Real-Time Systems

Lecture 10: PLC Automata

2017-11-30

Dr. Bernd Westphal
Dr. Jochen Hoenicke

Albert-Ludwigs-Universität Freiburg, Germany
The Plan

<table>
<thead>
<tr>
<th>Full DC</th>
<th>DC Implementables</th>
<th>PLC-Automata</th>
<th>IEC 61131-3</th>
<th>Binary</th>
</tr>
</thead>
</table>

- 'Req'
- 'Des'
- 'Impl'

synthesis / code generation (in the book)

by example

(convect?) compiler

prove properties of generated PLCA using DC

prove

prove

prove

\[ [A]_{DC} \]

later

\[ q_0 \rightarrow \text{no}_{-tr} \rightarrow \text{tr} \rightarrow q_1 \]

\[ 0.2s \]
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.
• 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?

```plaintext
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
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
ENDF
```

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

LD \( x \)  
OR \( y \)  
ST \( z \)  

\[ z := x \ OR \ y \]

Instruction List  
Structured Text

(Relay) Ladder Diagram  
Function Block Diagram

Figure 2.2: Implementations of the operation “\( x \) becomes \( y \ OR \ z \)”

Tied together by

- Sequential Function Charts (SFC)

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

Figure 2.3: Elements of sequential function charts
• 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
Definition 5.2. A **PLC-Automaton** is a structure

\[ A = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega) \]

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**,
- \(\delta : Q \times \Sigma \rightarrow Q\) is the **transition function** (!),
- \(S_t : Q \rightarrow \mathbb{R}_0^+\) assigns a **delay time** to each state,
- \(S_e : Q \rightarrow 2^\Sigma\) assigns a set of **delayed inputs** to each state,
- \(\Omega\) is a finite, non-empty set of **outputs**,
- \(\omega : Q \rightarrow \Omega\) assigns an **output** to each state,
- \(\varepsilon\) 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

\[ A = (Q = \{q_0, q_1\}, \]
\[ \Sigma = \{\text{tr, no\_tr}\}, \]
\[ \delta = \{(q_0, \text{tr}) \mapsto q_1, (q_0, \text{no\_tr}) \mapsto q_0, (q_1, \text{tr}) \mapsto q_1, (q_1, \text{no\_tr}) \mapsto q_0\}, \]
\[ q_0 = q_0, \]
\[ \varepsilon = 0.2, \]
\[ S_t = \{q_0 \mapsto 0, q_1 \mapsto 5\}, \]
\[ S_e = \{q_0 \mapsto \emptyset, q_1 \mapsto \Sigma\}, \]
\[ \Omega = \{N, T\}, \]
\[ \omega = \{q_0 \mapsto N, q_1 \mapsto T\} \)
PLC Automata Example: Stuttering Filter

\[ A = (Q = \{q_0, q_1\}, \]
\[ \Sigma = \{\text{tr, no\_tr}\}, \]
\[ \delta = \{(q_0, \text{tr}) \mapsto q_1, (q_0, \text{no\_tr}) \mapsto q_0, (q_1, \text{tr}) \mapsto q_1, (q_1, \text{no\_tr}) \mapsto q_0\}, \]
\[ q_0 = q_0, \]
\[ \varepsilon = 0.2, \]
\[ S_t = \{q_0 \mapsto 0, q_1 \mapsto 5\}, \]
\[ S_e = \{q_0 \mapsto \emptyset, q_1 \mapsto \Sigma\}, \]
\[ \Omega = \{N, T\}, \]
\[ \omega = \{q_0 \mapsto N, q_1 \mapsto T\} \]
PLC Automata Example: Stuttering Filter with Exception
PLC Automata Example: Stuttering Filter with Exception

[Diagram of PLC automata with transitions and states labeled as follows:
- States: N (0s), T (5s), X (0s), X (Os, ∅)
- Transitions: no_tr, tr
- Diagram includes error states and true states.
- Time delays: 0.2s]
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
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;
  ENDIF
  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
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#5.0 s );
  IF (%input = no_tr AND NOT tmr.Q) THEN
    state := 0;
    %output := N;
    tmr( IN := FALSE, PT := t#0.0 s );
  ELSIF %input = Error THEN
    state := 2;
    %output := X;
    tmr( IN := FALSE, PT := t#0.0 s );
  ENDIF
ENDIF
ENDIFF
```

```
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#5.0 s );
  IF (%input = no_tr AND NOT tmr.Q) THEN
    state := 0;
    %output := N;
    tmr( IN := FALSE, PT := t#0.0 s );
  ELSIF %input = Error THEN
    state := 2;
    %output := X;
    tmr( IN := FALSE, PT := t#0.0 s );
  ENDIF
ENDIF
ENDIFF
```
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;
  ELSEIF %input = Error THEN
    state := 2;
    %output := X;
  ENDIF
ELSIF state = 1 THEN
  tmr( IN := TRUE, PT := t#5.0 s );
  IF (%input = no_tr AND NOT tmr.Q) THEN
    state := 0;
    %output := N;
    tmr( IN := FALSE, PT := t#0.0 s );
  ELSEIF %input = Error THEN
    state := 2;
    %output := X;
    tmr( IN := FALSE, PT := t#0.0 s );
  ENDIF
ENDIF
END

0 0.2 0.4 0.6 0.8

no_tr tr no_tr tr no_tr Error no_tr

|N|N|N|N
\leq \varepsilon
PLCA Semantics: Examples

