Correctness Proof

Gas Burner Controller: The Complete Specification

PLCA Semantics

Example

What's New

Lecture 9: DC Implementables II

Real-Time Systems
Thus $\varepsilon \leq 1 - 0.1$. Hence $\varepsilon \leq \| \rho \| = 1 - 0.1$. Thus $\varepsilon$. Let $\mathcal{L} = \mathcal{L} \mathcal{L} = \mathcal{L} \mathcal{L} = \mathcal{L} \mathcal{L}$. Hence $\mathcal{L} \mathcal{L} = \mathcal{L} \mathcal{L} = \mathcal{L} \mathcal{L}$. We can conclude $\mathcal{L} = \mathcal{L} \mathcal{L}$. Let $\mathcal{L} = \mathcal{L} \mathcal{L} = \mathcal{L} \mathcal{L}$. Hence $\mathcal{L} = \mathcal{L} \mathcal{L} = \mathcal{L} \mathcal{L}$.
Thus \( 0 \leq \int = 30 = \ell \Rightarrow \int = 1 \). By 3.15 and Case (iii), we can conclude

\[
\begin{align*}
\int &\leq 30 \\
\Rightarrow \int &\leq \ell
\end{align*}
\]

Thus \( \int \) is sufficient for

\[
\begin{align*}
\int &\leq 30 = \ell \\
\Rightarrow \int &\leq \ell
\end{align*}
\]

Recall

\[
\begin{align*}
\int &\leq 30 = \ell \\
\Rightarrow \int &\leq \ell
\end{align*}
\]

Thus \( \int \) is sufficient for
How do PLCs look? 

Now Where's the Implementation? 

The Plan 

• Correctness Proof for the Gas Burner Implementables 
• Now where's the implementation? 
• Programmable Logic Controllers (PLC) 
• How do they look like? 
• What's special about them? 
• The read/compute/write cycle of PLC 
• Example: Stutter Filter 
• Structured Text example 
• Other IEC 61131-3 programming languages 
• PLC Automata 
• Example: Stutter Filter 
• PLCA Semantics by example 
• Cycle time 

What is a PLC?
What's special about PLC?

- Microprocessor, memory, timers
- Digital (or analog) I/O ports
- Possibly RS 232, fieldbuses, networking
- Robust hardware
- Reprogrammable
- Standardised programming model (IEC 61131-3)

Where are PLC employed?

- Mostly process automation
- Production lines
- Packaging lines
- Chemical plants
- Power plants
- Electric motors, pneumatic or hydraulic cylinders
- ... not so much:
- Product automation, there tailored or OTS controller boards
- Embedded controllers
- ... 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.

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.

Example: Stutter Filter

- stat : INT := 0 ; (*0 : = N, 1 : = T, 2 : = X*)
- tmr : TP ;
- ENDVAR
- IF stat = 0 THEN
- %output : = N ;
- IF %input = tr THEN
- stat : = 1 ;
- %output : = T ;
- ELSEIF %input = Error THEN
- stat : = 2 ;
- %output : = X ;
- ENDIF
- ELSEIF stat = 1 THEN
- tmr (IN : = TRUE , PT : = t #5.0 s ) ;
- IF (%input = no_tr AND NOT tmr . Q ) THEN
- stat : = 0 ;
- %output : = N ;
- tmr (IN : = FALSE , PT : = t #0.0 s ) ;
- ELSEIF %input = Error THEN
- stat : = 2 ;
- %output : = X ;
- tmr (IN : = FALSE , PT : = t #0.0 s ) ;
- ENDIF
- ENDIF
How are PLCs programmed, practically?

The read/compute/write cycle of PLCs is crucial. Here's an example of a program for a stutter filter:

```plaintext
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 = true THEN
    state := 1 ;
    output := T ;
  ELSEIF input = Error THEN
    state := 2 ;
    output := X ;
  ENDIF
ELSEIF state = 1 THEN
  timer (IN := TRUE , PT := t #5.0 s ) ;
  IF (input = no_tr AND NOT timer .Q ) THEN
    state := 0 ;
    output := N ;
    timer (IN := FALSE , PT := t #0.0 s ) ;
  ELSEIF input = Error THEN
    state := 2 ;
    output := X ;
  ENDIF
ENDIF
```

Intuitive semantics:
- do the assignment
- if assignment changed

Alternative Programming Languages by IEC 61131-3

LD: xOR yST z
z := x OR y

Instruction List Structured Text

q
y
q q
x
z
q

( )

Relay) Ladder Diagram Function Block Diagram

Figure 2.2: Implementations of the operation "x ⊕ y"

Lukoschus (2004)

Tied together by
- Sequential Function Charts (SFC)
Unfortunate: deviations in semantics...

Bauer (2003)

s
✛ step (initial)
✛ transition
g
1
✛ transition condition (guard)
s
1
S
action1
N
action2
✛ action block
❳❳❳❳ ②
action name
PPPPPPP ✐
action qualifier

g
2
s
2
R
action1

g
3
❄

Figure 2.3: Elements of sequential function charts

Lukoschus (2004)