Real-Time Systems

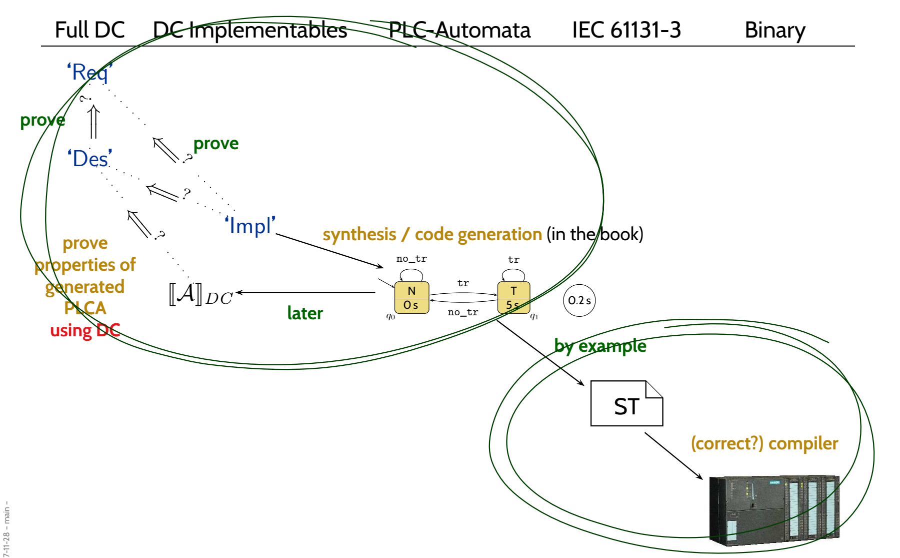
Lecture 10: PLC Automata

2017-11-30

~~Dr. Bernd Westphal~~
Dr. Jochen Hoenicke

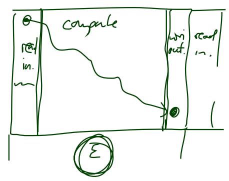
Albert-Ludwigs-Universität Freiburg, Germany

The Plan



How are PLC programmed?

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



- **read inputs**
- **compute**
- **write outputs**

- Cyclic operation is repeated until external interruption (such as shutdown or reset).
- Cycle time: typically a few milliseconds (Lukoschus, 2004).

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

Content

- **Programmable Logic Controllers** (PLC) continued
- **PLC Automata**
  - **Example**: Stutter Filter
  - **PLCA Semantics** by example
  - **Cycle time**
- An **over-approximating** **DC Semantics** for **PLC Automata**
  - **observables**, **DC formulae**
- **PLCA Semantics** at work:
  - effect of **transitions** (untimed),
  - **cycle time**, **delays**, **progress**.
- Application example: **Reaction times**
  - **Examples**: reaction times of the stutter filter

Why Study PLC?

Why study PLC?

- **Note:** the discussion here is **not limited** to PLC and IEC 61131-3 languages.

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.

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.)

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.)

  Strictly speaking, we don't even need a "full blown" operating system.

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.

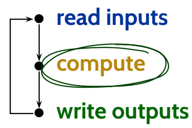
  (Where a **real-time clock** is a piece of hardware such that,

  - 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.)

  Strictly speaking, we don't even need a "full blown" operating system.

- PLC are just **a formalisation** on a **good level of abstraction**:
  - inputs are **somehow** available as local variables,
  - outputs are **somehow** available as local variables,
  - **somehow**, inputs are polled and outputs are updated,
  - there is **some** interface to a real-time clock.

How are PLC programmed, practically?




```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

declare timer `tmr`

duration

intuitive semantics:

- *do the assignment*
- *if assignment changed `IN` from FALSE to TRUE ("rising edge on `IN`") then set `tmr` to given duration (initially, `IN` is FALSE)*

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

Alternative Programming Languages by IEC 61131-3

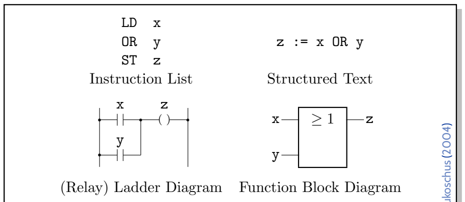


Figure 2.2: Implementations of the operation "x becomes $y \vee z$"

Tied together by

- Sequential Function Charts (SFC)

Unfortunate: deviations in semantics... Bauer (2003)

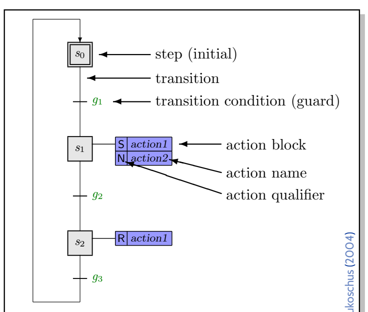


Figure 2.3: Elements of sequential function charts

Content

- **Programmable Logic Controllers** (PLC) continued
- **PLC Automata**
  - **Example**: Stutter Filter
  - **PLCA Semantics** by example
  - **Cycle time**
- An **over-approximating** **DC Semantics** for **PLC Automata**
  - **observables**, **DC formulae**
- **PLCA Semantics** at work:
  - effect of **transitions** (untimed),
  - **cycle time**, **delays**, **progress**.
- Application example: **Reaction times**
  - **Examples**:
    reaction times of the stutter filter

PLC Automata

PLC Automata

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

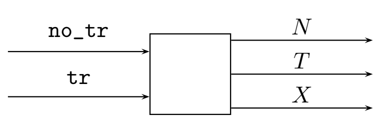
$$\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$$

where

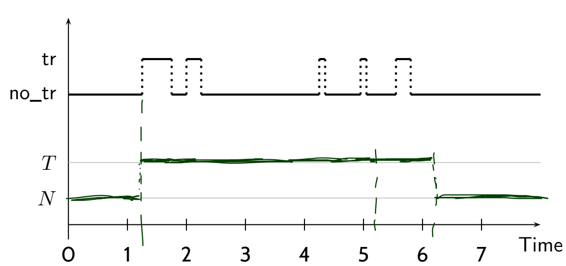
- $(q \in)$ Q is a finite set of **states**, $q_0 \in Q$ is the **initial state**,
- $(\sigma \in)$ Σ is a finite set of **inputs**,
- $\delta : Q \times \Sigma \to Q$ is the **transition function** (!),
- $S_t : Q \to \mathbb{R}_0^+$ assigns a **delay time** to each state,
- $S_e : Q \to 2^\Sigma$ assigns 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.

Example: Stutter Filter

- **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, it should ignore "no_tr" for 5 seconds.

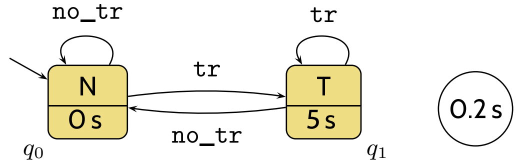


PLC Automata Example: Stuttering Filter

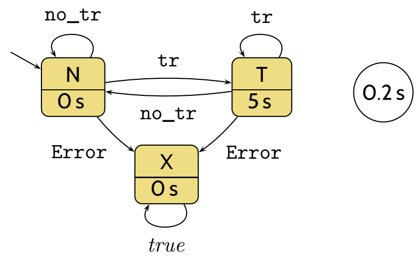
$$\begin{aligned}
\mathcal{A} = (&Q = \{q_0, q_1\},\\
&\Sigma = \{\mathtt{tr}, \mathtt{no_tr}\},\\
&\delta = \{(q_0, \mathtt{tr}) \mapsto q_1, (q_0, \mathtt{no_tr}) \mapsto q_0, (q_1, \mathtt{tr}) \mapsto q_1, (q_1, \mathtt{no_tr}) \mapsto q_0\},\\
&q_0 = q_0,\\
&\varepsilon = 0.2,\\
&S_t = \{q_0 \mapsto 0, q_1 \mapsto 5\},\\
&S_e = \{q_0 \mapsto \emptyset, q_1 \mapsto \Sigma\},\\
&\Omega = \{N, T\},\\
&\omega = \{q_0 \mapsto N, q_1 \mapsto T\})
\end{aligned}$$

PLC Automata Example: Stuttering Filter

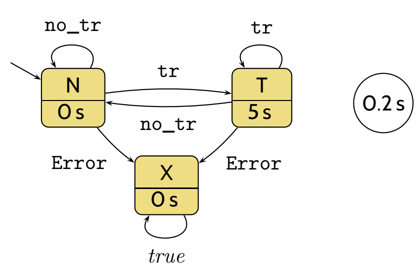
$$\begin{aligned}
\mathcal{A} = (&Q = \{q_0, q_1\},\\
&\Sigma = \{\mathtt{tr}, \mathtt{no_tr}\},\\
&\delta = \{(q_0, \mathtt{tr}) \mapsto q_1, (q_0, \mathtt{no_tr}) \mapsto q_0, (q_1, \mathtt{tr}) \mapsto q_1, (q_1, \mathtt{no_tr}) \mapsto q_0\},\\
&q_0 = q_0,\\
&\varepsilon = 0.2,\\
&S_t = \{q_0 \mapsto 0, q_1 \mapsto 5\},\\
&S_e = \{q_0 \mapsto \emptyset, q_1 \mapsto \Sigma\},\\
&\Omega = \{N, T\},\\
&\omega = \{q_0 \mapsto N, q_1 \mapsto T\})
\end{aligned}$$

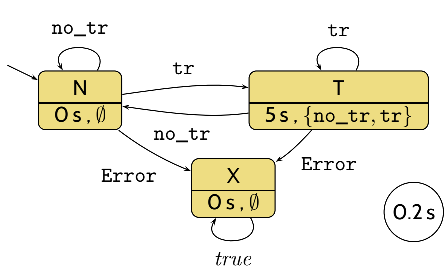


PLC Automata Example: Stuttering Filter with Exception



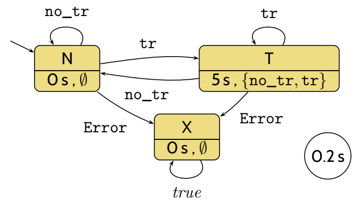
PLC Automata Example: Stuttering Filter with Exception





PLC Automaton Semantics