```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0 := N, 1 := T, 2 := X *)
  timer : 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
  timer (IN := TRUE, PT := t#5.0s);
  IF (input = no_tr AND NOT timer.Q) THEN
    state := 0;
    output := N;
    timer (IN := FALSE, PT := t#0.0s);
  ELSIF input = Error THEN
    state := 2;
    output := X;
    timer (IN := FALSE, PT := t#0.0s);
  ENDIF
ENDIF
```

Diagram showing the PLCA states and transitions:

- N (No Transition)
- T (Transition)
- X (Error)
- Empty States (not transition)

Timers:
- Timer 1: 0.2s
- Timer 2: 5s
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;
  ELSEIF %input = Error THEN
    state := 2;
    %output := X;
  ENDIF
ELSIF state = 1 THEN
  tmr(IN := TRUE, PT := t#5.0 s);
  IF (%input = no_tr AND NOT tmr.Q) THEN
    state := 0;
    %output := N;
  ELSEIF %input = Error THEN
    state := 2;
    %output := X;
  tmr(IN := FALSE, PT := t#0.0 s);
ENDIF
ELSIF state = 2 THEN
  tmr(IN := FALSE, PT := t#0.0 s);
ENDIF
ENDF

<table>
<thead>
<tr>
<th>N</th>
<th>N</th>
<th>N</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
</table>

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

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

Diagram:

- **States:**
  - N: no_tr
  - T: tr
  - X: Error

- **Transitions:**
  - N to T: 0.2 s
  - T to N: 5 s, \{no_tr, tr\}
  - N to X: 0.2 s

- **Timers:**
  - tmr

- **Program Flow:**
  0.2 s
  0.4 s
  0.6 s
  0.8 s

- **Input/Output States:**
  - no_tr
  - tr
  - no_tr
  - Error
  - no_tr
We assess correctness in terms of cycle time $\varepsilon$...

...but where does the cycle time come from?
We assess correctness in terms of cycle time $\varepsilon$...

...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 $\varepsilon$, don’t use this program on this PLC hardware;
  - we can estimate (approximate) the worst case execution time (WCET), if it’s larger than $\varepsilon$, 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 $\varepsilon$...

...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 $\varepsilon$, don’t use this program on this PLC hardware;
  - we can estimate (approximate) the worst case execution time (WCET), if it’s larger than $\varepsilon$, 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 $\varepsilon$,
  - 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 $\llbracket A \rrbracket_{DC}$ such that
    
    \[
    \{ I \in \llbracket A \rrbracket \} \implies I \models \llbracket A \rrbracket_{DC}
    \]
    
    but not necessarily the other way round.
  - **In even other words:** $\llbracket A \rrbracket \subseteq \{ I \mid I \models \llbracket A \rrbracket_{DC} \}$. 

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 $\llbracket A \rrbracket_{DC}$ such that
    \[
    I \in \llbracket A \rrbracket \Rightarrow I \models \llbracket A \rrbracket_{DC}
    \]
    but not necessarily the other way round.
  - **In even other words:** “$\llbracket A \rrbracket$” $\subseteq \{ I \mid I \models \llbracket A \rrbracket_{DC} \}$.
- **Applications:**
  - Assess **correctness** of over-approximation wrt. DC requirements.
    If $\models \llbracket A \rrbracket_{DC} \implies$ Req for a given PLCA $A$, the $A$ 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
  
  - \(\text{In}_\mathcal{A} : \Sigma\) – values of the inputs
  - \(\text{St}_\mathcal{A} : Q\) – current local state
  - \(\text{Out}_\mathcal{A} : \Omega\) – values of the outputs
Overview

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

- \( A \) arbitrary with \( \emptyset \neq A \subseteq \Sigma \).
- \([q \land A]\) abbreviates \([\text{St}_A = q \land \ln_A \in A]\).
- \( \delta(q, A) \) abbreviates \( \text{St}_A \in \{\delta(q, a) \mid a \in A\} \).

- **Initial State:**
  \[
  \left( \left[ \neg q \right] ; \left[ q \land A \right] \right) \quad \text{true} \]
  \( \text{St}_A = q_0 \)  
  \( \text{(DC-1)} \)

- **Effect of Transitions:**
  \[
  \left[ q \land A \right] \xrightarrow{\varepsilon} \left[ q \land \delta(q, A) \right] 
  \text{St}_A = q \land \ln_A \in A 
  \text{St}_A \in \delta(q, a) \mid a \in A
  \]
  \( \text{(DC-2)} \)
  \[
  \left[ q \land A \right] \rightarrow \left[ q \lor \delta(q, A) \right] 
  \text{(DC-3)}
  \]
Overview

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

- **Initial State:**
  \[ [\top] \lor [q_0] ; \text{true} \quad \text{(DC-1)} \]

- **Effect of Transitions:**
  \[ [\neg q] ; [q \land A] \rightarrow [q \lor \delta(q, A)] \quad \text{(DC-2)} \]

does not affect \[ q \land A \], \[ \varepsilon \rightarrow [q \lor \delta(q, A)] \]

- **Delays:**
  \[ S_t(q) > 0 \implies [\neg q] ; [q \land A] \xrightarrow{\leq S_t(q)} [q \lor \delta(q, A \setminus S_e(q))] \quad \text{(DC-4)} \]
  \[ S_t(q) > 0 \implies [\neg q] ; [q] ; [q \land A] \xrightarrow{\leq S_t(q)} [q \lor \delta(q, A \setminus S_e(q))] \quad \text{(DC-5)} \]
Overview

A = (Q, Σ, δ, q₀, ε, St, Sₑ, Ω, ω)

- A arbitrary with ∅ ≠ A ⊆ Σ,
- [q ∧ A] abbreviates
  [Stₐ = q ∧ lnₐ ∈ A],
- δ(q, A) abbreviates
  Stₐ ∈ {δ(q, a) | a ∈ A}.

- Progress from non-delayed inputs:

  \[ S_t(q) = 0 \land q \notin δ(q, A) \implies □([q \land A] \implies ℓ < 2ε) \]  \hspace{1cm} (DC-6)

  \[ S_t(q) = 0 \land q \notin δ(q, A) \implies [\neg q] ; [q \land A]^{ε} \rightarrow [\neg q] \]  \hspace{1cm} (DC-7)
Overview

\[ A = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega) \]

