Real-Time Systems

Lecture 9: DC Implementables II

2017-11-28

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

- **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**

Recall: Specification of a Gas Burner Controller

Gas Burner Controller: The Complete Specification

$C : \{ \text{idle}, \text{purge}, \text{ignite}, \text{burn} \}$

Controller: (**local**)

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

$$\lceil \text{idle} \rceil \longrightarrow \lceil \text{idle} \vee \text{purge} \rceil \qquad \text{(Seq-1)}$$

$$\lceil \text{purge} \rceil \longrightarrow \lceil \text{purge} \vee \text{ignite} \rceil \qquad \text{(Seq-2)}$$

$$\lceil \text{ignite} \rceil \longrightarrow \lceil \text{ignite} \vee \text{burn} \rceil \qquad \text{(Seq-3)}$$

$$\lceil \text{burn} \rceil \longrightarrow \lceil \text{burn} \vee \text{idle} \rceil \qquad \text{(Seq-4)}$$

$$\lceil \text{purge} \rceil \xrightarrow{30+\varepsilon} \lceil \neg\text{purge} \rceil \qquad \text{(Prog-1)}$$

$$\lceil \text{ignite} \rceil \xrightarrow{0.5+\varepsilon} \lceil \neg\text{ignite} \rceil \qquad \text{(Prog-2)}$$

$$\lceil \neg\text{purge} \rceil \,;\, \lceil \text{purge} \rceil \xrightarrow{\leq 30} \lceil \text{purge} \rceil \qquad \text{(Stab-2)}$$

$$\lceil \neg\text{ignite} \rceil \,;\, \lceil \text{ignite} \rceil \xrightarrow{\leq 0.5} \lceil \text{ignite} \rceil \qquad \text{(Stab-3)}$$

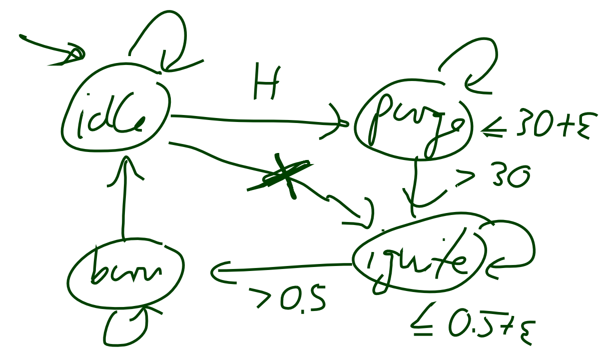
$$\lceil \text{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg\text{idle} \rceil \qquad \text{(Syn-1)}$$

$$\lceil \text{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg\text{burn} \rceil \qquad \text{(Syn-2)}$$

$$\lceil \neg\text{idle} \rceil \,;\, \lceil \text{idle} \wedge \neg H \rceil \longrightarrow \lceil \text{idle} \rceil \qquad \text{(Stab-1)}$$

$$\lceil \text{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \text{idle} \rceil \qquad \text{(Stab-1-init)}$$

$$\lceil \neg\text{burn} \rceil \,;\, \lceil \text{burn} \wedge H \wedge F \rceil \longrightarrow \lceil \text{burn} \rceil \qquad \text{(Stab-4)}$$



Gas Burner Controller: The Complete Specification

Controller: (**local**)

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

$$\lceil \text{idle} \rceil \longrightarrow \lceil \text{idle} \vee \text{purge} \rceil \qquad \text{(Seq-1)}$$

$$\lceil \text{purge} \rceil \longrightarrow \lceil \text{purge} \vee \text{ignite} \rceil \qquad \text{(Seq-2)}$$

$$\lceil \text{ignite} \rceil \longrightarrow \lceil \text{ignite} \vee \text{burn} \rceil \qquad \text{(Seq-3)}$$

$$\lceil \text{burn} \rceil \longrightarrow \lceil \text{burn} \vee \text{idle} \rceil \qquad \text{(Seq-4)}$$

$$\lceil \text{purge} \rceil \xrightarrow{30+\varepsilon} \lceil \neg\text{purge} \rceil \qquad \text{(Prog-1)}$$

$$\lceil \text{ignite} \rceil \xrightarrow{0.5+\varepsilon} \lceil \neg\text{ignite} \rceil \qquad \text{(Prog-2)}$$

$$\lceil \neg\text{purge} \rceil \,;\, \lceil \text{purge} \rceil \xrightarrow{\leq 30} \lceil \text{purge} \rceil \qquad \text{(Stab-2)}$$

$$\lceil \neg\text{ignite} \rceil \,;\, \lceil \text{ignite} \rceil \xrightarrow{\leq 0.5} \lceil \text{ignite} \rceil \qquad \text{(Stab-3)}$$

$$\lceil \text{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg\text{idle} \rceil \qquad \text{(Syn-1)}$$

$$\lceil \text{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg\text{burn} \rceil \qquad \text{(Syn-2)}$$

$$\lceil \neg\text{idle} \rceil \,;\, \lceil \text{idle} \wedge \neg H \rceil \longrightarrow \lceil \text{idle} \rceil \qquad \text{(Stab-1)}$$

$$\lceil \text{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \text{idle} \rceil \qquad \text{(Stab-1-init)}$$

$$\lceil \neg\text{burn} \rceil \,;\, \lceil \text{burn} \wedge H \wedge F \rceil \longrightarrow \lceil \text{burn} \rceil \qquad \text{(Stab-4)}$$

Gas Valve: (**output**)

$$\lceil\rceil \vee \lceil \neg G \rceil \,;\, true \qquad \text{(Init-4)}$$

$$\lceil G \wedge (\text{idle} \vee \text{purge}) \rceil \xrightarrow{\varepsilon} \lceil \neg G \rceil \qquad \text{(Syn-3)}$$

$$\lceil \neg G \wedge (\text{ignite} \vee \text{burn}) \rceil \xrightarrow{\varepsilon} \lceil G \rceil \qquad \text{(Syn-4)}$$

$$\lceil G \rceil \,;\, \lceil \neg G \wedge (\text{idle} \vee \text{purge}) \rceil \longrightarrow \lceil \neg G \rceil \qquad \text{(Stab-6)}$$

$$\lceil \neg G \wedge (\text{idle} \vee \text{purge}) \rceil \longrightarrow_0 \lceil \neg G \rceil \qquad \text{(Stab-6-init)}$$

$$\lceil \neg G \rceil \,;\, \lceil G \wedge (\text{ignite} \vee \text{burn}) \rceil \longrightarrow \lceil G \rceil \qquad \text{(Stab-7)}$$

Heating Request: (**input**)

$$\lceil\rceil \vee \lceil \neg H \rceil \,;\, true, \qquad \text{(Init-2)}$$

Flame: (**input**)

$$\lceil\rceil \vee \lceil \neg F \rceil \,;\, true, \qquad \text{(Init-3)}$$

$$\lceil F \rceil \,;\, \lceil \neg F \wedge \neg\text{ignite} \rceil \longrightarrow \lceil \neg F \rceil \qquad \text{(Stab-5)}$$

$$\lceil \neg F \wedge \neg\text{ignite} \rceil \longrightarrow_0 \lceil \neg F \rceil \qquad \text{(Stab-5-init)}$$

Implementable Gas Burner Controller: Correctness Proof

Gas Burner Controller Correctness Proof

Set GB-Ctrl $:=$ Init-1 $\wedge \cdots \wedge$ Stab-7 $\wedge\, \varepsilon > 0$.

In the following, we show

$$\models \text{GB-Ctrl} \wedge A(\varepsilon) \implies \text{Req-1}.$$

where $A(\varepsilon)$ constrains the **reaction time** of computers executing the control program.