```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

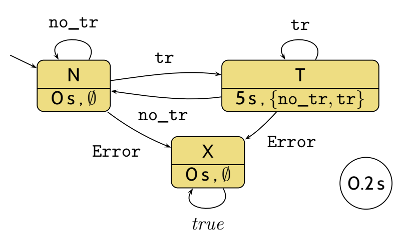


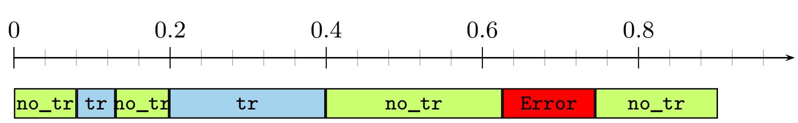
Recall:

- read inputs
- compute
- write outputs

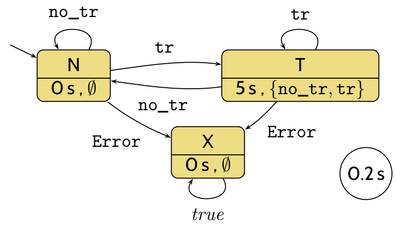
PLCA Semantics: Examples


```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

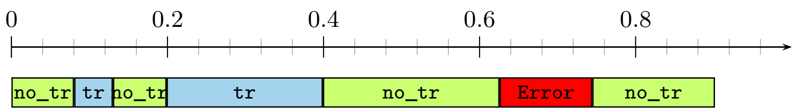



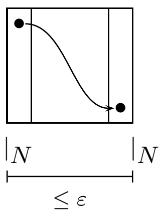


PLCA Semantics: Examples

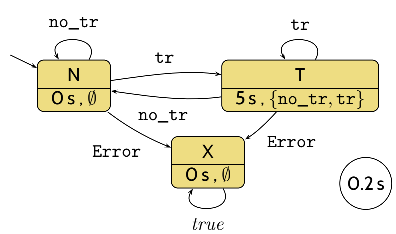



```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

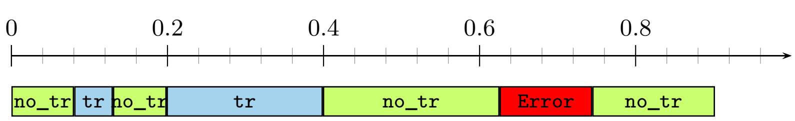


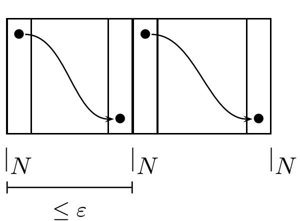


PLCA Semantics: Examples

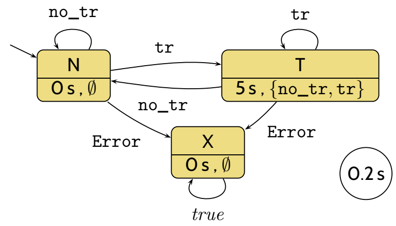



```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

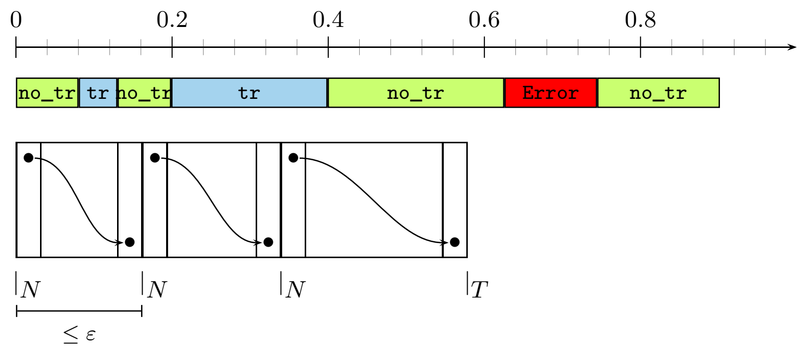





PLCA Semantics: Examples

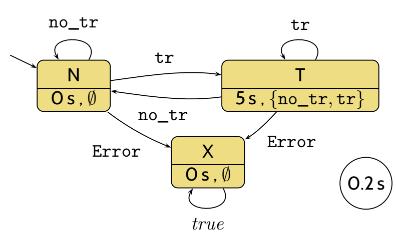