- A arbitrary with \( \emptyset \neq A \subseteq \Sigma \),
- \([q \land A]\) abbreviates \([\text{St}_A = q \land \text{In}_A \in A]\),
- \(\delta(q, A)\) abbreviates \(\text{St}_A \in \{\delta(q, a) | a \in A\}\).

• Progress from non-delayed inputs:

\[
S_t(q) = 0 \land q \notin \delta(q, A) \implies \square([q \land A] \implies \ell < 2\varepsilon) \quad \text{(DC-6)}
\]

\[
S_t(q) = 0 \land q \notin \delta(q, A) \implies [\neg q] ; [q \land A]^\varepsilon \rightarrow [\neg q] \quad \text{(DC-7)}
\]

• Progress from delayed inputs:

\[
S_t(q) > 0 \land q \notin \delta(q, A)
\]

\[
\implies \square([q]^{S_t(q)} ; [q \land A] \implies \ell < S_t(q) + 2\varepsilon) \quad \text{(DC-8)}
\]

\[
S_t(q) > 0 \land A \cap S_e(q) = \emptyset \land q \notin \delta(q, A)
\]

\[
\implies \square([q \land A] \implies \ell < 2\varepsilon) \quad \text{(DC-9)}
\]

\[
S_t(q) > 0 \land A \cap S_e(q) = \emptyset \land q \notin \delta(q, A)
\]

\[
\implies [\neg q] ; [q \land A]^\varepsilon \rightarrow [\neg q] \quad \text{(DC-10)}
\]
How to Read these Formulae

- How to read these formulae?
  - $A$ is a set with $\emptyset \neq A \subseteq \Sigma$,
  - $[q \land A]$ abbreviates $[\text{St}_A = q \land \text{In}_A \in A]$,
  - $\delta(q, A)$ abbreviates $\text{St}_A \in \{\delta(q, a) \mid a \in A\}$.

\[
\begin{align*}
\neg q ; [q \land A] & \rightarrow [q \lor \delta(q, A)] \\
[q \land A] & \xrightarrow{\varepsilon} [q \lor \delta(q, A)]
\end{align*}
\] (DC-2) (DC-3)
How to Read these Formulae

\[ \neg q ; [q \land A] \rightarrow [q \lor \delta(q, A)] \]
\[ [q \land A] \varepsilon \rightarrow [q \lor \delta(q, A)] \]  

How to read these formulae?

- \( A \) is a set with \( \emptyset \neq A \subseteq \Sigma \),
- \( [q \land A] \) abbreviates \( \text{St}_A = q \land \text{ln}_A \in A \),
- \( \delta(q, A) \) abbreviates \( \text{St}_A \in \{ \delta(q, a) \mid a \in A \} \).
How to Read these Formulae

\[ [\neg q] ; [q \land A] \rightarrow [q \lor \delta(q, A)] \quad \text{(DC-2)} \]
\[ [q \land A] \xrightarrow{\varepsilon} [q \lor \delta(q, A)] \quad \text{(DC-3)} \]

- How to read these formulae?
  - \( A \) is a set with \( \emptyset \neq A \subseteq \Sigma \),
  - \([q \land A]\) abbreviates \([\text{St}_A = q \land \text{ln}_A \in A]\),
  - \(\delta(q, A)\) abbreviates \(\text{St}_A \in \{\delta(q, a) \mid a \in A\}\).

- For the stutter filter, (DC-3) abbreviates:

\[ [\neg q_1] ; [q_1 \land \{\text{no\_tr}\}] \xrightarrow{\varepsilon} [q_1 \lor q_1] \]
\[ \land [\neg q_1] ; [q_1 \land \{\text{tr}\}] \xrightarrow{\varepsilon} [q_1 \lor q_2] \]
\[ \land [\neg q_1] ; [q_1 \land \{\text{Error}\}] \xrightarrow{\varepsilon} [q_1 \lor q_3] \]
\[ \land [\neg q_1] ; [q_1 \land \{\text{no\_tr}, \text{tr}\}] \xrightarrow{\varepsilon} [q_1 \lor q_1 \lor q_2] \]
\[ \land [\neg q_1] ; [q_1 \land \{\text{no\_tr}, \text{Error}\}] \xrightarrow{\varepsilon} [q_1 \lor q_1 \lor q_3] \]
\[ \land [\neg q_1] ; [q_1 \land \{\text{tr}, \text{Error}\}] \xrightarrow{\varepsilon} [q_1 \lor q_2 \lor q_3] \]
\[ \land [\neg q_1] ; [q_1 \land \{\text{no\_tr}, \text{tr}, \text{Error}\}] \xrightarrow{\varepsilon} [q_1 \lor q_2 \lor q_3] \]
(DC-2): Effect of Transitions

\[
[q_1 \land A] \quad \text{holds in} \quad \left[ t_0, t_1 \right] \\
A = \{\text{no\_tr}\} \\
\text{After} \quad t_1 \\
\text{state} \quad \{q_1, q_2\} \\
\text{output} \quad \{N\}
\]

\[
[q_1 \land A] \quad \text{holds in} \quad \left[ t_0, t_2 \right] \\
A = \{\text{no\_tr, tr}\} \\
\text{After} \quad t_2 \\
\text{state} \quad \{q_1, q_2\} \\
\text{output} \quad \{N, T\}
\]

\[
[q_1 \land A] \quad \text{holds in} \quad \left[ t_0, t_3 \right] \\
A = \{\text{no\_tr, tr}\} \\
\text{After} \quad t_3 \\
\text{state} \quad \{q_1, q_2\} \\
\text{output} \quad \{N, T\}
\]

\[
[q_1 \land A] \quad \text{holds in} \quad \left[ t_0, t_4 \right] \\
A = \{\text{no\_tr, tr}\} \\
\text{After} \quad t_4 \\
\text{state} \quad \{q_1, q_2\} \\
\text{output} \quad \{N, T\}
\]

\[
[q_1 \land A] \quad \text{holds in} \quad \left[ t_0, t_5 \right] \\
A = \{\text{no\_tr, tr, Error}\} \\
\text{After} \quad t_5 \\
\text{state} \quad \{q_1, q_2, q_3\} \\
\text{output} \quad \{N, T, X\}
\]

\[
[q_1 \land A] \quad \text{holds in} \quad \left[ t_0, t_6 \right] \\
A = \{\text{no\_tr, tr, Error}\} \\
\text{After} \quad t_6 \\
\text{state} \quad \{q_1, q_2, q_3\} \\
\text{output} \quad \{N, T, X\}
\]
### (DC-2): Effect of Transitions

- **Effect of Transitions:**

- **Diagram:**
  - Transition not taken: \( \text{no}_\text{tr} \)
  - Transition taken: \( \text{tr} \)

- **States:**
  - \( N \)
  - \( T \)
  - \( X \)

- **Initial State:**
  - \( q_1 \)
  - \( q_2 \)
  - \( q_3 \)

- **Transition Times:**
  - \( 0.2 \) seconds

- **Input and Output Diagram:**
  - Input: \( \text{no}_\text{tr} \), \( \text{tr} \), \( \text{no}_\text{tr} \), \( \text{Error} \)
  - Time: \( t_0, t_1, t_2, t_3, t_4, t_5, t_6 \)

- **Formula:**
  \[
  \neg q; [q \wedge A] \longrightarrow [q \vee \delta(q, A)]
  \]

- **Table:**

<table>
<thead>
<tr>
<th>( q_1 \wedge A ) holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( [t_0, t_1] )</td>
<td>( A = {\text{no}_\text{tr}} )</td>
<td>( t_1 )</td>
<td>( {q_1} )</td>
<td>( {N} )</td>
</tr>
<tr>
<td>( [t_0, t_2] )</td>
<td>( A = {\text{no}_\text{tr}, \text{tr}} )</td>
<td>( t_2 )</td>
<td>( {q_1, q_2} )</td>
<td>( {N, T} )</td>
</tr>
<tr>
<td>( [t_0, t_3] )</td>
