The Plan

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

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

prove

synthesis / code generation (in the book)

later

by example

(correct?) compiler
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?

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.

  (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.

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

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

```
Alternative Programming Languages by IEC 61131-3
```

Tied together by
- Sequential Function Charts (SFC)

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

\[
\begin{array}{c|c|c}
\text{no_tr} & \text{tr} & N \\
\hline
\text{tr} & \text{no_tr} & T \\
\hline
& & X \\
\end{array}
\]

After arrival of a train, it should ignore “no_tr” for 5 seconds.
A = (Q = \{q_0, q_1\},
Σ = \{\text{tr, no}_\text{tr}\},
δ = \{(q_0, \text{tr}) \mapsto q_1, (q_0, \text{no}_\text{tr}) \mapsto q_0, (q_1, \text{tr}) \mapsto q_1, (q_1, \text{no}_\text{tr}) \mapsto q_0\},
q_0 = q_0,
ε = 0.2,
S_t = \{q_0 \mapsto 0, q_1 \mapsto 5\},
S_e = \{q_0 \mapsto \emptyset, q_1 \mapsto \Sigma\},
Ω = \{N, T\},
ω = \{q_0 \mapsto N, q_1 \mapsto T\})
PLC Automaton Semantics

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

IF state = 0 THEN
  IF %input = tr THEN
    state := 1;
    %output := T;
  ELSEIF %input = Error THEN
    state := 2;
    %output := X;
  ENDIF
ENDIF
ELSIF state = 1 THEN
  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#0.0s );
ENDIF
ELSIF state = 2 THEN
  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#0.0s );
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 #5.0 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

\[ N \leq \varepsilon \]
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

ENDIF


\[ \frac{N}{0} s, \emptyset \]
\[ \frac{T}{5} s, \{\text{no_tr}, \text{tr}\} \]
\[ 0.2 s \]
\[ \frac{X}{0} s, \emptyset \]
\[ \text{true} \]

\[ \leq \varepsilon \]
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
  ELSEIF input = Error THEN
    state := 2
    output := X
  ENDIF
ENDIF
ELSIF state = 2 THEN
  tmr ( IN := FALSE , PT := t #0.0 s )
ENDIF
END
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
  ELSEIF 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
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**
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 $[A]_{DC}$ such that
    \[ I \in [A] \quad \implies \quad I \models [A]_{DC} \]
    but not necessarily the other way round.
  - In even other words: $[A] \subseteq \{ I \mid I \models [A]_{DC} \}$. 
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 $\llbracket A \rrbracket_{DC}$ such that
    \[
    \text{"} I \in \llbracket A \rrbracket \text{"} \implies I \models \llbracket A \rrbracket_{DC}
    \]
    but not necessarily the other way round.
- In even other words: $\llbracket A \rrbracket \subseteq \{ I \mid I \models \llbracket A \rrbracket_{DC} \}$. 

**Applications:**
- Assess correctness of over-approximation wrt. DC requirements.
  If $\models \llbracket A \rrbracket_{DC} \implies \text{Req}$ for a given PLCA $A$, the $A$ is correct.
- Prove generic properties of PLCA using DC, like reaction time.

Observables

- Consider the PLCA $A = (Q, \Sigma, \delta, q_0, \varepsilon, S_t, S_e, \Omega, \omega)$.
- The DC formula $\llbracket A \rrbracket_{DC}$ we construct ranges over the observables
  - $\text{In}_A : \Sigma$ – values of the inputs
  - $\text{St}_A : Q$ – current local state
  - $\text{Out}_A : \Omega$ – values of the outputs
Overview

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

- Initial State:
  \[ \lceil \top \rceil \lor \lceil q_0 \rceil : \text{true} \]  
  (DC-1)

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

- Delays:
  \[ S_t(q) > 0 \implies [\neg q] ; [q \land A] \overset{S_t(q)}{\leq} [q \lor \delta(q, A \setminus S_e(q))] \]  
  (DC-4)

• \( A \) 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 \(\text{St}_A \in \{\delta(q, a) | a \in A\}\).
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 \( \text{St}_A \in \{ \delta(q, a) \mid a \in A \} \).