```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

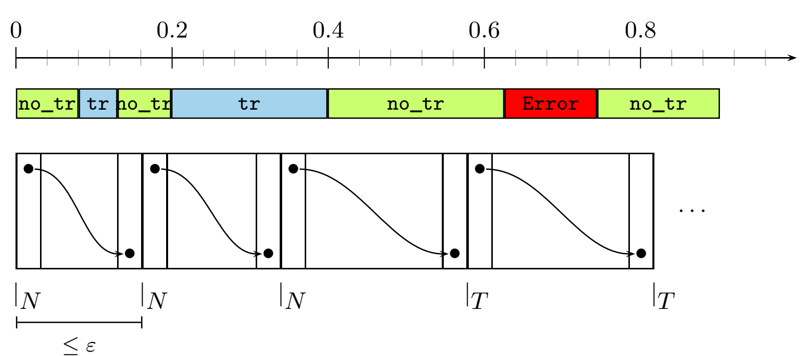


– 10 – 2017-11-30 – Spicdef –

PLCA Semantics: Examples

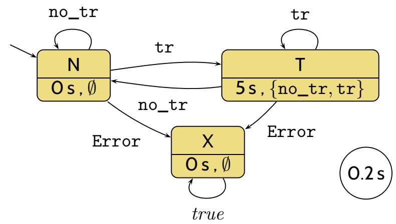



```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

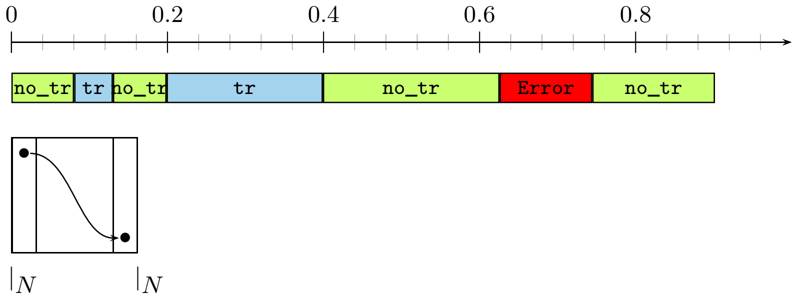



– 10 – 2017-11-30 – Spicdef –

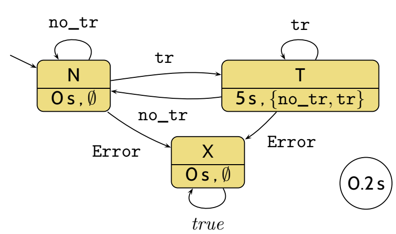
PLCA Semantics: Examples



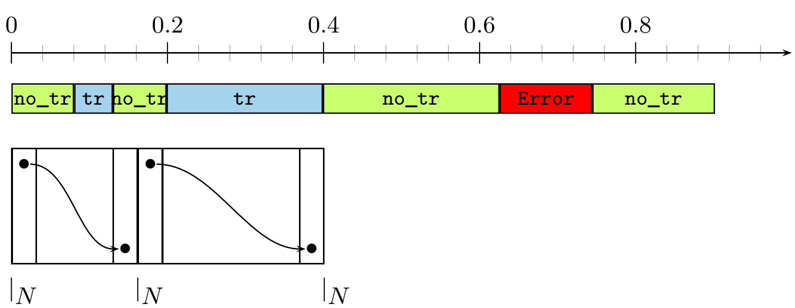

```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```



PLCA Semantics: Examples

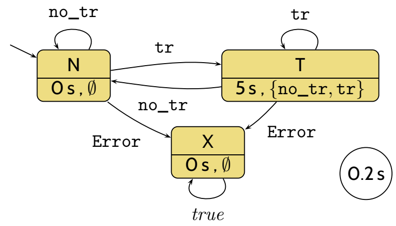


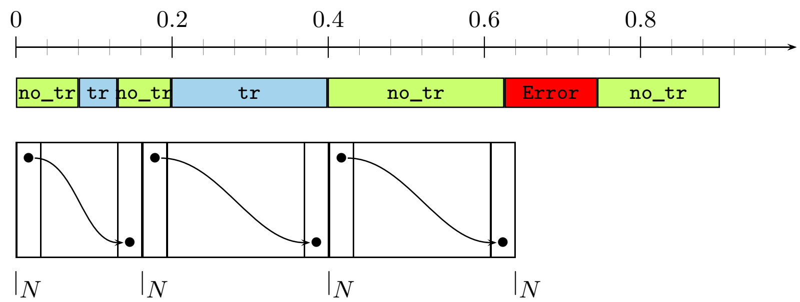

```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

PLCA Semantics: Examples

```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

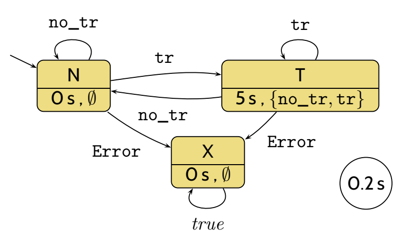


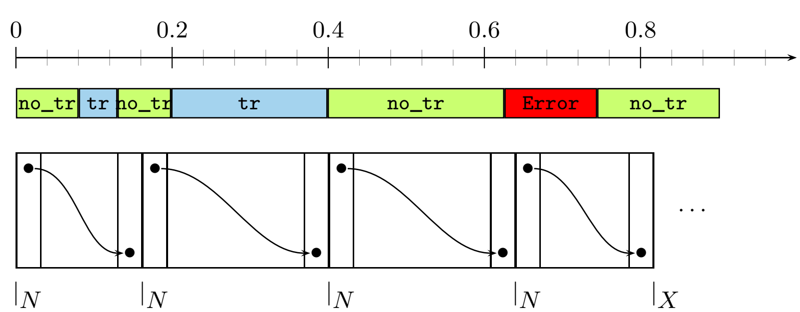


– 10 – 2017-11-30 – Spicdef –

PLCA Semantics: Examples

```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

IF state = 0 THEN
    %output := N;
    IF %input = tr THEN
        state := 1;
        %output := T;
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
    ENDIF
ELSIF state = 1 THEN

    tmr( IN := TRUE, PT := t#5.0s );
    IF (%input = no_tr AND NOT tmr.Q) THEN
        state := 0;
        %output := N;
        tmr( IN := FALSE, PT := t#0.0s );
    ELSIF %input = Error THEN
        state := 2;
        %output := X;
        tmr( IN := FALSE, PT := t#0.0s );
    ENDIF
ENDIF
```

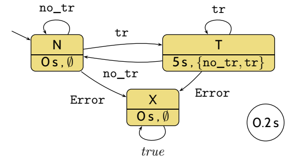





– 10 – 2017-11-30 – Spicdef –

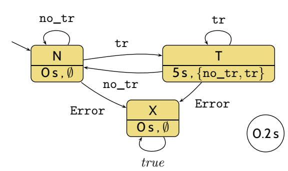
We assess correctness in terms of cycle time ε...

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



We assess correctness in terms of cycle time ε...

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

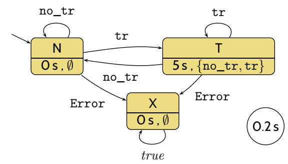


- First of all, ST on the hardware **has** a **cycle time**
  - so we can **measure** it – if it is larger than ε, don't use this program on this PLC hardware;
  - we can **estimate** (approximate) the **worst case execution time** (WCET), if it's larger than ε, don't use it, if it's smaller we're safe.

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

We assess correctness in terms of cycle time ε...

...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 PLC hardware;
  - we can **estimate** (approximate) the **worst case execution time** (WCET), 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**

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).

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.
  - **In other words:** define a DC formula $[\![\mathcal{A}]\!]_{DC}$ 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:** “$[\![\mathcal{A}]\!]$” $\subseteq \{\mathcal{I} \mid \mathcal{I} \models [\![\mathcal{A}]\!]_{DC}\}$.

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.
  - **In other words:** define a DC formula $[\![\mathcal{A}]\!]_{DC}$ 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:** “$[\![\mathcal{A}]\!]$” $\subseteq \{\mathcal{I} \mid \mathcal{I} \models [\![\mathcal{A}]\!]_{DC}\}$.
- **Applications**:
  - Assess **correctness** of over-approximation wrt. DC **requirements**.
    If $\models [\![\mathcal{A}]\!]_{DC} \implies$ Req for a given PLCA $\mathcal{A}$, the $\mathcal{A}$ is **correct**.
  - Prove **generic properties** of PLCA **using DC**, like **reaction time**.

Observables

- Consider the PLCA
  $$\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega).$$
- The DC formula $[\![\mathcal{A}]\!]_{DC}$ we construct ranges over the observables
  - $\mathrm{In}_{\mathcal{A}} : \Sigma$ – values of the **inputs**
  - $\mathrm{St}_{\mathcal{A}} : Q$ – current **local state**
  - $\mathrm{Out}_{\mathcal{A}} : \Omega$ – 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**:

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

$St_A = q_0$

- **Effect of Transitions**:

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

$St_A = q \wedge In_A \in A$ $\quad$ $St_A \in \{\delta(q,a) \mid a \in A\}$

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

$$(\lceil q \wedge A \rceil \wedge \ell = \varepsilon) \longrightarrow \lceil q \vee \delta(q, A) \rceil$$

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**:

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

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

- **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

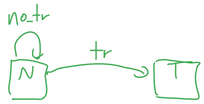
$\mathcal{A} = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$

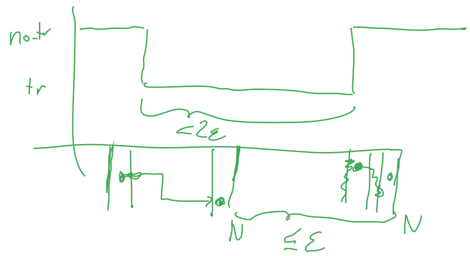
- A arbitrary with $\emptyset \neq A \subseteq \Sigma$,
- $\lceil q \wedge A \rceil$ abbreviates $\lceil \mathsf{St}_\mathcal{A} = q \wedge \mathsf{In}_\mathcal{A} \in A \rceil$,
- $\delta(q, A)$ abbreviates $\mathsf{St}_\mathcal{A} \in \{\delta(q, a) \mid a \in A\}$.