<td>( A = {\text{no}_\text{tr}, \text{tr}} )</td>
<td>( t_3 )</td>
<td>( {q_1, q_2} )</td>
<td>( {N, T} )</td>
</tr>
<tr>
<td>( [t_0, t_4] )</td>
<td>( A = {\text{no}_\text{tr}, \text{tr}} )</td>
<td>( t_4 )</td>
<td>( {q_1, q_2} )</td>
<td>( {N, T} )</td>
</tr>
<tr>
<td>( [t_0, t_5] )</td>
<td>( A = {\text{no}_\text{tr}, \text{tr}, \text{Error}} )</td>
<td>( t_5 )</td>
<td>( {q_1, q_2, q_3} )</td>
<td>( {N, T, X} )</td>
</tr>
<tr>
<td>( [t_0, t_6] )</td>
<td>( A = {\text{no}_\text{tr}, \text{tr}, \text{Error}} )</td>
<td>( t_6 )</td>
<td>( {q_1, q_2, q_3} )</td>
<td>( {N, T, X} )</td>
</tr>
</tbody>
</table>
### (DC-2): Effect of Transitions

**Diagram:**

The diagram illustrates the effect of transitions with states and inputs over time.

- **States:**
  - N
  - T
  - X

- **Inputs:**
  - no_tr
  - tr

- **Output:**
  - N
  - T
  - X

**Transitions Overview:**

<table>
<thead>
<tr>
<th>Transition</th>
<th>Input</th>
<th>Time</th>
<th>State Changes</th>
<th>Output Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 0.2]</td>
<td>no_tr</td>
<td>t_0</td>
<td>q_2</td>
<td>q_1</td>
</tr>
<tr>
<td>[0.2, 0.4]</td>
<td>tr</td>
<td>t_1</td>
<td>q_1</td>
<td>N</td>
</tr>
<tr>
<td>[0.4, 0.8]</td>
<td>no_tr</td>
<td>t_2</td>
<td>q_1, q_2</td>
<td>N, T</td>
</tr>
<tr>
<td>[0.8, 1]</td>
<td>tr</td>
<td>t_3</td>
<td>q_1, q_2</td>
<td>N, T</td>
</tr>
<tr>
<td>[1, 1.2]</td>
<td>tr</td>
<td>t_4</td>
<td>q_1, q_2</td>
<td>N, T</td>
</tr>
<tr>
<td>[1.2, 2]</td>
<td>no_tr, tr, Error</td>
<td>t_5</td>
<td>q_1, q_2, q_3</td>
<td>N, T, X</td>
</tr>
<tr>
<td>[2, 2.2]</td>
<td>no_tr, tr, Error</td>
<td>t_6</td>
<td>q_1, q_2, q_3</td>
<td>N, T, X</td>
</tr>
</tbody>
</table>

**Formal Expression:**

\[
\neg q; [q \land A] \rightarrow [q \lor \delta(q, A)]
\]
(DC-2): Effect of Transitions

\[
\neg q ; [q \land A] \longrightarrow [q \lor \delta(q, A)]
\]

<table>
<thead>
<tr>
<th>([q_1 \land A]) holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_0, t_1])</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_1)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_0, t_2])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_2)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_3])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_4])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_4)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_5])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_5)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
<tr>
<td>([t_0, t_6])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_6)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
</tbody>
</table>
**Effect of Transitions**

\[ \neg q ; [q \land A] \rightarrow [q \lor \delta(q, A)] \]  

**Table: Change in State and Output**

<table>
<thead>
<tr>
<th>Interval ([t_0, t_1])</th>
<th>Input ([A])</th>
<th>After ([t_1])</th>
<th>State ([{\text{state}}}])</th>
<th>Output ([{\text{output}}}])</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_0, t_1])</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_1)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_0, t_2])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_2)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_3])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_4])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_4)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_5])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_5)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
<tr>
<td>([t_0, t_6])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_6)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
</tbody>
</table>
(DC-2): Effect of Transitions

\[ \neg q \land [q \land A] \rightarrow [q \lor \delta(q, A)] \]  

<table>
<thead>
<tr>
<th>([q_1 \land A]) holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_0, t_1])</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_1)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_0, t_2])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_2)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_3])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_4])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_4)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_5])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_5)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
<tr>
<td>([t_0, t_6])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_6)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
</tbody>
</table>
(DC-2): Effect of Transitions

\[ \neg q \land \left[ q \land A \right] \rightarrow \left[ q \lor \delta(q, A) \right] \tag{DC-2} \]

<table>
<thead>
<tr>
<th>([q_1 \land A]) holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_0, t_1])</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_1)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_0, t_2])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_2)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_3])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_4])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_4)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_5])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_5)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
<tr>
<td>([t_0, t_6])</td>
<td>(A = {\text{no_tr, tr, Error}})</td>
<td>(t_6)</td>
<td>({q_1, q_2, q_3})</td>