Read: **if** a program behaving like 'GB-Ctrl' is executed on a computer with reaction time ε such that $A(\varepsilon)$ holds, then 'Req' is **satisfied** in the system.

Recall:

$$\text{Req} :\iff \Box(\ell \geq 60 \implies 20 \cdot \textstyle\int L \leq \ell)$$

and (cf. Olderog and Dierks (2008))

$$\models \text{Req-1} \implies \text{Req}$$

for the **simplified** requirement

$$\text{Req-1} := \Box(\ell \leq 30 \implies \textstyle\int L \leq 1).$$

Lemma 3.15

$$\models \text{GB-Ctrl} \implies \Box \left(\begin{array}{rrcl} & (\lceil \text{idle} \rceil & \implies & \int G \leq \varepsilon) \\ \wedge & (\lceil \text{purge} \rceil & \implies & \int G \leq \varepsilon) \\ \wedge & (\lceil \text{ignite} \rceil & \implies & \ell \leq 0.5 + \varepsilon) \\ \wedge & (\lceil \text{burn} \rceil & \implies & \int \neg F \leq 2\varepsilon) \end{array} \right)$$

Proof: Let $\mathcal{I}$ be an interpretation, $\mathcal{V}$ a valuation, and $[c, d]$ an interval with $\mathcal{I}, \mathcal{V}, [c, d] \models \text{GB-Ctrl}$.

Let $[b, e] \subseteq [c, d]$.

- **Case 1**: $\mathcal{I}, \mathcal{V}, [b, e] \models \lceil \text{idle} \rceil$
  From

  $$\lceil G \wedge (\text{idle} \vee \text{purge}) \rceil \xrightarrow{\varepsilon} \lceil \neg G \rceil \qquad \text{(Syn-3)}$$

  $$\lceil G \rceil \,;\, \lceil \neg G \wedge (\text{idle} \vee \text{purge}) \rceil \longrightarrow \lceil \neg G \rceil \qquad \text{(Stab-6)}$$

  we can conclude

  $$\mathcal{I}, \mathcal{V}, [b, e] \models \underbrace{\Box(\lceil G \rceil \implies \ell \leq \varepsilon)}_{} \wedge \underbrace{\neg \Diamond(\lceil G \rceil \,;\, \lceil \neg G \rceil \,;\, \lceil G \rceil)}_{}$$

  by (Syn-3), the valve is closed within ε time units when in 'idle'

  by (Stab-6), the valve doesn't open again when in 'idle'

  Thus $\mathcal{I}, \mathcal{V}, [b, e] \models \int G \leq \varepsilon$.

- **Case 2**: $\mathcal{I}, \mathcal{V}, [b, e] \models \lceil \text{purge} \rceil$ Analogously to case 1.

Lemma 3.15 Cont'd

$$\text{GB-Ctrl} \implies \Box \left(\begin{array}{rl} (\lceil \text{idle} \rceil \implies & \int G \leq \varepsilon) \\ \wedge\, (\lceil \text{purge} \rceil \implies & \int G \leq \varepsilon) \\ \wedge\, (\lceil \text{ignite} \rceil \implies & \ell \leq 0.5 + \varepsilon) \\ \wedge\, (\lceil \text{burn} \rceil \implies & \int \neg F \leq 2\varepsilon) \end{array} \right)$$

- **Case 3**: $\mathcal{I}, \mathcal{V}, [b,e] \models \lceil \text{ignite} \rceil$
  From
  $$\lceil \text{ignite} \rceil \xrightarrow{0.5+\varepsilon} \lceil \neg \text{ignite} \rceil \qquad \text{(Prog-2)}$$
  we can directly conclude $\mathcal{I}, \mathcal{V}, [b,e] \models \ell \leq 0.5 + \varepsilon$.
- **Case 4**: $\mathcal{I}, \mathcal{V}, [b,e] \models \lceil \text{burn} \rceil$
  From

$$\begin{array}{l} \lceil\rceil \;\vee\; \lceil \neg F \rceil \;\vee\; \lceil F \rceil \\ \vee\; \lceil F \rceil ; \lceil \neg F \rceil \;\vee\; \lceil \neg F \rceil ; \lceil F \rceil \\ \vee\; \lceil \neg F \rceil ; \lceil F \rceil ; \lceil \neg F \rceil \end{array}$$

$$\lceil \text{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg \text{burn} \rceil \qquad \text{(Syn-2)}$$

$$\lceil F \rceil ; \lceil \neg F \wedge \neg \text{ignite} \rceil \longrightarrow \lceil \neg F \rceil \qquad \text{(Stab-5)}$$

we can conclude

$$\mathcal{I}, \mathcal{V}, [b,e] \models \underbrace{\Box(\lceil \neg F \rceil \implies \ell \leq \varepsilon)}_{\text{by (Syn-2)}} \wedge \underbrace{\neg \Diamond(\lceil F \rceil ; \lceil \neg F \rceil ; \lceil F \rceil)}_{\text{by (Stab-5)}}$$

Thus $\mathcal{I}, \mathcal{V}, [b,e] \models \int \neg F \leq 2\varepsilon$.

Lemma 3.16

$$\models \exists \varepsilon \bullet \text{GB-Ctrl} \implies \underbrace{\Box(\ell \le 30 \implies \textstyle\int L \le 1)}_{\text{Req-1}}$$

Proof: Let $\mathcal{I}, \mathcal{V}$, and $[b,e]$ such that $\mathcal{I}, \mathcal{V}, [b,e] \models \text{GB-Ctrl} \wedge \ell \le 30$.

Distinguish 5 cases:

(i) $\mathcal{I}, \mathcal{V}, [b,e] \models \lceil\rceil$ ✓

(ii) $\mathcal{I}, \mathcal{V}, [b,e] \models (\lceil \text{idle} \rceil \,;\, true \wedge \ell \le 30)$

(iii) $\mathcal{I}, \mathcal{V}, [b,e] \models (\lceil \text{purge} \rceil \,;\, true \wedge \ell \le 30)$

(iv) $\mathcal{I}, \mathcal{V}, [b,e] \models (\lceil \text{ignite} \rceil \,;\, true \wedge \ell \le 30)$

(v) $\mathcal{I}, \mathcal{V}, [b,e] \models (\lceil \text{burn} \rceil \,;\, true \wedge \ell \le 30)$

Lemma 3.16 Cont'd

$$\text{3.15: GB-Ctrl} \implies \Box \left(\begin{array}{rcr} (\lceil \mathsf{idle} \rceil & \implies & \int G \leq \varepsilon) \\ \wedge\ (\lceil \mathsf{purge} \rceil & \implies & \int G \leq \varepsilon) \\ \wedge\ (\lceil \mathsf{ignite} \rceil & \implies & \ell \leq 0.5 + \varepsilon) \\ \wedge\ (\lceil \mathsf{burn} \rceil & \implies & \int \neg F \leq 2\varepsilon) \end{array} \right)$$

- **Case (i)**: $\mathcal{I}, \mathcal{V}, [b, e] \models \lceil\rceil$
- **Case (ii)**: $\mathcal{I}, \mathcal{V}, [b, e] \models \lceil \mathsf{idle} \rceil \,;\, \mathit{true} \wedge \ell \leq 30$
  From

$$\lceil \mathsf{idle} \rceil \longrightarrow \lceil \mathsf{idle} \vee \mathsf{purge} \rceil \qquad \text{(Seq-1)}$$

$$\lceil \neg\mathsf{purge} \rceil \,;\, \lceil \mathsf{purge} \rceil \xrightarrow{\leq 30} \lceil \mathsf{purge} \rceil \qquad \text{(Stab-2)}$$

we can conclude

$$\mathcal{I}, \mathcal{V}, [b, e] \models \lceil \mathsf{idle} \rceil \vee \lceil \mathsf{idle} \rceil \,;\, \lceil \mathsf{purge} \rceil$$

By 3.15,

$$\mathcal{I}, \mathcal{V}, [b, e] \models \left(\int L \leq \varepsilon\right) \vee \left(\int L \leq \varepsilon \,;\, \int L \leq \varepsilon\right)$$

hence

$$\mathcal{I}, \mathcal{V}, [b, e] \models \int L \leq 2\varepsilon$$

Thus $\boxed{\varepsilon \leq 0.5}$ is sufficient for Req-1 ($\int L \leq 1$) **in this case**.