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

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

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





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**:

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

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

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

How to Read these Formulae

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

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

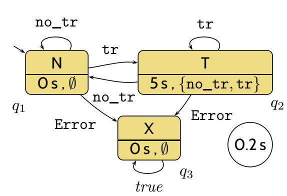
- How to read these formulae?
  - A is a set with $\emptyset \neq A \subseteq \Sigma$,
  - $\lceil q \wedge A \rceil$ abbreviates $\lceil \mathsf{St}_{\mathcal{A}} = q \wedge \mathsf{In}_{\mathcal{A}} \in A \rceil$,
  - $\delta(q, A)$ abbreviates $\mathsf{St}_{\mathcal{A}} \in \{\delta(q, a) \mid a \in A\}$.

How to Read these Formulae

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

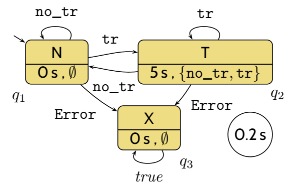
$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

- How to read these formulae?
  - A is a set 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 $\mathsf{St}_{\mathcal{A}} \in \{\delta(q, a) \mid a \in A\}$.



How to Read these Formulae

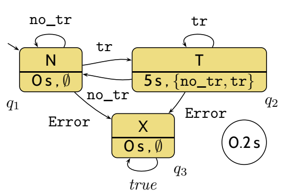
$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$
$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

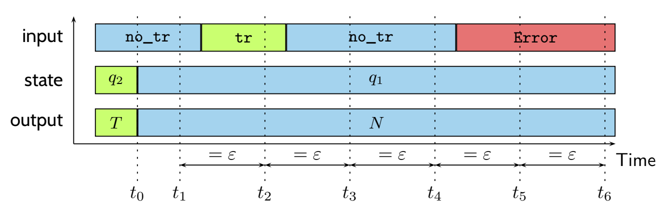


- How to read these formulae?
  - A is a set with $\emptyset \neq A \subseteq \Sigma$,
  - $\lceil q \wedge A \rceil$ abbreviates $\lceil \mathsf{St}_{\mathcal{A}} = q \wedge \mathsf{In}_{\mathcal{A}} \in A \rceil$,
  - $\delta(q, A)$ abbreviates $\mathsf{St}_{\mathcal{A}} \in \{\delta(q, a) \mid a \in A\}$.
- For the stutter filter, (DC-3) abbreviates:

$$\begin{aligned}
&\lceil \neg q_1 \rceil \,;\, \lceil q_1 \wedge \{\mathtt{no_tr}\} \rceil \xrightarrow{\varepsilon} \lceil q_1 \vee q_1 \rceil \\
\wedge\; &\lceil \neg q_1 \rceil \,;\, \lceil q_1 \wedge \{\mathtt{tr}\} \rceil \xrightarrow{\varepsilon} \lceil q_1 \vee q_2 \rceil \\
\wedge\; &\lceil \neg q_1 \rceil \,;\, \lceil q_1 \wedge \{\mathtt{Error}\} \rceil \xrightarrow{\varepsilon} \lceil q_1 \vee q_3 \rceil \\
\wedge\; &\lceil \neg q_1 \rceil \,;\, \lceil q_1 \wedge \{\mathtt{no_tr}, \mathtt{tr}\} \rceil \xrightarrow{\varepsilon} \lceil q_1 \vee q_1 \vee q_2 \rceil \\
\wedge\; &\lceil \neg q_1 \rceil \,;\, \lceil q_1 \wedge \{\mathtt{no_tr}, \mathtt{Error}\} \rceil \xrightarrow{\varepsilon} \lceil q_1 \vee q_1 \vee q_3 \rceil \\
\wedge\; &\lceil \neg q_1 \rceil \,;\, \lceil q_1 \wedge \{\mathtt{tr}, \mathtt{Error}\} \rceil \xrightarrow{\varepsilon} \lceil q_1 \vee q_2 \vee q_3 \rceil \\
\wedge\; &\lceil \neg q_1 \rceil \,;\, \lceil q_1 \wedge \{\mathtt{no_tr}, \mathtt{tr}, \mathtt{Error}\} \rceil \xrightarrow{\varepsilon} \lceil q_1 \vee q_2 \vee q_3 \rceil
\end{aligned}$$

(DC-2): Effect of Transitions

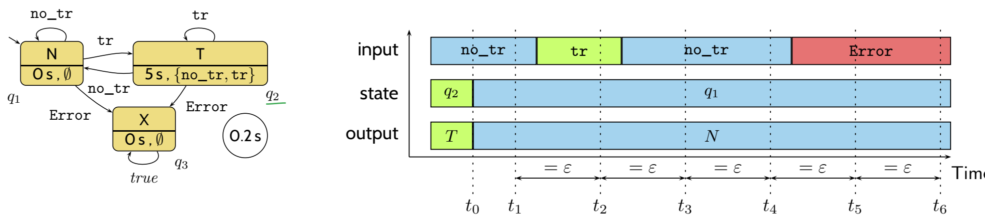




$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

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

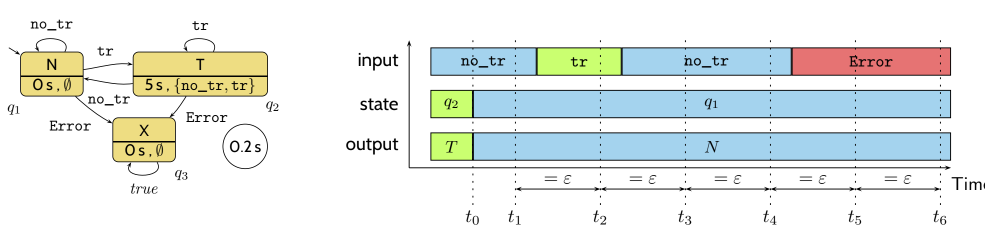
(DC-2): Effect of Transitions



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

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	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	$\{q_1, q_2, q_3\}$	$\{N, T, X\}$

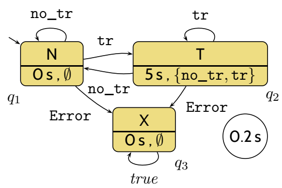
(DC-2): Effect of Transitions

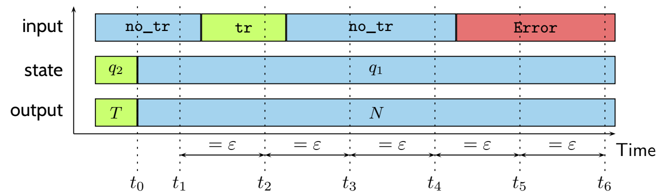


$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	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	$\{q_1, q_2, q_3\}$	$\{N, T, X\}$

(DC-2): Effect of Transitions

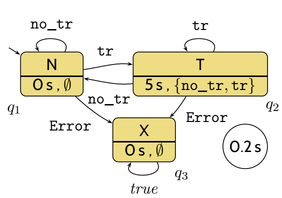


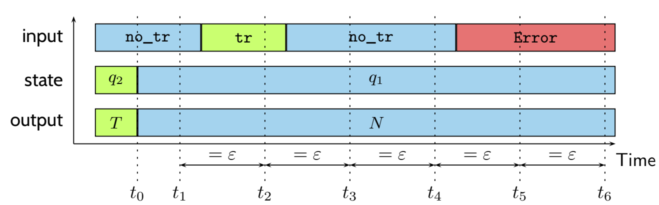


$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	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	$\{q_1, q_2, q_3\}$	$\{N, T, X\}$