<td>({N, T, X})</td>
</tr>
</tbody>
</table>
(DC-3): Inputs and Cycle Time

\[
[q \land A] \xrightarrow{\epsilon} [q \lor \delta(q, A)]
\]  

(DC-3)
(DC-3): Inputs and Cycle Time

\[ [q \land A] \xrightarrow{\varepsilon} [q \lor \delta(q, A)] \]  

(DC-3)
**DC-3): Inputs and Cycle Time**

\[
[q \land A] \xrightarrow{\varepsilon} [q \lor \delta(q, A)] \quad \text{(DC-3)}
\]

<table>
<thead>
<tr>
<th>([q_1 \land A]) holds in</th>
<th>with input (A)</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_1, t_2])</td>
<td>(A = {\text{no\textunderscore tr, tr}})</td>
<td>(t_2)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_2, t_3])</td>
<td>(A = {\text{no\textunderscore tr, tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_3, t_4])</td>
<td>(A = {\text{no\textunderscore tr}})</td>
<td>(t_4)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_4, t_5])</td>
<td>(A = {\text{no\textunderscore tr, Error}})</td>
<td>(t_5)</td>
<td>({q_1, q_3})</td>
<td>({N, X})</td>
</tr>
<tr>
<td>([t_5, t_6])</td>
<td>(A = {\text{Error}})</td>
<td>(t_6)</td>
<td>({q_1, q_3})</td>
<td>({N, X})</td>
</tr>
</tbody>
</table>
(DC-3): Inputs and Cycle Time

\[
[q \land A] \xrightarrow{\varepsilon} [q \lor \delta(q, A)]
\]

<table>
<thead>
<tr>
<th>[[q_1 \land A]] holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>[[t_1, t_2]]</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_2)</td>
<td>(q_1, q_2)</td>
<td>({N, T})</td>
</tr>
<tr>
<td>[[t_2, t_3]]</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_3)</td>
<td>(q_1, q_2)</td>
<td>({N, T})</td>
</tr>
<tr>
<td>[[t_3, t_4]]</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_4)</td>
<td>(q_1)</td>
<td>({N})</td>
</tr>
<tr>
<td>[[t_4, t_5]]</td>
<td>(A = {\text{no_tr, Error}})</td>
<td>(t_5)</td>
<td>(q_1, q_3)</td>
<td>({N, X})</td>
</tr>
<tr>
<td>[[t_5, t_6]]</td>
<td>(A = {\text{Error}})</td>
<td>(t_6)</td>
<td>(q_1, q_3)</td>
<td>({N, X})</td>
</tr>
</tbody>
</table>
(DC-3): Inputs and Cycle Time

\[ [q \land A] \xrightarrow{\varepsilon} [q \lor \delta(q, A)] \]  

<table>
<thead>
<tr>
<th>([q_1 \land A]) holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_1, t_2])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_2)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_2, t_3])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_3, t_4])</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_4)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_4, t_5])</td>
<td>(A = {\text{no_tr, Error}})</td>
<td>(t_5)</td>
<td>({q_1, q_3})</td>
<td>({N, X})</td>
</tr>
<tr>
<td>([t_5, t_6])</td>
<td>(A = {\text{Error}})</td>
<td>(t_6)</td>
<td>({q_1, q_3})</td>
<td>({N, X})</td>
</tr>
</tbody>
</table>
(DC-3): Inputs and Cycle Time

\[ [q \wedge A] \xrightarrow{\epsilon} [q \lor \delta(q, A)] \]  

<table>
<thead>
<tr>
<th>$[q_1 \wedge A]$ holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[t_1, t_2]$</td>
<td>$A = {\text{no_tr}, \text{tr}}$</td>
<td>$t_2$</td>
<td>${q_1, q_2}$</td>
<td>${N, T}$</td>
</tr>
<tr>
<td>$[t_2, t_3]$</td>
<td>$A = {\text{no_tr}, \text{tr}}$</td>
<td>$t_3$</td>
<td>${q_1, q_2}$</td>
<td>${N, T}$</td>
</tr>
<tr>
<td>$[t_3, t_4]$</td>
<td>$A = {\text{no_tr}}$</td>
<td>$t_4$</td>
<td>${q_1}$</td>
<td>${N}$</td>
</tr>
<tr>
<td>$[t_4, t_5]$</td>
<td>$A = {\text{no_tr}, \text{Error}}$</td>
<td>$t_5$</td>
<td>${q_1, q_3}$</td>
<td>${N, X}$</td>
</tr>
<tr>
<td>$[t_5, t_6]$</td>
<td>$A = {\text{Error}}$</td>
<td>$t_6$</td>
<td>${q_1, q_3}$</td>
<td>${N, X}$</td>
</tr>
</tbody>
</table>
(DC-3): Inputs and Cycle Time

\[ [q \land A] \xrightarrow{\varepsilon} [q \lor \delta(q, A)] \]  

<table>
<thead>
<tr>
<th>([q_1 \land A]) holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_1, t_2])</td>
<td>(A = {\text{no_tr}, \text{tr}})</td>
<td>(t_2)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_2, t_3])</td>
<td>(A = {\text{no_tr}, \text{tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_3, t_4])</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_4)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_4, t_5])</td>
<td>(A = {\text{no_tr}, \text{Error}})</td>
<td>(t_5)</td>
<td>({q_1, q_3})</td>
<td>({N, X})</td>
</tr>
<tr>
<td>([t_5, t_6])</td>
<td>(A = {\text{Error}})</td>
<td>(t_6)</td>
<td>({q_1, q_3})</td>
<td>({N, X})</td>
</tr>
</tbody>
</table>
(DC-3): Inputs and Cycle Time

\[ [q \land A] \xrightarrow{\varepsilon} [q \lor \delta(q, A)] \]  

\begin{align*}
\begin{array}{c|c|c|c}
[ q_1 \land A ] & \text{holds in} & \text{with input} & \text{After} & \text{state} & \text{output} \\
\hline
[t_1, t_2] & A = \{no\_tr, tr\} & t_2 & \{q_1, q_2\} & \{N, T\} \\
[t_2, t_3] & A = \{no\_tr, tr\} & t_3 & \{q_1, q_2\} & \{N, T\} \\
[t_3, t_4] & A = \{no\_tr\} & t_4 & \{q_1\} & \{N\} \\
[t_4, t_5] & A = \{no\_tr, Error\} & t_5 & \{q_1, q_3\} & \{N, X\} \\
[t_5, t_6] & A = \{Error\} & t_6 & \{q_1, q_3\} & \{N, X\}
\end{array}
\end{align*}
\begin{align*}
S_t(q) > 0 \implies \boxed{\neg q} ; \boxed{[q \land A]} \xrightarrow{\leq S_t(q)} \boxed{[q \lor \delta(q, A \setminus S^e(q))]} \tag{DC-4}
\end{align*}

<table>
<thead>
<tr>
<th>$[q_1 \land A]$ holds in</th>
<th>with input</th>
<th>After $t_i$</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[t_0, t_1]$</td>
<td>$A = {\text{no\textunderscore tr}}$</td>
<td>$t_1$</td>
<td>${q_2}$</td>
<td>${T}$</td>
</tr>
<tr>
<td>$[t_0, t_2]$</td>