Progress from non-delayed inputs:

\[ S_t(q) = 0 \land q \notin \delta(q, A) \implies \Box([q \land A] \implies \ell < 2\varepsilon) \]  
\[ (DC -6) \]

\[ S_t(q) = 0 \land q \notin \delta(q, A) \implies [\neg q]; [q \land A]^{\varepsilon} \rightarrow [\neg q] \]  
\[ (DC -7) \]
How to Read these Formulae

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

<table>
<thead>
<tr>
<th>$\neg q$</th>
<th>$[q \land A] \implies [q \lor \delta(q, A)]$</th>
<th>(DC-2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[q \land A]$</td>
<td>$\implies [q \lor \delta(q, A)]$</td>
<td>(DC-3)</td>
</tr>
</tbody>
</table>

$\text{St}_{A} = q \land \text{ln}_{A} \in A$,
$\text{St}_{A} \in \{\delta(q, a) \mid a \in A\}$.
How to Read these Formulae

\[ [-q] : [q \land A] \longrightarrow [q \lor \delta(q, A)] \]  \hspace{1cm} (DC-2)

\[ [q \land A] \dashrightarrow [q \lor \delta(q, A)] \]  \hspace{1cm} (DC-3)

- How to read these formulae?
- \( A \) is a set with \( \emptyset \neq A \subseteq \Sigma \).
- \([q \land A]\) abbreviates \( [St_A = q \land \text{ln}_A \in A] \).
- \( \delta(q, A) \) abbreviates \( St_A \in \{\delta(q, a) \mid a \in A\} \).
- For the stutter filter, (DC-3) abbreviates:

\begin{align*}
[-q_1] : [q_1 \land \{\text{no_tr}\}] & \dashrightarrow [q_1 \lor q_1] \\
\land [-q_1] : [q_1 \land \{\text{tr}\}] & \dashrightarrow [q_1 \lor q_2] \\
\land [-q_1] : [q_1 \land \{\text{Error}\}] & \dashrightarrow [q_1 \lor q_3] \\
\land [-q_1] : [q_1 \land \{\text{no_tr, tr}\}] & \dashrightarrow [q_1 \lor q_1 \lor q_2] \\
\land [-q_1] : [q_1 \land \{\text{no_tr, Error}\}] & \dashrightarrow [q_1 \lor q_1 \lor q_3] \\
\land [-q_1] : [q_1 \land \{\text{tr, Error}\}] & \dashrightarrow [q_1 \lor q_2 \lor q_3] \\
\land [-q_1] : [q_1 \land \{\text{no_tr, tr, Error}\}] & \dashrightarrow [q_1 \lor q_2 \lor q_3]
\end{align*}

\[(DC-2): \text{Effect of Transitions}\]

\[ [-q] : [q \land A] \longrightarrow [q \lor \delta(q, A)] \]  \hspace{1cm} (DC-2)

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

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

\[
\begin{array}{cccc}
\text{[q1 \land A]} & \text{with input} & \text{After} & \text{output} \\
[t_0, t_1] & A = \{\text{no_tr}\} & t_1 & \{q_1\} \{N\} \\
[t_0, t_2] & A = \{\text{no_tr, tr}\} & t_2 & \{q_1, q_2\} \{N, T\} \\
[t_0, t_3] & A = \{\text{no_tr}\} & t_3 & \{q_1, q_2\} \{N, T\} \\
[t_0, t_4] & A = \{\text{no_tr}\} & t_4 & \{q_1, q_2\} \{N, T\} \\
[t_0, t_5] & A = \{\text{no_tr, tr, Error}\} & t_5 & \{q_1, q_2, q_3\} \{N, T, X\} \\
[t_0, t_6] & A = \{\text{no_tr, tr, Error}\} & t_6 & \{q_1, q_2, q_3\} \{N, T, X\}
\end{array}
\]
### DC-2: Effect of Transitions

\([-q] : [q \land A] \rightarrow [q \lor \delta(q, A)]\) (DC-2)