(DC-2): Effect of Transitions

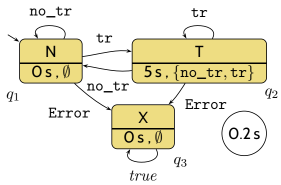


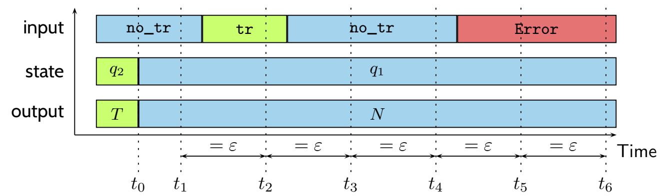


$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	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	$\{q_1, q_2, q_3\}$	$\{N, T, X\}$

(DC-2): Effect of Transitions

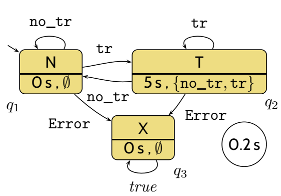


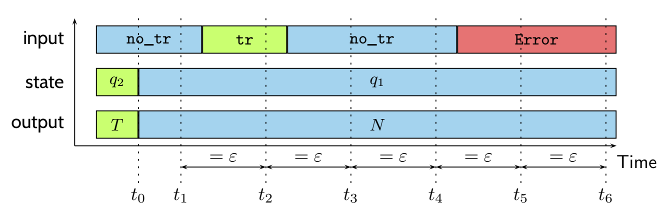


$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	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	$\{q_1, q_2, q_3\}$	$\{N, T, X\}$

(DC-2): Effect of Transitions

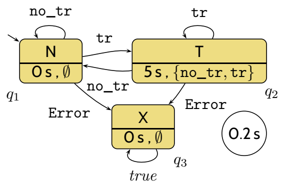




$$\lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \longrightarrow \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-2)}$$

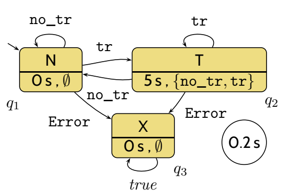
$\lceil q_1 \wedge A \rceil$ holds in	with input	After	state	output
$[t_0, t_1]$	$A = \{\texttt{no\_tr}\}$	t_1	$\{q_1\}$	$\{N\}$
$[t_0, t_2]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_2	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_3]$	$A = \{\texttt{no\_tr}, \texttt{tr}\}$	t_3	$\{q_1, q_2\}$	$\{N, T\}$
$[t_0, t_4]$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

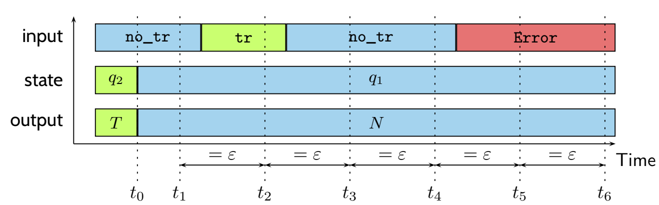
(DC-3): Inputs and Cycle Time



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

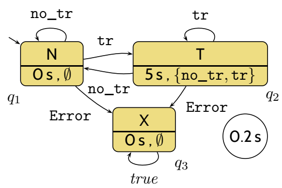
(DC-3): Inputs and Cycle Time

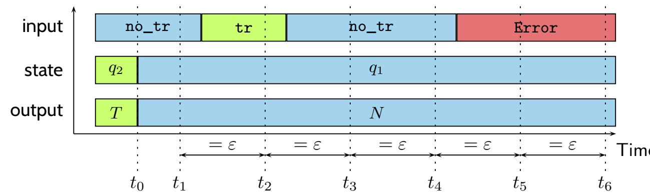




$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

(DC-3): Inputs and Cycle Time

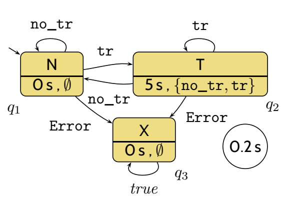


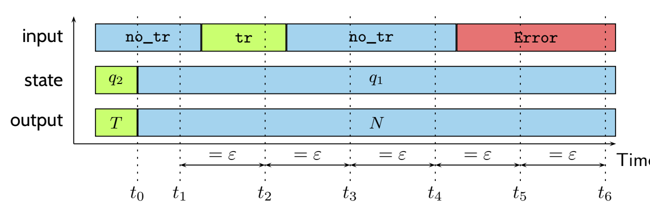


$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_3\}$	$\{N, X\}$

(DC-3): Inputs and Cycle Time

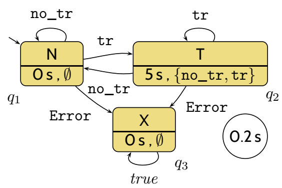


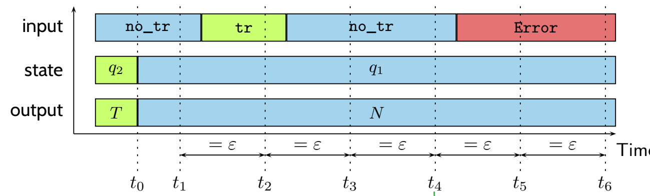


$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_3\}$	$\{N, X\}$

(DC-3): Inputs and Cycle Time

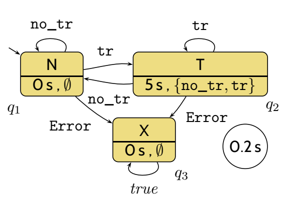


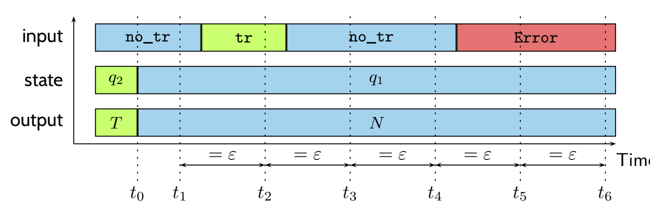


$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_3\}$	$\{N, X\}$

(DC-3): Inputs and Cycle Time

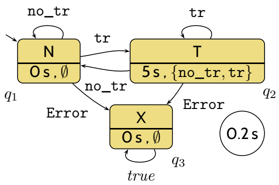


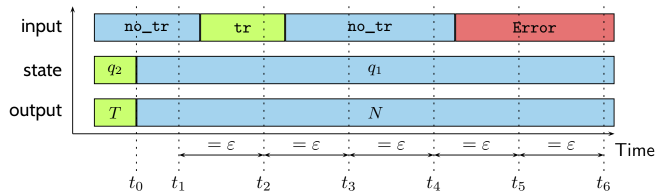


$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_3\}$	$\{N, X\}$

(DC-3): Inputs and Cycle Time



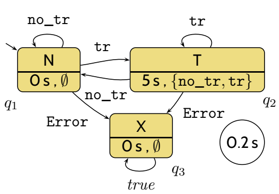


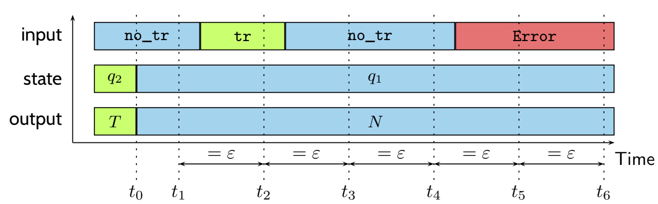
$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_3\}$	$\{N, X\}$

– 10 - 2017-11-30 - Splcdc –

(DC-3): Inputs and Cycle Time



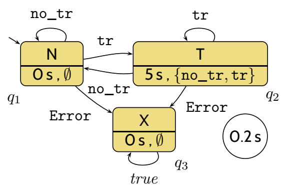


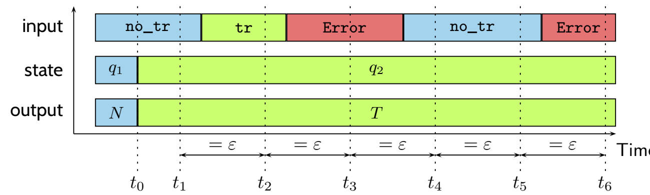
$$\lceil q \wedge A \rceil \xrightarrow{\varepsilon} \lceil q \vee \delta(q, A) \rceil \qquad \text{(DC-3)}$$