<td>$A = {\text{no\textunderscore tr}, \text{tr}}$</td>
<td>$t_2$</td>
<td>${q_2}$</td>
<td>${T}$</td>
</tr>
<tr>
<td>$[t_0, t_3]$</td>
<td>$A = {\text{no\textunderscore tr}, \text{tr}, \text{Error}}$</td>
<td>$t_3$</td>
<td>${q_2, q_3}$</td>
<td>${T, X}$</td>
</tr>
<tr>
<td>$[t_0, t_4]$</td>
<td>$A = {\text{no\textunderscore tr}, \text{tr}, \text{Error}}$</td>
<td>$t_4$</td>
<td>${q_2, q_3}$</td>
<td>${T, X}$</td>
</tr>
<tr>
<td>$[t_0, t_5]$</td>
<td>$A = {\text{no\textunderscore tr}, \text{tr}, \text{Error}}$</td>
<td>$t_5$</td>
<td>${q_2, q_3}$</td>
<td>${T, X}$</td>
</tr>
<tr>
<td>$[t_0, t_6]$</td>
<td>$A = {\text{no\textunderscore tr}, \text{tr}, \text{Error}}$</td>
<td>$t_6$</td>
<td>${q_2, q_3}$</td>
<td>${T, X}$</td>
</tr>
</tbody>
</table>
(DC-5): Delays

\[ S_t(q) > 0 \implies [\neg q] ; [q] ; [q \land A] \overset{\leq S_t(q)}{\rightarrow} [q \lor \delta(q, A \setminus S_e(q))] \]

<table>
<thead>
<tr>
<th>([q_1 \land A]) holds in</th>
<th>with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_1, t_2])</td>
<td>(A = {\text{no_tr, tr}})</td>
<td>(t_2)</td>
<td>({q_2})</td>
<td>({T})</td>
</tr>
<tr>
<td>([t_2, t_3])</td>
<td>(A = {\text{tr, Error}})</td>
<td>(t_3)</td>
<td>({q_2, q_3})</td>
<td>({T, X})</td>
</tr>
<tr>
<td>([t_3, t_4])</td>
<td>(A = {\text{no_tr, Error}})</td>
<td>(t_4)</td>
<td>({q_2, q_3})</td>
<td>({T, X})</td>
</tr>
<tr>
<td>([t_4, t_5])</td>
<td>(A = {\text{no_tr}})</td>
<td>(t_5)</td>
<td>({q_2})</td>
<td>({T})</td>
</tr>
<tr>
<td>([t_5, t_6])</td>
<td>(A = {\text{no_tr, Error}})</td>
<td>(t_6)</td>
<td>({q_2, q_3})</td>
<td>({T, X})</td>
</tr>
</tbody>
</table>
(DC-6) / (DC-7): Progress from non-delayed inputs

\[ S_t(q) = 0 \land q \notin \delta(q, A) \implies \Box([q \land A] \implies \ell < 2\varepsilon) \]  

\[ S_t(q) = 0 \land q \notin \delta(q, A) \implies [\neg q] ; [q \land A]^\varepsilon \rightarrow [\neg q] \]  

- 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

\[ S_t(q) > 0 \land q \notin \delta(q, A) \]
\[ \implies \Box([q]^{S_t(q)}; [q \land A] \implies \ell < S_t(q) + 2\varepsilon) \]

\[ S_t(q) > 0 \land A \cap S_e(q) = \emptyset \land q \notin \delta(q, A) \]
\[ \implies \Box([q \land A] \implies \ell < 2\varepsilon) \]

\[ S_t(q) > 0 \land A \cap S_e(q) = \emptyset \land q \notin \delta(q, A) \]
\[ \implies [\neg q]; [q \land A]^\varepsilon \rightarrow [\neg q] \]

- 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

\[ \square([q] \implies [\omega(q)]) \]  

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

\[ \square([q] \implies [\omega(q)]) \]  \hfill (DC-11)

\[ [q_0 \land A] \xrightarrow{\leq S_t(q_0)} [q_0 \lor \delta(q_0, A)] \] \hfill (DC-2')

\[ S_t(q_0) > 0 \implies [q_0 \land A] \xrightarrow{\leq S_t(q_0)} [q_0 \lor \delta(q_0, A \setminus S_e(q_0))] \] \hfill (DC-4')

\[ S_t(q_0) > 0 \implies [q_0] ; [q_0 \land A]^\varepsilon \xrightarrow{\leq S_t(q_0)} [q_0 \lor \delta(q_0, A \setminus S_e(q_0))] \] \hfill (DC-5')

\[ S_t(q_0) = 0 \land q_0 \notin \delta(q_0, A) \implies [q_0 \land A]^\varepsilon \xrightarrow{0} [\neg q_0] \] \hfill (DC-7')

\[ S_t(q_0) > 0 \land A \cap S_e(q_0) = \emptyset \land q_0 \notin \delta(q_0, A) \implies [q_0 \land A]^\varepsilon \xrightarrow{0} [\neg q_0] \] \hfill (DC-10')
**Definition 5.3.**

The **Duration Calculus semantics** of a PLC Automaton $A$ is

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

**Claim:**

- Let $P_A$ be the ST program semantics of $A$.
- Let $\pi$ be a recording over time of then inputs, local states, and outputs of a PLC device running the ST $P_A$.
- Let $I_\pi$ be an encoding of $\pi$ as an interpretation of $\text{In}_A$, $\text{St}_A$, and $\text{Out}_A$.
- Then $I_\pi \models [[A]]_{DC}$. (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
Given a PLC-Automaton, one often wants to know whether it guarantees properties of the form

\[
[\text{St}_A \in Q \land \text{In}_A = \text{emergency\_signal}] \xrightarrow{0.1} [\text{St}_A = \text{motor\_off}]
\]
One Application: Reaction Times

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

\[
\left[ \text{St}_A \in Q \land \text{ln}_A = \text{emergency}_\text{signal} \right] \xrightarrow{0.1} \left[ \text{St}_A = \text{motor}_\text{off} \right]
\]

(“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

\[
\left[ \text{St}_A \in Q \land \text{ln}_A = \text{emergency\_signal} \right] \xrightarrow{0.1} \left[ \text{St}_A = \text{motor\_off} \right]
\]

(“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,
  - \( c \in \text{Time} \) be a time bound, and
  - \( \Pi_{\text{target}} \subseteq Q \) be a set of target states.

- Then we seek to establish properties of the form

\[
\left[ \text{St}_A \in \Pi \land \text{ln}_A \in A \right] \xrightarrow{c} \left[ \text{St}_A \in \Pi_{\text{target}} \right],
\]

abbreviated as

\[
\left[ \Pi \land A \right] \xrightarrow{c} \left[ \Pi_{\text{target}} \right].
\]
Actually, the reaction time theorem addresses only the special case

\[ [\Pi \land A] \xrightarrow{c_n} [\delta^n(\Pi, A)] = \Pi_{\text{target}} \]

for PLC Automata with

\[ \delta(\Pi, A) \subseteq \Pi. \]

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

\[ \delta(\Pi, A) := \{\delta(q, a) \mid q \in \Pi \land a \in A\}. \]
Examples:

- $\Pi = \{N, T\}$, $A = \{\text{no\_tr}\}$

- $\delta(\Pi, A) = \{N\} \subseteq \Pi$
Examples:

- $\Pi = \{N, T\}$, $A = \{\text{no\_tr}\}$