Lemma 3.16 Cont'd

$$\text{3.15: GB-Ctrl} \implies \Box \begin{pmatrix} & (\lceil \mathsf{idle} \rceil \implies & \textstyle\int G \leq \varepsilon) \\ \wedge & (\lceil \mathsf{purge} \rceil \implies & \textstyle\int G \leq \varepsilon) \\ \wedge & (\lceil \mathsf{ignite} \rceil \implies & \ell \leq 0.5 + \varepsilon) \\ \wedge & (\lceil \mathsf{burn} \rceil \implies & \textstyle\int \neg F \leq 2\varepsilon) \end{pmatrix}$$

- **Case (iii)**: $\mathcal{I}, \mathcal{V}, [b, e] \models \lceil \mathsf{burn} \rceil \, ; \, \mathit{true} \wedge \ell \leq 30$

  From

  $$\lceil \mathsf{burn} \rceil \longrightarrow \lceil \mathsf{burn} \vee \mathsf{idle} \rceil \qquad \text{(Seq-4)}$$

  we can conclude

  $$\mathcal{I}, \mathcal{V}, [b, e] \models (\lceil \mathsf{burn} \rceil \vee \lceil \mathsf{burn} \rceil \, ; \underbrace{\lceil \mathsf{idle} \rceil \, ; \, \mathit{true}}_{\text{Case (ii)}}) \wedge \ell \leq 30.$$

  By 3.15 and Case (ii),

  $$\mathcal{I}, \mathcal{V}, [b, e] \models ((\textstyle\int L \leq 2\varepsilon) \vee (\textstyle\int L \leq 2\varepsilon) \, ; \, (\textstyle\int L \leq 2\varepsilon)) \wedge \ell \leq 30.$$

  hence

  $$\mathcal{I}, \mathcal{V}, [b, e] \models \textstyle\int L \leq 4\varepsilon.$$

  Thus $\boxed{\varepsilon \leq 0.25}$ is sufficient for Req-1 ($\int L \leq 1$) **in this case**.

Lemma 3.16 Cont'd

$$\text{3.15: GB-Ctrl} \implies \Box \left(\begin{array}{rl} (\lceil \mathsf{idle} \rceil \implies & \int G \leq \varepsilon) \\ \wedge (\lceil \mathsf{purge} \rceil \implies & \int G \leq \varepsilon) \\ \wedge (\lceil \mathsf{ignite} \rceil \implies & \ell \leq 0.5 + \varepsilon) \\ \wedge (\lceil \mathsf{burn} \rceil \implies & \int \neg F \leq 2\varepsilon) \end{array} \right)$$

- **Case (iv)**: $\mathcal{I}, \mathcal{V}, [b,e] \models \lceil \mathsf{ignite} \rceil \,;\, true \wedge \ell \leq 30$
  From

$$\lceil \mathsf{ignite} \rceil \longrightarrow \lceil \mathsf{ignite} \vee \mathsf{burn} \rceil \qquad \text{(Seq-3)}$$

we can conclude

$$\mathcal{I}, \mathcal{V}, [b,e] \models (\lceil \mathsf{ignite} \rceil \vee \lceil \mathsf{ignite} \rceil \,;\, \underbrace{\lceil \mathsf{burn} \rceil \,;\, true)}_{\text{Case (iii)}} \wedge \ell \leq 30.$$

By 3.15 and Case (iii),

$$\mathcal{I}, \mathcal{V}, [b,e] \models ((\int L \leq 0.5 + \varepsilon) \vee (\int L \leq 0.5 + \varepsilon) \,;\, (\int L \leq 4\varepsilon)) \wedge \ell \leq 30$$

hence

$$\mathcal{I}, \mathcal{V}, [b,e] \models \int L \leq 0.5 + 5\varepsilon.$$

Thus $\boxed{\varepsilon \leq 0.1}$ is sufficient for Req-1 ($\int L \leq 1$) **in this case**.

Lemma 3.16 Cont'd

$$\text{3.15: GB-Ctrl} \implies \Box \left(\begin{array}{rl} (\lceil \text{idle} \rceil \implies & \int G \leq \varepsilon) \\ \wedge\, (\lceil \text{purge} \rceil \implies & \int G \leq \varepsilon) \\ \wedge\, (\lceil \text{ignite} \rceil \implies & \ell \leq 0.5 + \varepsilon) \\ \wedge\, (\lceil \text{burn} \rceil \implies & \int \neg F \leq 2\varepsilon) \end{array} \right)$$

- **Case (v)**: $\mathcal{I}, \mathcal{V}, [b, e] \models \lceil \text{purge} \rceil \,;\, true \wedge \ell \leq 30$
  From

$$\lceil \text{purge} \rceil \longrightarrow \lceil \text{purge} \vee \text{ignite} \rceil \qquad \text{(Seq-2)}$$

and 3.15 and Case (iv) we can conclude

$$\mathcal{I}, \mathcal{V}, [b, e] \models \int L \leq 0.5 + 6\varepsilon.$$

Thus $\boxed{\varepsilon \leq \frac{1}{12}}$ is sufficient for Req-1 ($\int L \leq 1$) in this case. $\Box$

Lemma 3.16.

$$\models \exists\, \varepsilon \bullet \text{GB-Ctrl} \implies \underbrace{\Box(\ell \leq 30 \implies \int L \leq 1)}_{\text{Req-1}}$$

Correctness Result

Theorem 3.17.

$$\models \left(\text{GB-Ctrl} \wedge \varepsilon \leq \frac{1}{12} \right) \implies \text{Req}$$

Recall:

- Req-1 $= \Box(\ell \leq 30 \implies \int L \leq 1)$ implies Req.
- 3.15: $\lceil \text{purge} \rceil \implies \int L \leq \varepsilon$, $\lceil \text{ignite} \rceil \implies \int L \leq 0.5 + \varepsilon$, $\lceil \text{burn} \rceil \implies \int L \leq 2\varepsilon$, $\lceil \text{idle} \rceil \implies \int L \leq \varepsilon$.

purge	ignite	burn	idle	purge
$— \ell \geq 30 \longrightarrow$	$\longleftarrow \ell \geq 0.5 \longrightarrow$			$\longleftarrow \ell \geq 30 —$
$\int L \leq \varepsilon$	$\int L \leq 0.5 + \varepsilon$	$\int L \leq 2\varepsilon$	$\int L \leq \varepsilon$	$\int L \leq \varepsilon$

$—— \ell \leq 30 ——$

- Thus $\int L \leq 0.5 + 6\varepsilon$, so a sufficient reaction time constraint is $A(\varepsilon) := \varepsilon \leq \frac{1}{12}$.