<table>
<thead>
<tr>
<th>[(q_1 \land A)] holds in with input</th>
<th>After</th>
<th>state</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>([t_0, t_1]) (A = {\text{no}__\text{tr}})</td>
<td>(t_1)</td>
<td>({q_1})</td>
<td>({N})</td>
</tr>
<tr>
<td>([t_0, t_2]) (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]) (A = {\text{no}__\text{tr}})</td>
<td>(t_3)</td>
<td>({q_1, q_2})</td>
<td>({N, T})</td>
</tr>
<tr>
<td>([t_0, t_4]) (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]) (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]) (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>
[−q] : [q ∧ A] → [q ∨ δ(q, A)]
\[ [q \land A] \xrightarrow{e} [q \lor \delta(q, A)] \] (DC-3)
\[ [q \land A] \xrightarrow{ε} [q \lor \delta(q, A)] \]

(DC-3)

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

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

(\text{DC-3})

<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\textunderscore 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\textunderscore 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\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}, \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>

25/49
### (DC-3): Inputs and Cycle Time

![Diagram](DC-3)

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

<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-4): Delays

\[
S_i(q) > 0 \implies [-q] \vdash [q \land A] \xrightarrow{\leq S_i(q)} [q \lor \delta(q, A \setminus S_e(q))] 
\]  

DC-4

<table>
<thead>
<tr>
<th>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_2})</td>
<td>({T})</td>
</tr>
<tr>
<td>([t_0, 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_0, t_3])</td>
<td>(A = {\text{no_tr, tr, 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_tr, tr, 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_tr, tr, 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_tr, tr, Error}})</td>
<td>(t_6)</td>
<td>({q_2, q_3})</td>
<td>({T, X})</td>
</tr>
</tbody>
</table>

(DC-5): Delays

\[
S_i(q) > 0 \implies [-q] \vdash [q \land A] \xrightarrow{\leq S_i(q)} [q \lor \delta(q, A \setminus S_e(q))] 
\]  

DC-5

<table>
<thead>
<tr>
<th>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

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

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

- 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

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

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

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

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

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

\[
\implies [\neg q] ; [q \land A]^\varepsilon \rightarrow [\neg q] \quad (DC-10)
\]

- Due to (DC-8):
  - \( t_5 - t_4 < 2\varepsilon \)

- Due to (DC-9):
  - \( t_3 - t_2 < 2\varepsilon \)

- Due to (DC-10):
  - \( t_1 - t_0 < \varepsilon \)
\( \Box([q] \implies [\omega(q)]) \)

\( [q_0 \land A] \longrightarrow \land \; [q_0 \lor \delta(q_0, A)] \)  \hspace{1cm} (DC-2')

\( S_t(q_0) > 0 \implies [q_0 \land A] \xrightarrow{\leq S_t(q_0)} [q_0 \lor \delta(q_0, A \setminus S_e(q_0))] \)  \hspace{1cm} (DC-4')

\( S_t(q_0) > 0 \implies [q_0] ; [q_0 \land A]^{\varepsilon} \xrightarrow{\leq S_t(q_0)} [q_0 \lor \delta(q_0, A \setminus S_e(q_0))] \)  \hspace{1cm} (DC-5')

\( S_t(q_0) = 0 \land q_0 \notin \delta(q_0, A) \implies [q_0 \land A]^{\varepsilon} \longrightarrow_0 [\neg q_0] \)  \hspace{1cm} (DC-7')

\( S_t(q_0) > 0 \land A \cap S_e(q_0) = \emptyset \land q_0 \notin \delta(q_0, A) \implies [q_0 \land A]^{\varepsilon} \longrightarrow_0 [\neg q_0] \)  \hspace{1cm} (DC-10')
Definition 5.3.
The Duration Calculus semantics of a PLC Automaton \( A \) is
\[
[A]_{DC} := \bigwedge_{q \in Q, \emptyset \neq A \subseteq \Sigma} DC_{-1} \land \cdots \land DC_{-11} \land DC_{-2}' \land DC_{-4}' \land DC_{-5}' \land DC_{-7}' \land DC_{-10}'.
\]

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 \( In_A, St_A, \) and \( Out_A \).
- 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

\[ [St_A \in Q \land In_A = emergency\_signal] \xrightarrow{0.1} [St_A = motor\_off] \]
Given a PLC-Automaton, one often wants to know whether it guarantees properties of the form