- $\delta(\Pi, A) = \{N\} \subseteq \Pi$
Premise Examples

Examples:

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

- \( \Pi = \{N, T, X\}, \ A = \{\text{Error}\} \)
  - \( \delta(\Pi, A) = \{X\} \subseteq \Pi \)
Premise Examples

Examples:

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

- $\Pi = \{N, T, X\}, \quad A = \{\text{Error}\}$
  - $\delta(\Pi, A) = \{X\} \subseteq \Pi$
Examples:

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

- $\Pi = \{N, T, X\}, \ A = \{\text{Error}\}$
  
  $\delta(\Pi, A) = \{X\} \subseteq \Pi$

- $\Pi = \{T\}, \ A = \{\text{no\_tr}\}$
  
  $\delta(\Pi, A) = \{N\} \not\subseteq \Pi$
Examples:

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

- $\Pi = \{N, T, X\}$, $A = \{\text{Error}\}$
  - $\delta(\Pi, A) = \{X\} \subseteq \Pi$

- $\Pi = \{T\}$, $A = \{\text{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \not \subseteq \Pi$
Theorem 5.6.
Let \( A = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega) \), \( \Pi \subseteq Q \), and \( A \subseteq \Sigma \) with
\[
\delta(\Pi, A) \subseteq \Pi.
\]
Then
\[
[\Pi \land A] \xrightarrow{c} [\delta(\Pi, A)] = \Pi_{target}
\]
where
\[
c := \varepsilon + \max(\{0\} \cup \{s(\pi, A) \mid \pi \in \Pi \setminus \delta(\Pi, A)\})
\]
and
\[
s(\pi, A) := \begin{cases} 
S_t(\pi) + 2\varepsilon & \text{, if } S_t(\pi) > 0 \text{ and } A \cap S_e(\pi) \neq \emptyset \\
\varepsilon & \text{, otherwise.}
\end{cases}
\]
(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?
(1) If we are in state $N$ or $T$, how long does $N$ or $T$ need to persist together with input $\text{no}_\text{tr}$, to ensure that we observe $N$ again?

**Your estimation?**

- $\varepsilon$
- $2\varepsilon$
- $3\varepsilon$
- $5\text{ s}$
- $5\text{ s }+ \varepsilon$
- $5\text{ s }+ 2\varepsilon$
- $5\text{ s }+ 3\varepsilon$
- ...

![Diagram illustrating the states and transitions](image-url)
(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?

$$\lceil \{N, T\} \land \{\text{no}_\text{tr}\} \rceil \xrightarrow{5+3\varepsilon} \lceil N \rceil$$
(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?

$$[\{N, T\} \land \{\text{no_tr}\}] \xrightarrow{5+3\varepsilon} [N]$$

- **Because:** earlier we have shown

$$\delta(\{N, T\}, \{\text{no_tr}\}) = \{N\}$$
(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?

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

- **Because:** earlier we have shown

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

- Thus Theorem 5.6 yields

$$\lceil \{N, T\} \land \{\text{no\_tr}\} \rceil \xrightarrow{c} \lceil N \rceil$$
(1) If we are in state $N$ or $T$, how long does $N$ or $T$ need to persist together with input $\text{no}_\_\text{tr}$, to ensure that we observe $N$ again?

\[
[\{N, T\} \land \{\text{no}_\_\text{tr}\}] \xrightarrow{5+3\varepsilon} [N]
\]

- **Because**: earlier we have shown

\[
\delta(\{N, T\}, \{\text{no}_\_\text{tr}\}) = \{N\}
\]

- **Thus** Theorem 5.6 yields

\[
[\{N, T\} \land \{\text{no}_\_\text{tr}\}] \xrightarrow{c} [N]
\]

with

\[
c = \varepsilon + \max(\{0\} \cup \{s(\pi, \{\text{no}_\_\text{tr}\}) \mid \pi \in \{N, T\} \setminus \{N\}\})
\]

\[
= \varepsilon + \max(\{0\} \cup \{s(T, \{\text{no}_\_\text{tr}\})\})
\]

\[
= \varepsilon + 5 + 2\varepsilon = 5 + 3\varepsilon
\]
(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?
(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?

$$\lceil\{N, T, X\} \land \{\text{Error}\}\rceil \xrightarrow{2\varepsilon} \lceil X \rceil$$
(2) If we are in state $N$, $T$, or $X$, how long does input $\text{Error}$ need to persist to ensure that we observe $X$ again?

\[
\left[\{N, T, X\} \land \{\text{Error}\}\right] \xrightarrow{2\varepsilon} [X]
\]

- **Because:** earlier we have shown

\[
\delta(\{N, T, X\}, \{\text{Error}\}) = \{X\}
\]
(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?

\[
\left[ \{N, T, X\} \land \{\text{Error}\} \right] \overset{2\varepsilon}{\longrightarrow} \left[ X \right]
\]

- **Because**: earlier we have shown

\[
\delta(\{N, T, X\}, \{\text{Error}\}) = \{X\}
\]

- **Thus Theorem 5.6 yields**

\[
\left[ \{N, T, X\} \land \{\text{Error}\} \right] \overset{c}{\longrightarrow} \left[ X \right]
\]
(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?

$$\left[\{N, T, X\} \land \{\text{Error}\}\right] \xrightarrow{2\epsilon} [X]$$

- **Because**: earlier we have shown

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

- Thus Theorem 5.6 yields

$$\left[\{N, T, X\} \land \{\text{Error}\}\right] \xrightarrow{c} [X]$$

with

$$c = \epsilon + \max(\{0\} \cup \{s(\pi, \{\text{Error}\}) \mid \pi \in \{N, T, X\} \setminus \{X\}\})$$

$$= \epsilon + \max(\{0\} \cup \{s(N, \{\text{Error}\}), s(T, \{\text{Error}\})\})$$

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

\[
\left\{ N, T \right\} \land \left\{ \text{no\_tr}, \text{tr} \right\} \xrightarrow{\varepsilon} \left[ N, T \right]
\]
(2) If we are in state $N$ or $T$, how long do inputs $\text{no\_tr}$ or $\text{tr}$ need to persist to ensure that we observe $N$ or $T$ again?

$$\left[\{N, T\} \land \{\text{no\_tr, tr}\}\right] \xrightarrow{\varepsilon} [N, T]$$

- **Because**: earlier we have shown

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

$$\{\{N, T\} \land \{\text{no\_tr}, \text{tr}\}\} \xrightarrow{\varepsilon} \{N, T\}$$

- **Because**: earlier we have shown

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