$\lceil q_1 \wedge A \rceil$ 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	$\{q_1, q_3\}$	$\{N, X\}$

– 10 - 2017-11-30 - Splcdc –

(DC-4): Delays

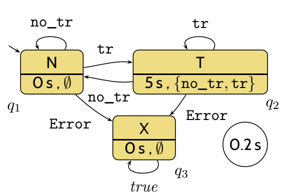


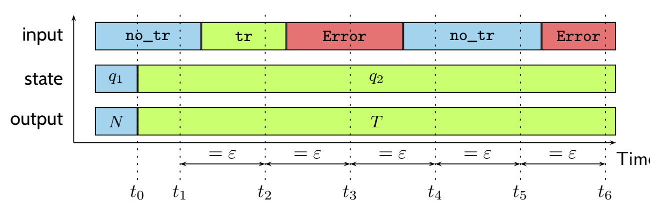


$$S_t(q) > 0 \implies \lceil \neg q \rceil \,;\, \lceil q \wedge A \rceil \xrightarrow{\le S_t(q)} \lceil q \vee \delta(q, A \setminus S_e(q)) \rceil \qquad \text{(DC-4)}$$

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

(DC-5): Delays

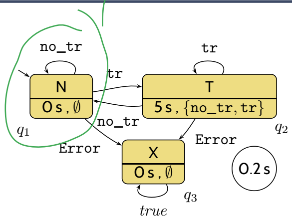


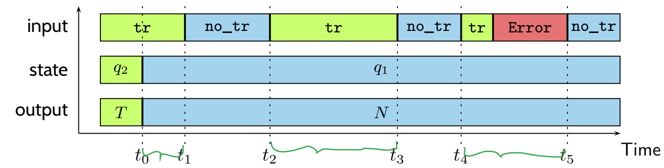


$$S_t(q) > 0 \implies \lceil \neg q \rceil \,;\, \lceil q \rceil \,;\, \lceil q \wedge A \rceil^{\varepsilon} \xrightarrow{\le S_t(q)} \lceil q \vee \delta(q, A \setminus S_e(q)) \rceil \qquad \text{(DC-5)}$$

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

(DC-6) / (DC-7): Progress from non-delayed inputs



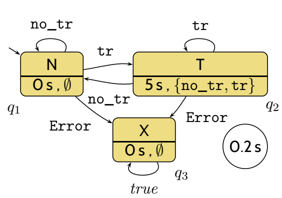


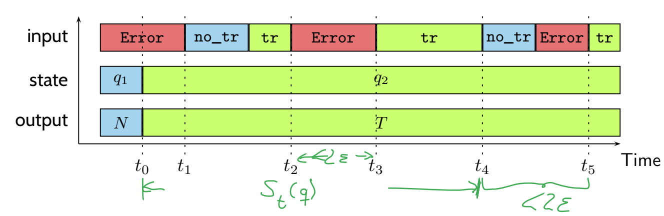
$$S_t(q) = 0 \wedge q \notin \delta(q, A) \implies \Box(\lceil q \wedge A \rceil \implies \ell < 2\varepsilon) \qquad \text{(DC-6)}$$

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

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

(DC-8, DC-9, DC-10): Progress from delayed inputs





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

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

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

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

(DC-11): Behaviour of the Output and System Start

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

(DC-11): Behaviour of the Output and System Start

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

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

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

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

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

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

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 the ST** $P_{\mathcal{A}}$.
- 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.)

Content

- **Programmable Logic Controllers** (PLC) continued
- **PLC Automata**
  - **Example**: Stutter Filter
  - **PLCA Semantics** by example
  - **Cycle time**
- An **over-approximating** **DC Semantics** for **PLC Automata**
  - **observables**, **DC formulae**
- **PLCA Semantics** at work:
  - effect of **transitions** (untimed),
  - **cycle time**, **delays**, **progress**.
- Application example: **Reaction times**
  - **Examples**:
    reaction times of the stutter filter

One Application: Reaction Times

One Application: Reaction Times

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

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

One Application: Reaction Times

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

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

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

- Which is (**why?**) far from obvious from the PLC Automaton in general.

One Application: Reaction Times

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

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

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

- Which is (**why?**) far from obvious from the PLC Automaton in general.

- We will give a theorem,
  which 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**

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

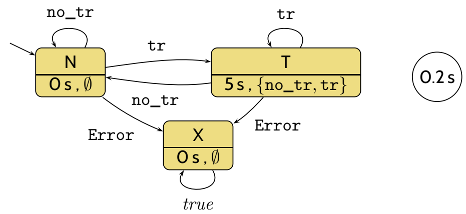
for PLC Automata with

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

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

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

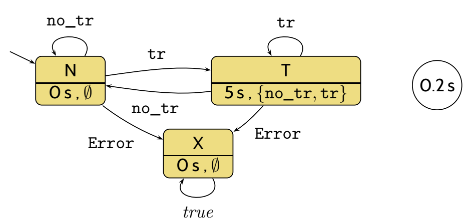
Premise Examples



Examples:

- $\Pi = \{N, T\}, \quad A = \{\texttt{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \subseteq \Pi$

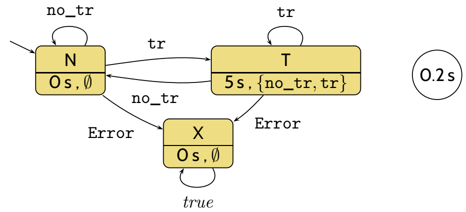
Premise Examples



Examples:

- $\Pi = \{N, T\}, \quad A = \{\texttt{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \subseteq \Pi$

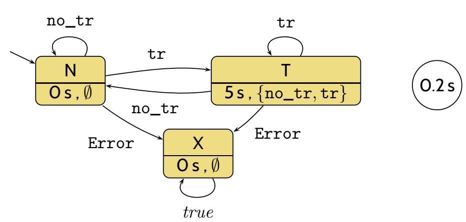
Premise Examples



Examples:

- $\Pi = \{N, T\}, \quad A = \{\texttt{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \subseteq \Pi$
- $\Pi = \{N, T, X\}, \quad A = \{\texttt{Error}\}$
  - $\delta(\Pi, A) = \{X\} \subseteq \Pi$

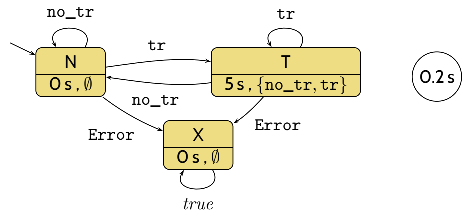
Premise Examples



Examples:

- $\Pi = \{N, T\}, \quad A = \{\texttt{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \subseteq \Pi$
- $\Pi = \{N, T, X\}, \quad A = \{\texttt{Error}\}$
  - $\delta(\Pi, A) = \{X\} \subseteq \Pi$

Premise Examples

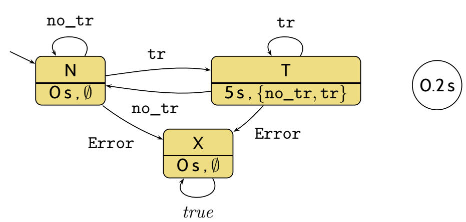


Examples:

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

– 10 - 2017-11-30 - Sreactt –

Premise Examples



Examples:

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

– 10 - 2017-11-30 - Sreactt –

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

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

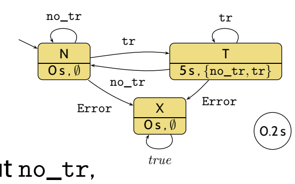
where

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

and

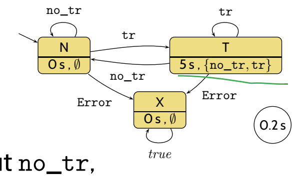
$$s(\pi, A) := \begin{cases} S_t(\pi) + 2\varepsilon & \text{, if } S_t(\pi) > 0 \text{ and } A \cap S_e(\pi) \neq \emptyset \\ \varepsilon & \text{, otherwise.} \end{cases}$$

Reaction Time Theorem: Example 1



(1) If we are in state N or T,
how long does N or T need to **persist together with** input `no_tr`,
to **ensure** that we observe N again?