Discussion

- We used only

$$\text{‘Seq-1’},\quad \text{‘Seq-2’},\quad \text{‘Seq-3’},\quad \text{‘Seq-4’},$$
$$\text{‘Prog-2’},\quad \text{‘Syn-2’},\quad \text{‘Syn-3’},$$
$$\text{‘Stab-2’},\quad \text{‘Stab-5’},\quad \text{‘Stab-6’}.$$

What about

$$\text{Prog-1} = \lceil \text{purge} \rceil \xrightarrow{30+\varepsilon} \lceil \neg\text{purge} \rceil$$

for instance?

Gas Burner Controller: The Complete Specification

Controller: (**local**)

$\lceil\rceil \vee \lceil \text{idle} \rceil \,;\, true,$ (Init-1)

$\lceil \text{idle} \rceil \longrightarrow \lceil \text{idle} \vee \text{purge} \rceil$ (Seq-1)

$\lceil \text{purge} \rceil \longrightarrow \lceil \text{purge} \vee \text{ignite} \rceil$ (Seq-2)

$\lceil \text{ignite} \rceil \longrightarrow \lceil \text{ignite} \vee \text{burn} \rceil$ (Seq-3)

$\lceil \text{burn} \rceil \longrightarrow \lceil \text{burn} \vee \text{idle} \rceil$ (Seq-4)

$\lceil \text{purge} \rceil \xrightarrow{30+\varepsilon} \lceil \neg\text{purge} \rceil$ (Prog-1)

$\lceil \text{ignite} \rceil \xrightarrow{0.5+\varepsilon} \lceil \neg\text{ignite} \rceil$ (Prog-2)

$\lceil \neg\text{purge} \rceil \,;\, \lceil \text{purge} \rceil \xrightarrow{\leq 30} \lceil \text{purge} \rceil$ (Stab-2)

$\lceil \neg\text{ignite} \rceil \,;\, \lceil \text{ignite} \rceil \xrightarrow{\leq 0.5} \lceil \text{ignite} \rceil$ (Stab-3)

$\lceil \text{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg\text{idle} \rceil$ (Syn-1)

$\lceil \text{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg\text{burn} \rceil$ (Syn-2)

$\lceil \neg\text{idle} \rceil \,;\, \lceil \text{idle} \wedge \neg H \rceil \longrightarrow \lceil \text{idle} \rceil$ (Stab-1)

$\lceil \text{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \text{idle} \rceil$ (Stab-1-init)

$\lceil \neg\text{burn} \rceil \,;\, \lceil \text{burn} \wedge H \wedge F \rceil \longrightarrow \lceil \text{burn} \rceil$ (Stab-4)

Gas Valve: (**output**)

$\lceil\rceil \vee \lceil \neg G \rceil \,;\, true$ (Init-4)

$\lceil G \wedge (\text{idle} \vee \text{purge}) \rceil \xrightarrow{\varepsilon} \lceil \neg G \rceil$ (Syn-3)

$\lceil \neg G \wedge (\text{ignite} \vee \text{burn}) \rceil \xrightarrow{\varepsilon} \lceil G \rceil$ (Syn-4)

$\lceil G \rceil \,;\, \lceil \neg G \wedge (\text{idle} \vee \text{purge}) \rceil \longrightarrow \lceil \neg G \rceil$ (Stab-6)

$\lceil \neg G \wedge (\text{idle} \vee \text{purge}) \rceil \longrightarrow_0 \lceil \neg G \rceil$ (Stab-6-init)

$\lceil \neg G \rceil \,;\, \lceil G \wedge (\text{ignite} \vee \text{burn}) \rceil \longrightarrow \lceil G \rceil$ (Stab-7)

Heating Request: (**input**)

$\lceil\rceil \vee \lceil \neg H \rceil \,;\, true,$ (Init-2)

Flame: (**input**)

$\lceil\rceil \vee \lceil \neg F \rceil \,;\, true,$ (Init-3)

$\lceil F \rceil \,;\, \lceil \neg F \wedge \neg\text{ignite} \rceil \longrightarrow \lceil \neg F \rceil$ (Stab-5)

$\lceil \neg F \wedge \neg\text{ignite} \rceil \longrightarrow_0 \lceil \neg F \rceil$ (Stab-5-init)

Discussion

- We used only

'Seq-1', 'Seq-2', 'Seq-3', 'Seq-4',
'Prog-2', 'Syn-2', 'Syn-3',
'Stab-2', 'Stab-5', 'Stab-6'.

What about

$$\text{Prog-1} = \lceil \text{purge} \rceil \xrightarrow{30+\varepsilon} \lceil \neg\text{purge} \rceil$$

for instance?

We only proved the **safety** property on leakage,
we did not consider the (not formalised) **liveness** requirement:

the controller **should do something** finally,
e.g. heating requests should be served finally by trying an ignition.

Content

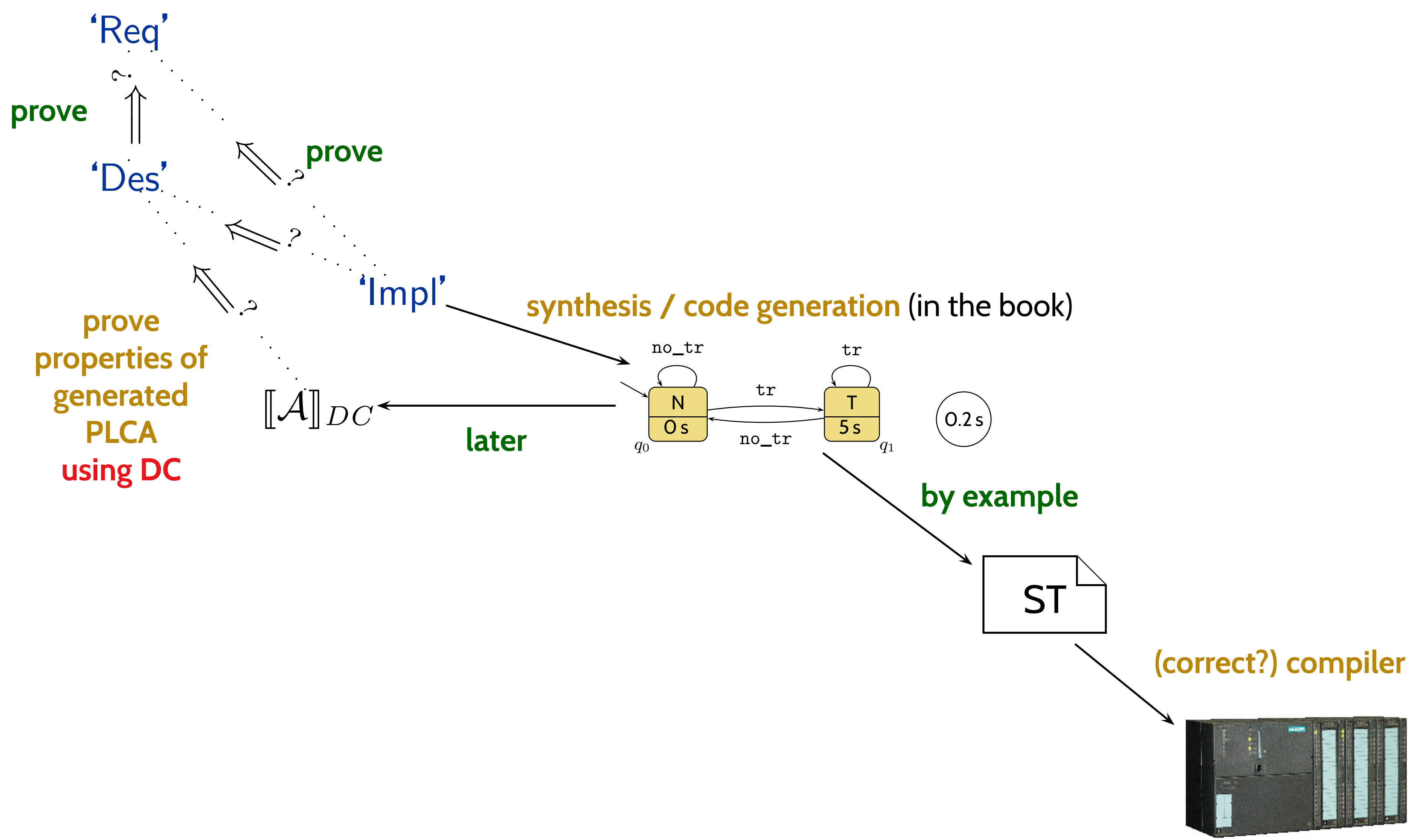
- **Correctness Proof**
  for the Gas Burner Implementables