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

("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
  \[ [\text{St}_A \in \Pi \land \text{ln}_A \in A] \xrightarrow{c^n} [\text{St}_A \in \Pi_{\text{target}}], \]
  abbreviated as
  \[ [\Pi \land A] \xrightarrow{c^n} [\Pi_{\text{target}}]. \]

Reaction Time Theorem Premises

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

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

• $\Pi = \{N, T\}, \ A = \{\text{no}_\text{tr}\}$
  • $\delta(\Pi, A) = \{N\} \subseteq \Pi$

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

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

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

- \( \Pi = \{T\}, \ A = \{\text{no\_tr}\} \)
  - \( \delta(\Pi, A) = \{N\} \not\subseteq \Pi \)
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
\[ [\Pi \land A] \xrightarrow{c} \left[ \delta(\Pi, A) \right] \]
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?
(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?

- $\varepsilon$
- $2\varepsilon$
- $3\varepsilon$
- $5\ s$
- $5\ s + \varepsilon$
- $5\ s + 2\varepsilon$
- $5\ s + 3\varepsilon$
- ...

\[\{N, T\} \land \{\text{no_tr}\} \xrightarrow{5+3\varepsilon} [N]\]
(1) If we are in state $N$ or $T$, how long does $N$ or $T$ need to persist together with input $\text{no\_tr}$, to ensure that we observe $N$ again?

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

- Because: earlier we have shown

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

- Thus Theorem 5.6 yields

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

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

- **Because:** earlier we have shown

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

- **Thus Theorem 5.6 yields**

$$\lceil \{N, T\} \land \{\text{no}_\text{tr}\} \rceil \xrightarrow{c} \lceil N \rceil$$

with

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

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

$$= \varepsilon + 5 + 2\varepsilon = 5 + 3\varepsilon$$

---

**Reaction Time Theorem: Example 2**

(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?
(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?

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

Because: earlier we have shown

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

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

- **Because:** earlier we have shown

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

- Thus Theorem 5.6 yields

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

with

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

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

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

\[
\left\{ N, T \right\} \land \{ \text{no} \_ \text{tr}, \text{tr} \} \xrightarrow{c} [N, T]
\]
(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?

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

• Because: earlier we have shown

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

• Thus Theorem 5.6 yields

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

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

- **Because**: earlier we have shown

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

- Thus Theorem 5.6 yields

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

with

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

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

\[
= \varepsilon
\]

---

**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:

**Proposition 5.4.**

\[
\Pi \subseteq \Pi' \subseteq Q \text{ and } A \subseteq A' \subseteq \Sigma \text{ implies } \delta(\Pi, A) \subseteq \delta(\Pi', A').
\]
Examples:
- $\Pi = \{N, T\}, A = \{\text{no\_tr}\}$
- $\delta^0(\Pi, A) = \{N, T\}$
Examples:

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

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

---

[Diagram of contraction examples]
Examples:

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

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

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

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

- $\Pi = \{T\}$, $A = \{\text{no\_tr}\}$
  - $\delta(\Pi, A) = \{N\} \not\subseteq \Pi$
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 \( n \in \mathbb{N}_0 \),
\[
[\Pi \land A] \xrightarrow{c_n} [\delta^n(\Pi, A)]_{=\Pi_{\text{target}}}
\]
where
\[
c_n := \varepsilon + \max(\{0\} \cup \left\{ \sum_{i=1}^{k} s(\pi_i, A) \mid 1 \leq k \leq n \land \exists \pi_1, \ldots, \pi_k \in \Pi \setminus \delta^n(\Pi, A) \land \forall j \in \{1, \ldots, k-1\} : \pi_{j+1} \in \delta(\pi_j, A) \right\})
\]
and \( s(\pi, A) \) as before.

Proof Idea of Reaction Time Theorem

(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

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

• This is equivalent to not having

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

• Which is equivalent to having

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

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

• Which is equivalent to having

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

• 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
Programmable Logic Controllers (PLC) are epitomically for real-time controller platforms:
- have real-time clock device, 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

Recent Results:
- Timed Sequence Diagrams, or Quasi-equal Clocks,
  or Automatic Code Generation, or …