Reaction Time Theorem: Example 1

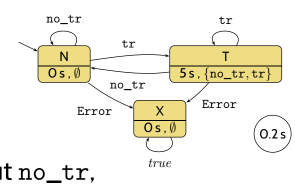


(1) If we are in state N or T,
how long does N or T need to **persist** **together with** input `no_tr`,
to **ensure** that we observe N again?

Your estimation?

- ε
- 2ε
- 3ε
- $5\,s$
- $5\,s + \varepsilon$
- $5\,s + 2\varepsilon$
- $5\,s + 3\varepsilon$
- $\ldots$

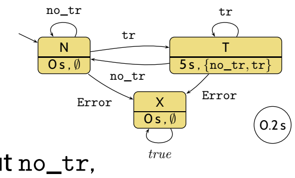
Reaction Time Theorem: Example 1



(1) If we are in state N or T,
how long does N or T need to **persist** **together with** input `no_tr`,
to **ensure** that we observe N again?

$$\lceil\{N, T\} \wedge \{\mathtt{no\_tr}\}\rceil \xrightarrow{5+3\varepsilon} \lceil N \rceil$$

Reaction Time Theorem: Example 1



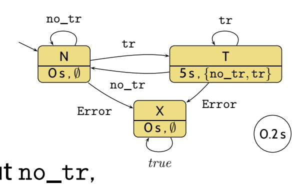
(1) If we are in state N or T,
how long does N or T need to **persist together with** input `no_tr`,
to **ensure** that we observe N again?

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

- **Because**: earlier we have shown

$$\delta(\{N, T\}, \{\texttt{no\_tr}\}) = \{N\}$$

Reaction Time Theorem: Example 1



(1) If we are in state N or T,
how long does N or T need to **persist together with** input `no_tr`,
to **ensure** that we observe N again?

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

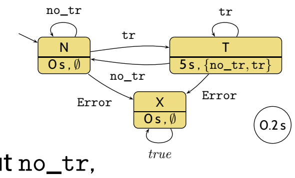
- **Because**: earlier we have shown

$$\delta(\{N, T\}, \{\texttt{no\_tr}\}) = \{N\}$$

- Thus Theorem 5.6 yields

$$\lceil \{N, T\} \wedge \{\texttt{no\_tr}\} \rceil \xrightarrow{c} \lceil N \rceil$$

Reaction Time Theorem: Example 1



(1) If we are in state N or T,
how long does N or T need to **persist together with** input `no_tr`,
to **ensure** that we observe N again?

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

- **Because**: earlier we have shown

$$\delta(\{N, T\}, \{\texttt{no\_tr}\}) = \{N\}$$

- Thus Theorem 5.6 yields

$$\lceil \{N, T\} \wedge \{\texttt{no\_tr}\} \rceil \xrightarrow{c} \lceil N \rceil$$

with

$$\begin{aligned} c &= \varepsilon + \max(\{0\} \cup \{s(\pi, \{\texttt{no\_tr}\}) \mid \pi \in \{N, T\} \setminus \{N\}\}) \\ &= \varepsilon + \max(\{0\} \cup \{s(T, \{\texttt{no\_tr}\})\}) \\ &= \varepsilon + 5 + 2\varepsilon = 5 + 3\varepsilon \end{aligned}$$

Reaction Time Theorem: Example 2

(2) If we are in state N, T, or X,
how long does input `Error` need to **persist**
to **ensure** that we observe X again?

Reaction Time Theorem: Example 2

(2) If we are in state N, T, or X,
how long does input `Error` need to **persist**
to **ensure** that we observe X again?

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

Reaction Time Theorem: Example 2

(2) If we are in state N, T, or X,
how long does input `Error` need to **persist**
to **ensure** that we observe X again?

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

- **Because**: earlier we have shown

$$\delta(\{N, T, X\}, \{\texttt{Error}\}) = \{X\}$$

Reaction Time Theorem: Example 2

(2) If we are in state N, T, or X,
how long does input $\mathtt{Error}$ need to **persist**
to **ensure** that we observe X again?

$$\lceil \{N, T, X\} \wedge \{\mathtt{Error}\} \rceil \xrightarrow{2\varepsilon} \lceil X \rceil$$

- **Because**: earlier we have shown

$$\delta(\{N, T, X\}, \{\mathtt{Error}\}) = \{X\}$$

- Thus Theorem 5.6 yields

$$\lceil \{N, T, X\} \wedge \{\mathtt{Error}\} \rceil \xrightarrow{c} \lceil X \rceil$$

Reaction Time Theorem: Example 2

(2) If we are in state N, T, or X,
how long does input $\mathtt{Error}$ need to **persist**
to **ensure** that we observe X again?

$$\lceil \{N, T, X\} \wedge \{\mathtt{Error}\} \rceil \xrightarrow{2\varepsilon} \lceil X \rceil$$

- **Because**: earlier we have shown

$$\delta(\{N, T, X\}, \{\mathtt{Error}\}) = \{X\}$$

- Thus Theorem 5.6 yields

$$\lceil \{N, T, X\} \wedge \{\mathtt{Error}\} \rceil \xrightarrow{c} \lceil X \rceil$$

with

$$\begin{aligned} c &= \varepsilon + \max(\{0\} \cup \{s(\pi, \{\mathtt{Error}\}) \mid \pi \in \{N, T, X\} \setminus \{X\}\}) \\ &= \varepsilon + \max(\{0\} \cup \{s(N, \{\mathtt{Error}\}), s(T, \{\mathtt{Error}\})\}) \\ &= \varepsilon + \varepsilon = 2\varepsilon \end{aligned}$$

Reaction Time Theorem: Example 3

(2) If we are in state N or T,
how long do inputs `no_tr` or `tr` need to **persist**
to **ensure** that we observe N or T again?

Reaction Time Theorem: Example 3

(2) If we are in state N or T,
how long do inputs `no_tr` or `tr` need to **persist**
to **ensure** that we observe N or T again?

$$\lceil \{N, T\} \wedge \{\texttt{no\_tr}, \texttt{tr}\} \rceil \xrightarrow{\varepsilon} \lceil N, T \rceil$$

Reaction Time Theorem: Example 3

(2) If we are in state N or T,
how long do inputs $\texttt{no\_tr}$ or $\texttt{tr}$ need to **persist**
to **ensure** that we observe N or T again?

$$\lceil \{N, T\} \wedge \{\texttt{no\_tr}, \texttt{tr}\} \rceil \xrightarrow{\varepsilon} \lceil N, T \rceil$$

- **Because**: earlier we have shown

$$\delta(\{N, T\}, \{\texttt{no\_tr}, \texttt{tr}\}) = \{N, T\}$$

– 10 - 2017-11-30 - Sreact –

Reaction Time Theorem: Example 3

(2) If we are in state N or T,
how long do inputs $\texttt{no\_tr}$ or $\texttt{tr}$ need to **persist**
to **ensure** that we observe N or T again?

$$\lceil \{N, T\} \wedge \{\texttt{no\_tr}, \texttt{tr}\} \rceil \xrightarrow{\varepsilon} \lceil N, T \rceil$$

- **Because**: earlier we have shown

$$\delta(\{N, T\}, \{\texttt{no\_tr}, \texttt{tr}\}) = \{N, T\}$$

- Thus Theorem 5.6 yields

$$\lceil \{N, T\} \wedge \{\texttt{no\_tr}, \texttt{tr}\} \rceil \xrightarrow{c} \lceil N, T \rceil$$

– 10 - 2017-11-30 - Sreact –

Reaction Time Theorem: Example 3

(2) If we are in state N or T,
how long do inputs `no_tr` or `tr` need to **persist**
to **ensure** that we observe N or T again?

$$\lceil \{N, T\} \wedge \{\mathtt{no\_tr}, \mathtt{tr}\} \rceil \xrightarrow{\varepsilon} \lceil N, T \rceil$$