- **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**

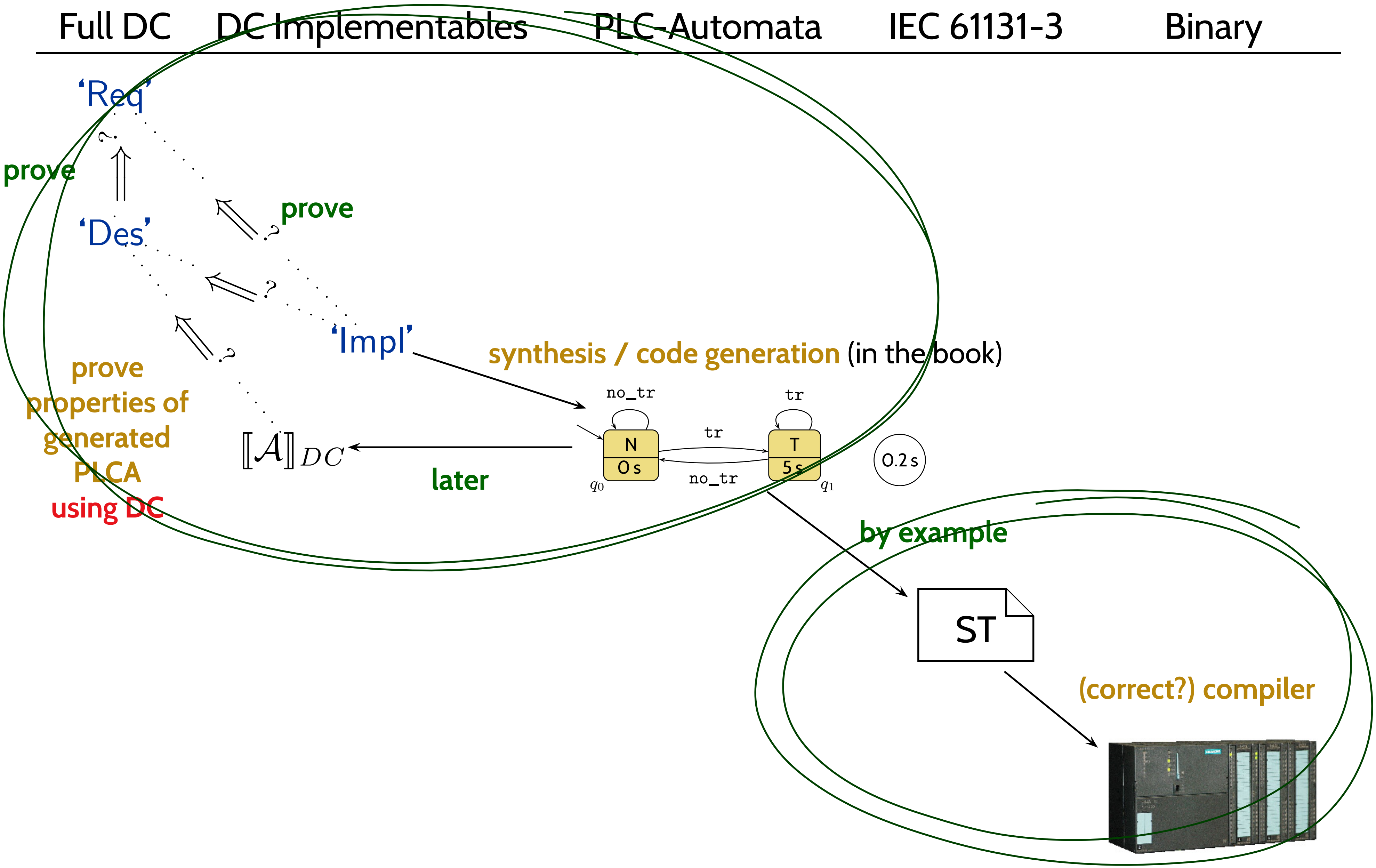
Now Where's the Implementation?

The Plan

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



The Plan


Full DC
DC Implementables
PLC-Automata
IEC 61131-3
Binary
'Req'
prove
'Des'
prove
'Impl'
synthesis / code generation (in the book)
prove properties of generated PLCA using DC
$[\![\mathcal{A}]\!]_{DC}$
later
no_tr
tr
tr
no_tr
N
0 s
T
5 s
q_0
q_1
0.2 s
by example
ST
(correct?) compiler

Content

- **Correctness Proof**
  for the Gas Burner Implementables

- **Now where's the implementation?**

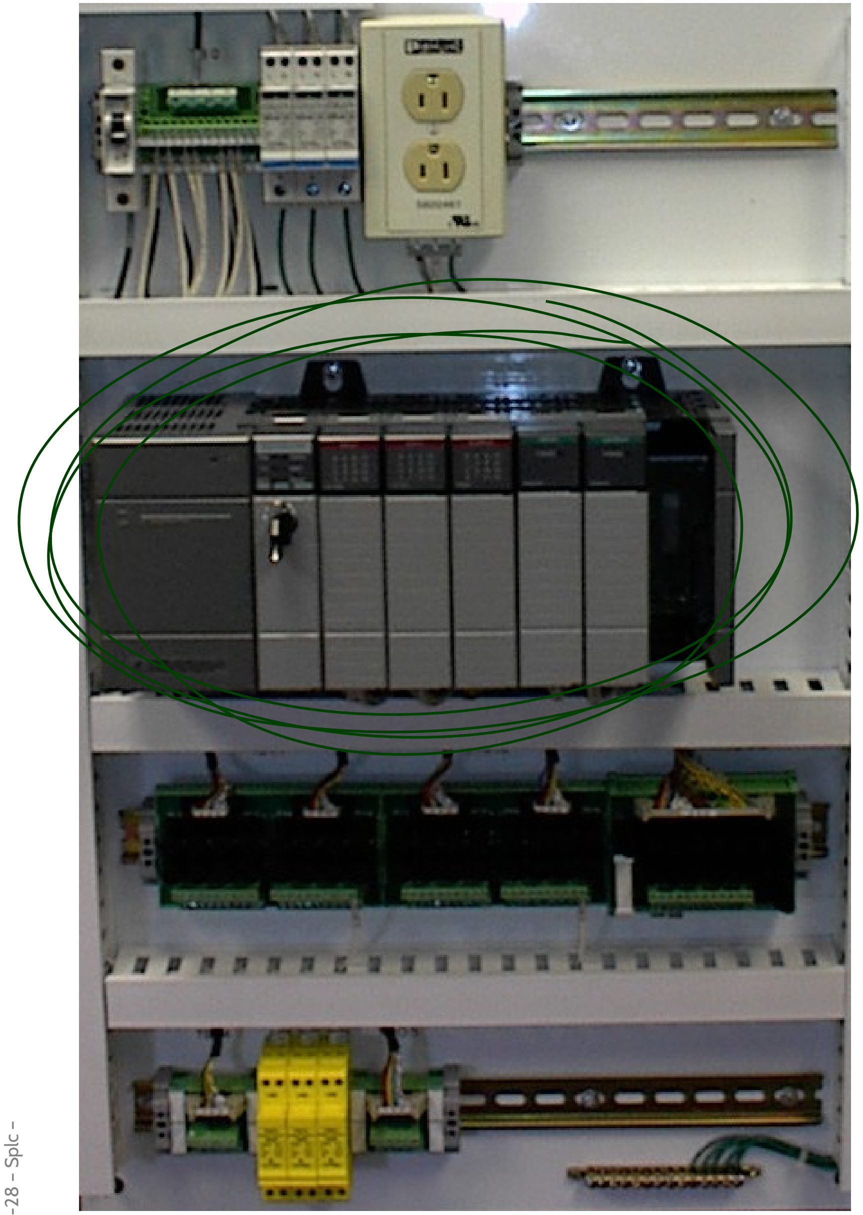
- **Programmable Logic Controllers** (PLC)
  - How do they **look like**?
  - What's **special** about them?
  - The **read/compute/write** cycle of PLC