- **Thus Theorem 5.6 yields**

$$\{\{N, T\} \land \{\text{no\_tr}, \text{tr}\}\} \xrightarrow{c} \{N, T\}$$
(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?

$$[\{N, T\} \land \{\text{no}_\text{tr}, \text{tr}\}] \xrightarrow{\varepsilon} [N, T]$$

- **Because**: earlier we have shown

  $$\delta(\{N, T\}, \{\text{no}_\text{tr}, \text{tr}\}) = \{N, T\}$$

- **Thus Theorem 5.6 yields**

  $$[\{N, T\} \land \{\text{no}_\text{tr}, \text{tr}\}] \xrightarrow{c} [N, T]$$

with

$$c = \varepsilon + \max(\{0\} \cup \{s(\pi, \{\text{no}_\text{tr}, \text{tr}\}) \mid \pi \in \{N, T\} \setminus \{N, T\}\})$$

$$= \varepsilon + \max(\{0\} \cup \emptyset)$$

$$= \varepsilon$$
Monotonicity of Generalised Transition Function

- Define
\[ \delta^0(\Pi, A) := \Pi, \quad \delta^{n+1}(\Pi, A) := \delta(\delta^n(\Pi, A), A). \]

- If we have \( \delta(\Pi, A) \subseteq \Pi \), then we have
\[
\delta^{n+1}(\Pi, A) \subseteq \delta^n(\Pi, A) \subseteq \cdots \subseteq \delta(\delta(\Pi, A), A) \subseteq \delta(\Pi, A) \subseteq \Pi
\]

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

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

\[
\text{Proposition 5.4.} \quad \Pi \subseteq \Pi' \subseteq Q \text{ and } A \subseteq A' \subseteq \Sigma \text{ implies } \delta(\Pi, A) \subseteq \delta(\Pi', A').
\]
Contraction Examples

Examples:

- $\Pi = \{N, T\}, A = \{\text{no}\_\text{tr}\}$

- $\delta^0(\Pi, A) = \{N, T\}$
Contraction Examples

Examples:

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

Examples:

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

Examples:

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

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

Examples:

- \( \Pi = \{N, T\}, A = \{\text{no\_tr}\} \)
- \( \delta^0(\Pi, A) = \{N, T\} \)
- \( \delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi \)
- \( \delta^n(\delta^0(\Pi, A), A) = \{N\} \)
Contraction Examples

Examples:

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

- $\Pi = \{N, T, X\}, A = \{\text{Error}\}$
  - $\delta^0(\Pi, A) = \{N, T, X\}$
  - $\delta(\delta^0(\Pi, A), A) = \{X\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{X\}$
Contraction Examples

Examples:

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

- \( \Pi = \{N,T,X\}, A = \{\text{Error}\} \)
  - \( \delta^0(\Pi, A) = \{N, T, X\} \)
  - \( \delta(\delta^0(\Pi, A), A) = \{X\} \subseteq \Pi \)
  - \( \delta^n(\delta^0(\Pi, A), A) = \{X\} \)
Contraction Examples

Examples:
- $\Pi = \{N,T\}, A = \{\text{no}_\text{tr}\}$
  - $\delta^0(\Pi, A) = \{N,T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{N\}$
- $\Pi = \{N,T,X\}, A = \{\text{Error}\}$
  - $\delta^0(\Pi, A) = \{N,T,X\}$
  - $\delta(\delta^0(\Pi, A), A) = \{X\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{X\}$
- $\Pi = \{T\}, A = \{\text{no}_\text{tr}\}$
  - $\delta(\Pi, A) = \{N\} \notin \Pi$
### Contraction Examples

#### Examples:

- $\Pi = \{N, T\}, A = \{\text{no}_\text{tr}\}$
  - $\delta^0(\Pi, A) = \{N, T\}$
  - $\delta(\delta^0(\Pi, A), A) = \{N\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{N\}$

- $\Pi = \{N, T, X\}, A = \{\text{Error}\}$
  - $\delta^0(\Pi, A) = \{N, T, X\}$
  - $\delta(\delta^0(\Pi, A), A) = \{X\} \subseteq \Pi$
  - $\delta^n(\delta^0(\Pi, A), A) = \{X\}$

- $\Pi = \{T\}, A = \{\text{no}_\text{tr}\}$
  - $\delta(\Pi, A) = \{N\} \not\subseteq \Pi$
Theorem 5.8.
Let $A = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$, $\Pi \subseteq Q$, and $A \subseteq \Sigma$ with

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

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

$$[\Pi \land A] \xrightarrow{c_n} [\delta^n(\Pi, A)] = \Pi_{target}$$

where

$$c_n := \varepsilon + \max\left(\{0\} \cup \left\{ \sum_{i=1}^k s(\pi_i, A) \right\} \right)$$

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

- Assume, we would not have

\[ [\Pi \land A] \xrightarrow{c_n} [\delta^n (\Pi, A)] . \]
Proof Idea of Reaction Time Theorem

(by contradiction)

• Assume, we would not have

\[ [\Pi \land A] \overset{c_n}{\rightarrow} [\delta^n (\Pi, A)]. \]

• This is equivalent to not having

\[ \neg (true ; [\Pi \land A]^{c_n} ; \neg \delta^n (\Pi, A) ; true) \]
Proof Idea of Reaction Time Theorem

(by contradiction)

- Assume, we would not have

\[
[\Pi \land A]^{c_n} \rightarrow [\delta^n (\Pi, A)].
\]

- This is equivalent to not having

\[
\neg (true ; [\Pi \land A]^{c_n} ; [\neg \delta^n (\Pi, A)] ; true)
\]

- Which is equivalent to having

\[
true ; [\Pi \land A]^{c_n} ; [\neg \delta^n (\Pi, A)] ; true.
\]
Proof Idea of Reaction Time Theorem

(by contradiction)

- Assume, we would not have

\[ [\Pi \land A] \xrightarrow{c_n} [\delta^n (\Pi, A)]. \]

- This is equivalent to not having

\[ \neg \left( true ; [\Pi \land A]^{c_n} ; [\neg \delta^n (\Pi, A)] ; true \right) \]

- Which is equivalent to having

\[ true ; [\Pi \land A]^{c_n} ; [\neg \delta^n (\Pi, A)] ; true. \]

- 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 \([\Pi \land A]\) 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

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

\[ \text{obs} : \text{Time} \rightarrow \mathcal{D}(\text{obs}) \]

\[ \langle \text{obs}_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_0} \langle \text{obs}_1, \nu_1 \rangle, t_1 \ldots \]

• 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