- **Because**: earlier we have shown

$$\delta(\{N, T\}, \{\mathtt{no\_tr}, \mathtt{tr}\}) = \{N, T\}$$

- Thus Theorem 5.6 yields

$$\lceil \{N, T\} \wedge \{\mathtt{no\_tr}, \mathtt{tr}\} \rceil \xrightarrow{c} \lceil N, T \rceil$$

with

$$\begin{aligned} c &= \varepsilon + \max(\{0\} \cup \{s(\pi, \{\mathtt{no\_tr}, \mathtt{tr}\}) \mid \pi \in \{N, T\} \setminus \{N, T\}\}) \\ &= \varepsilon + \max(\{0\} \cup \emptyset) \\ &= \varepsilon \end{aligned}$$

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

$$\delta^{n+1}(\Pi, A) \subseteq \delta^n(\Pi, A) \subseteq \dots \subseteq \underbrace{\delta(\delta(\Pi, A), A)}_{=\delta^2(\Pi, A)} \subseteq \delta(\Pi, A) \subseteq \Pi$$

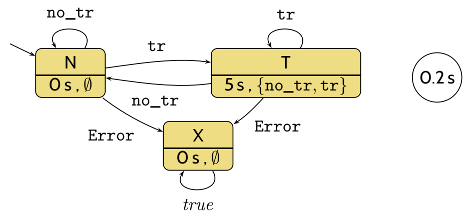
  i.e. the sequence is a **contraction**.

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

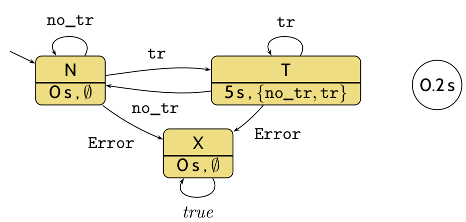
Contraction Examples



Examples:

- $\Pi = \{N, T\}, A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$

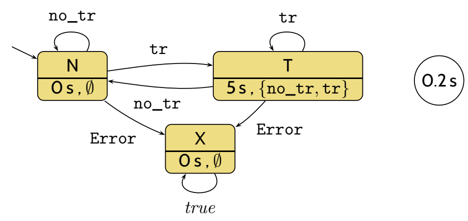
Contraction Examples



Examples:

- $\Pi = \{N, T\}, A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$

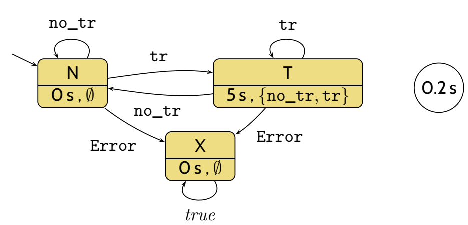
Contraction Examples



Examples:

- $\Pi = \{N, T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$

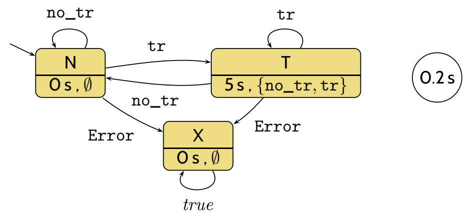
Contraction Examples



Examples:

- $\Pi = \{N, T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$

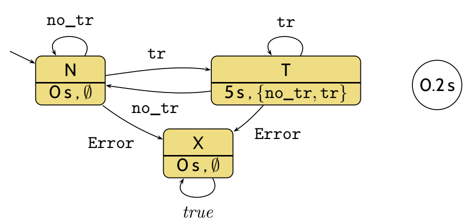
Contraction Examples



Examples:

- $\Pi = \{N, T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{N\}$

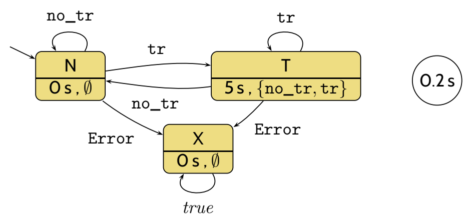
Contraction Examples



Examples:

- $\Pi = \{N, T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{N\}$

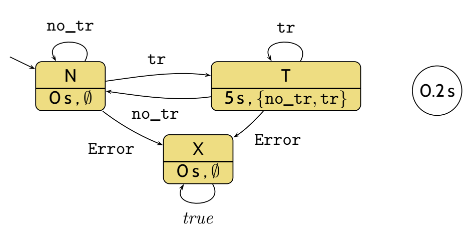
Contraction Examples



Examples:

- $\Pi = \{N, T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{N\}$
- $\Pi = \{N, T, X\}$, $A = \{\texttt{Error}\}$
  - $\delta^0(\Pi, A) = \{N, T, X\}$
  - $\delta(\delta^0(\Pi, A), A) = \{X\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{X\}$

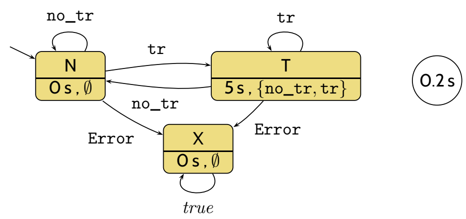
Contraction Examples



Examples:

- $\Pi = \{N, T\}$, $A = \{\texttt{no\_tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{N\}$
- $\Pi = \{N, T, X\}$, $A = \{\texttt{Error}\}$
  - $\delta^0(\Pi, A) = \{N, T, X\}$
  - $\delta(\delta^0(\Pi, A), A) = \{X\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{X\}$

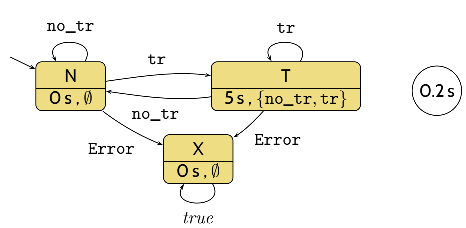
Contraction Examples



Examples:

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

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

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

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

Proof Idea of Reaction Time Theorem

(by contradiction)

- Assume, we would **not** have

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

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

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

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

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

- Which is equivalent to having

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

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

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

- Which is equivalent to having

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

- 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.

Content

- **Programmable Logic Controllers** (PLC) continued
- **PLC Automata**
  - **Example**: Stutter Filter
  - **PLCA Semantics** by example
  - **Cycle time**
- An **over-approximating** **DC Semantics** for **PLC Automata**
  - **observables**, **DC formulae**
- **PLCA Semantics** at work:
  - effect of **transitions** (untimed),
  - **cycle time**, **delays**, **progress**.
- Application example: **Reaction times**
  - **Examples**: reaction times of the stutter filter

Tell Them What You've Told Them...

- **Programmable Logic Controllers** (PLC) are epitomic for real-time controller platforms:
  - have **real-time clock** device, **read inputs** / **write outputs**, manage **local state**.
- The set of evolutions of a **PLC Automaton** can be over-approximated by a set of **DC formulae**.
- This **DC-Semantics** of PLCA can be used to establish **generic properties** of PLCA like **reaction time**.
- The **reaction time theorems** give us "recipes" to analyse PLCA for reaction time (just considering the PLCA, not its DC semantics).
- And that's **Duration Calculus** for now...
  - Next block: **Timed Automata**
  - Later: verifying that a **Network of Timed Automata** **satisfies** a requirement formalised using DC.

    Thus connecting both "worlds".

Content

Introduction

- **Observables and Evolutions**
- **Duration Calculus** (DC)
- Semantical Correctness Proofs
- DC Decidability
- DC Implementables
- **PLC-Automata**

$obs : \mathsf{Time} \to \mathscr{D}(obs)$

- **Timed Automata** (TA), Uppaal
- Networks of Timed Automata
- Region/Zone-Abstraction
- TA model-checking
- Extended Timed Automata
- Undecidability Results

$\langle obs_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle obs_1, \nu_1 \rangle, t_1 \ldots$

- **Automatic Verification...**

  ...whether a TA satisfies a DC formula, observer-based
- **Recent Results**:
  - **Timed Sequence Diagrams**, or **Quasi-equal Clocks**, or **Automatic Code Generation**, or ...

References

Olderog, E.-R. and Dierks, H. (2008). *Real-Time Systems - Formal Specification and Automatic Verification*. Cambridge University Press.