- **Example**: Stutter Filter
  - **Structured Text** example
  - Other IEC 61131-3 programming languages

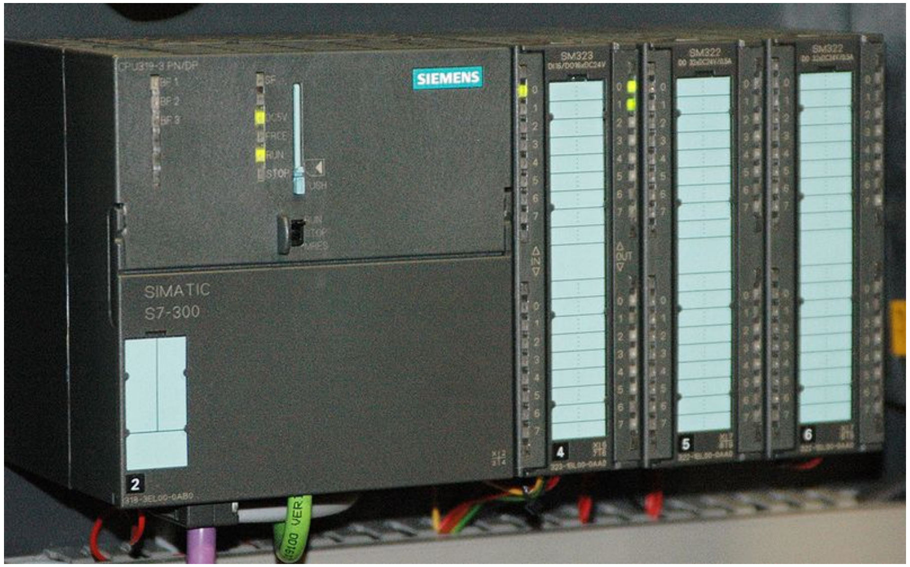
- **PLC Automata**
  - **Example**: Stutter Filter
  - **PLCA Semantics** by example
  - **Cycle time**

What is a PLC?

How do PLC look like?

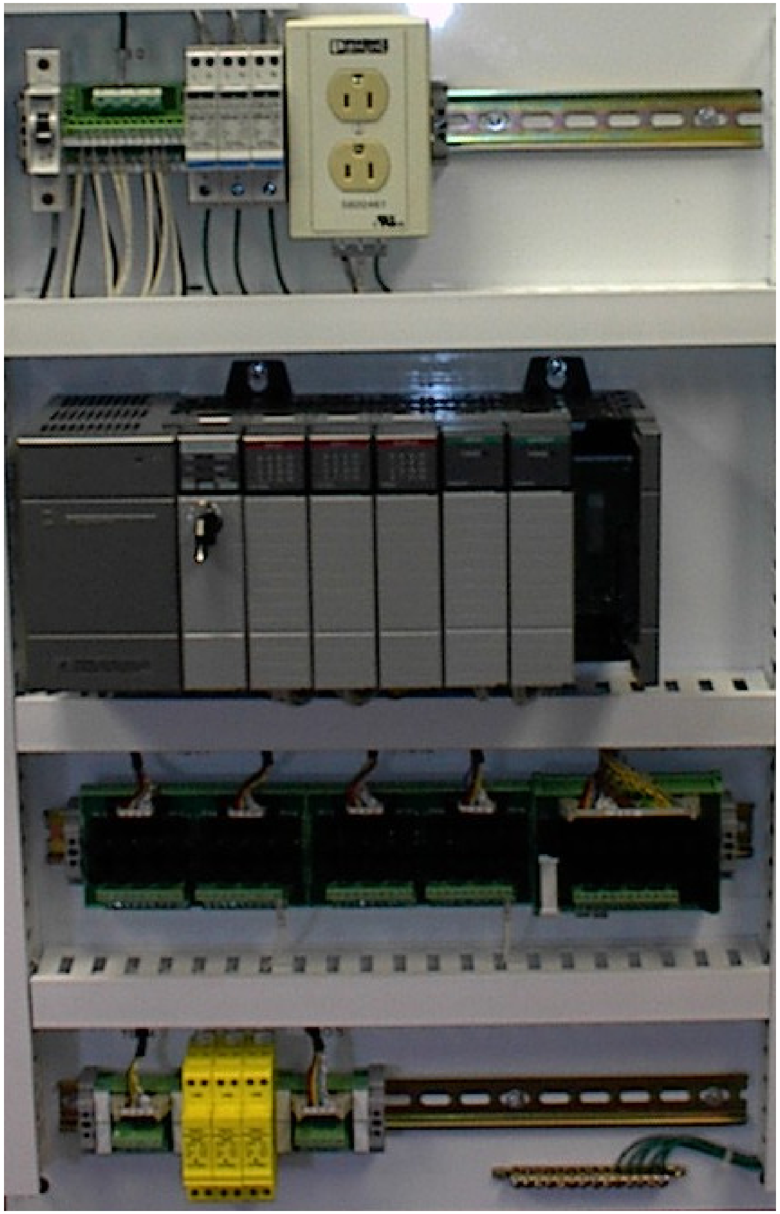


http://wikimedia.org (public domain)



