in \{1, \ldots, k-1\} : \\ \quad\quad \pi_{j+1} \in \delta(\pi_j, A) \end{array} \right\} \right)$$

and $s(\pi, A)$ as before.

Proof Idea of Reaction Time Theorem

(by contradiction)

- Assume, we would **not** have

$$\lceil \Pi \wedge A \rceil \xrightarrow{c_n} \lceil \delta^n(\Pi, A) \rceil.$$

- This is equivalent to **not** having

$$\neg(true \,;\, \lceil \Pi \wedge A \rceil^{c_n} \,;\, \lceil \neg\delta^n(\Pi, A) \rceil \,;\, true)$$

- Which is equivalent to having

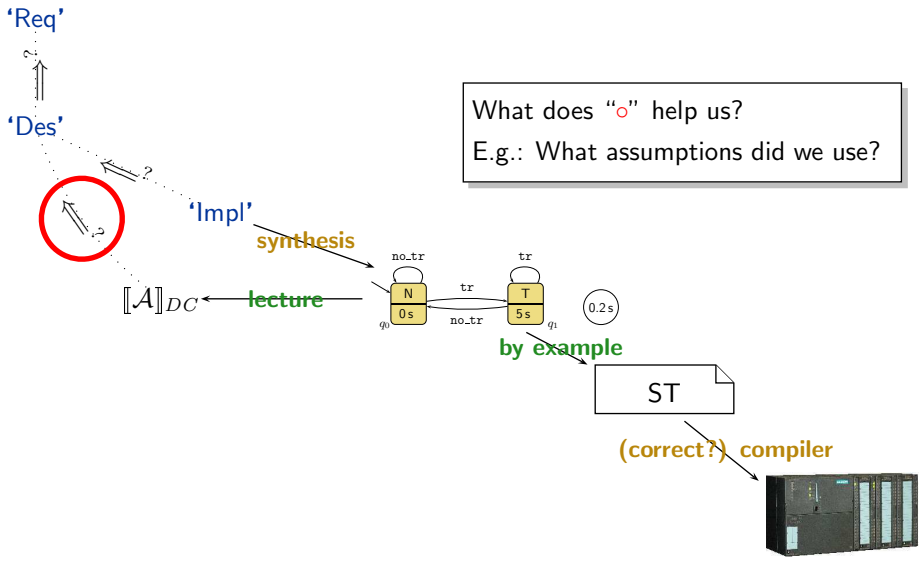
$$true \,;\, \lceil \Pi \wedge A \rceil^{c_n} \,;\, \lceil \neg\delta^n(\Pi, A) \rceil \,;\, true.$$

- Using finite variability, (DC-2), (DC-3), (DC-6), (DC-7), (DC-8), (DC-9), and (DC-10) we can show that the duration of $\lceil \Pi \wedge A \rceil$ is strictly smaller than c_n.

Methodology: Overview

Methodology

Full DC | DC Implementables | PLC-Automata | IEC 61131-3 | Binary



What does "○" help us?
E.g.: What assumptions did we use?

References