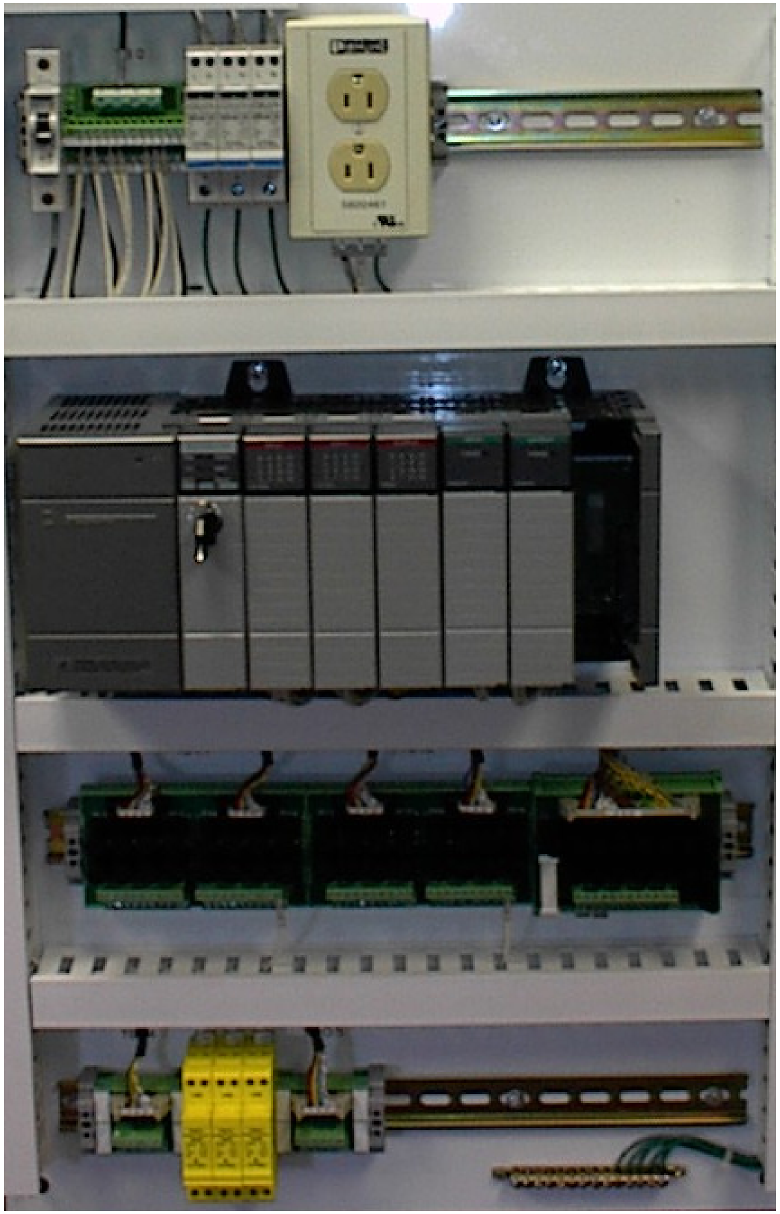
http://wikimedia.org (CC nc-sa 2.5, Ulli1105)

What's special about PLC?



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

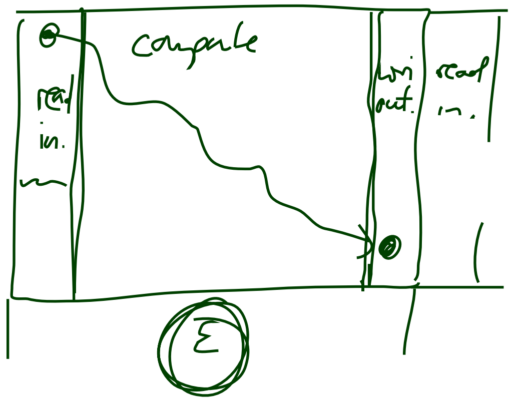
Where are PLC employed?

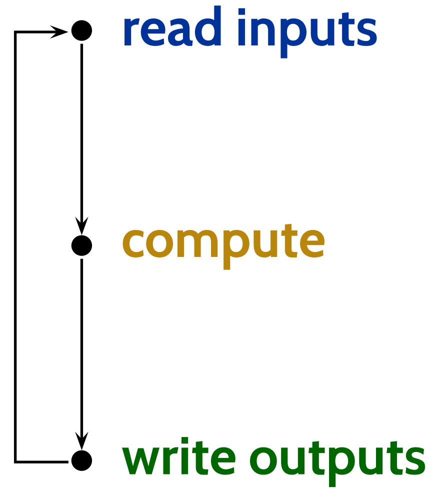


- mostly **process automatisation**
  - production lines
  - packaging lines
  - chemical plants
  - power plants
  - electric motors, pneumatic or hydraulic cylinders
  - …
- not so much: **product automatisation**, there
  - tailored or OTS controller boards
  - embedded controllers
  - …

How are PLC programmed?

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



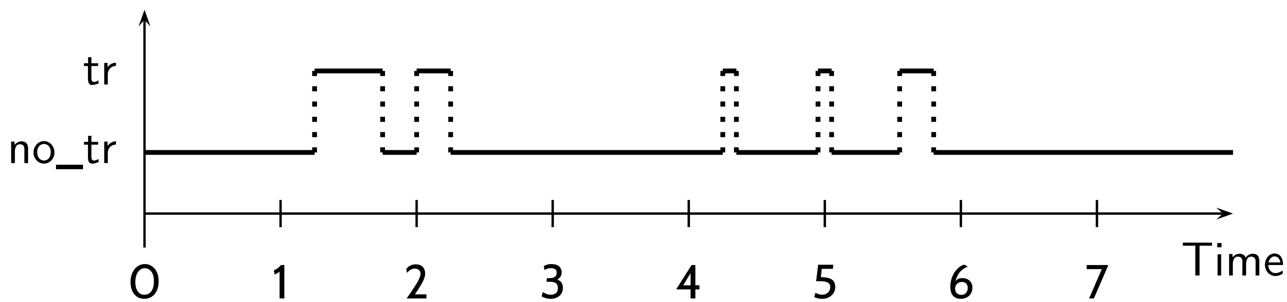


- Cyclic operation is repeated until external interruption (such as shutdown or reset).
- Cycle time: typically a few milliseconds (Lukoschus, 2004).

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

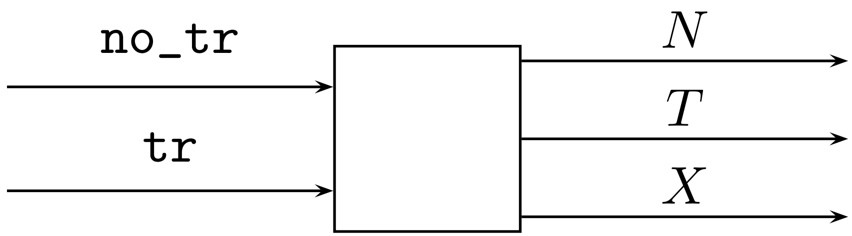
How are PLC programmed, practically?

- **Example:** reliable, stutter-free train sensor.
  - Assume a track-side sensor which outputs:
    - `no_tr` – iff "no passing train"
    - `tr` – iff "a train is passing"
  - Assume that a change from "`no_tr`" to "`tr`" signals arrival of a train. (No spurious sensor values.)
  - **Problem:** the sensor may **stutter**, i.e. oscillate between "`no_tr`" and "`tr`" multiple times.

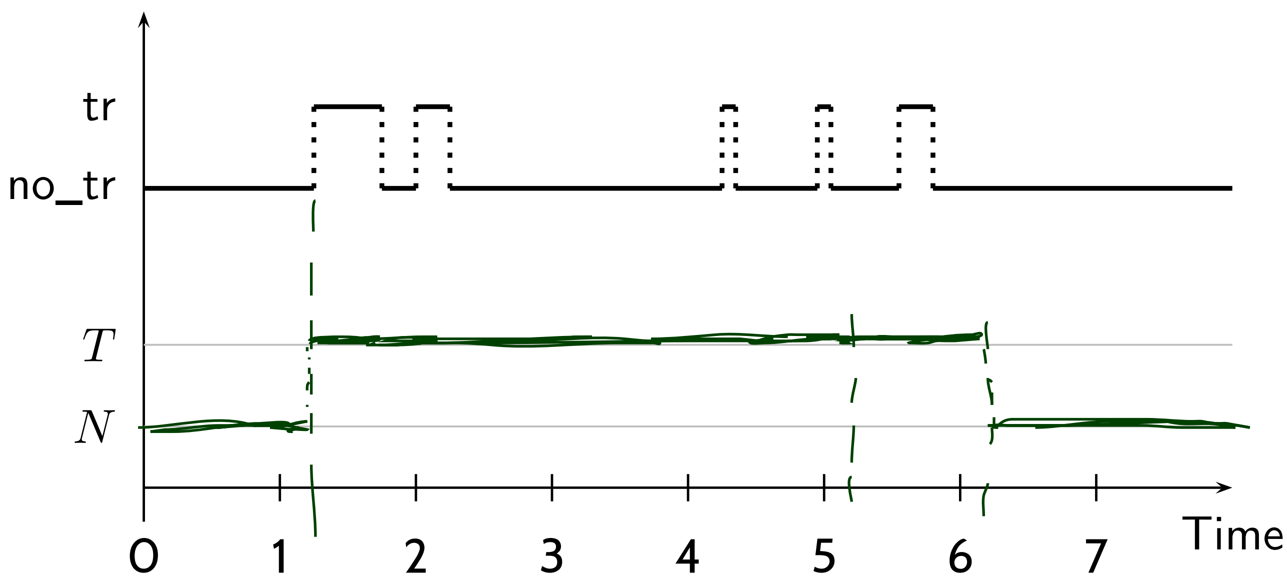


Example: Stutter Filter

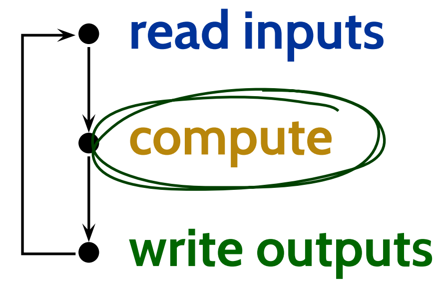
- **Idea:** a stutter **filter** with outputs N and T, for "no train" and "train passing" (and possibly X, for error).



After arrival of a train, it should ignore "`no_tr`" for 5 seconds.



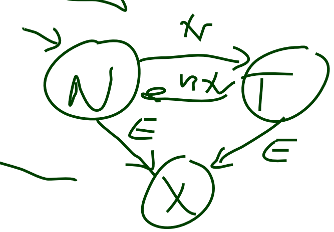
How are PLC programmed, practically?



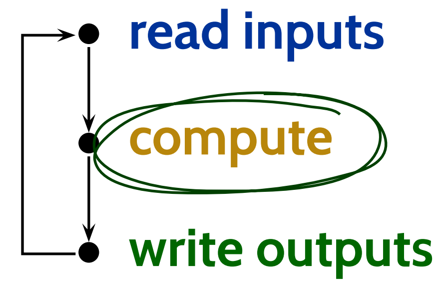

```
PROGRAM PLC_PRG_FILTER
VAR
  state : INT := 0; (* 0:=N, 1:=T, 2:=X *)
  tmr   : TP;
ENDVAR

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

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



How are PLC programmed, practically?




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

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

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

declare timer `tmr`

duration

intuitive semantics:

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

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

Alternative Programming Languages by IEC 61131-3

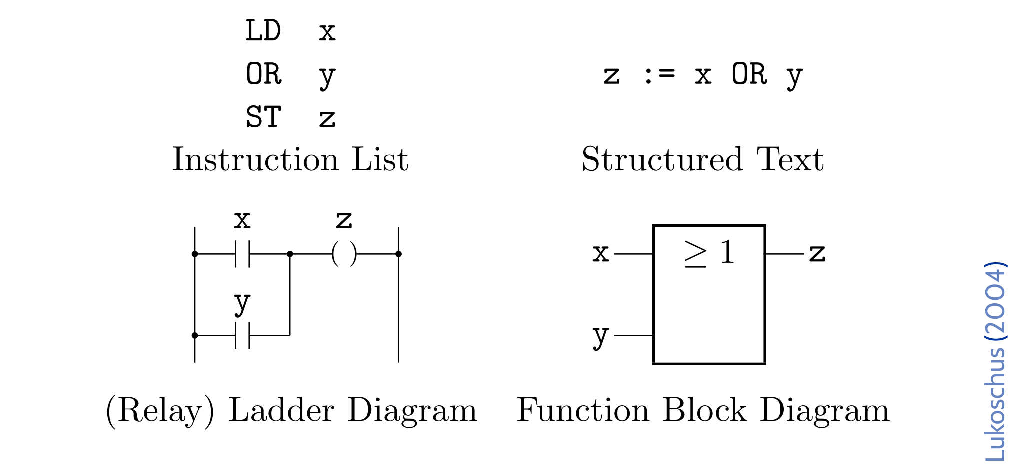


Figure 2.2: Implementations of the operation “x becomes $y \vee z$”

Tied together by

- Sequential Function Charts (SFC)

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

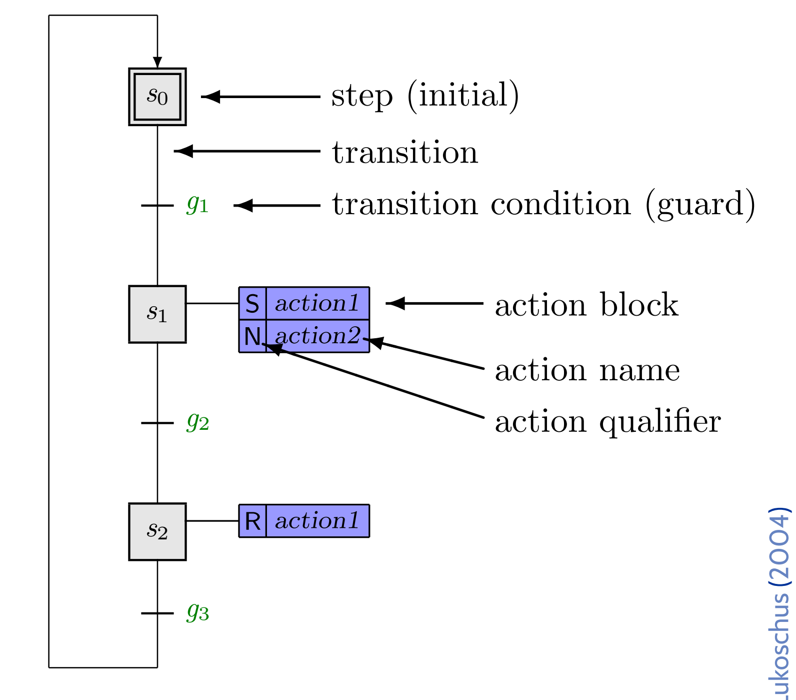


Figure 2.3: Elements of sequential function charts

Content

- **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**

Tell Them What You've Told Them...

- We can **prove** the Gas Burner implementables **correct** by carefully considering its phases.
- A **crucial aspect** is **reaction time**:
  - Controller programs executed on some hardware platform do not react in 0-**time**,
  - some platforms may be **too slow** to satisfy requirements.
- **Programmable Logic Controllers** (PLC) are epitomic for real-time controller platforms:
  - have a **real-time clock** device,
  - can **read inputs** and **write outputs**,
  - can manage **local state**.
- **PLC programs**
  - are executed in **read/compute/write** cycles,
  - have a **cycle-time** (possibly a watchdog).
- **PLC Automata** are a more abstract (than IEC 61131-3) way of describing and studying PLC programs.

References

References

Bauer, N. (2003). *Formale Analyse von Sequential Function Charts*. PhD thesis, Universität Dortmund.

Lukoschus, B. (2004). *Compositional Verification of Industrial Control Systems*. PhD thesis, Christian-Albrechts-Universität zu Kiel.

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