Real-Time Systems

Lecture 8: DC Implementables I

2017-11-23

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Content

Introduction

- **Observables and Evolutions** ✓
- **Duration Calculus** (DC) ✓
- Semantical Correctness Proofs ✓
- DC Decidability ✓
- DC Implementables 8–9
- **PLC-Automata** 10

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

- **Timed Automata** (TA), Uppaal 11–
- Networks of Timed Automata
- Region/Zone-Abstraction
- TA model-checking
- Extended Timed Automata
- Undecidability Results

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

- **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 ...

Content

- **Motivation**: Why DC Implementables?
  - What can we assume of controller platforms?
- **DC Standard Forms**
  - **Followed-by**, **Followed-by-initially**
  - **(Timed) Leads-to**
  - **(Timed) Up-to**, **(Timed) Up-to-initially**
- **Control Automata**
  - **phases**, **basic phases**
- **DC Implementables**
  - **Initialisation**, **Sequencing**, **Progress**
  - **Synchronisation**, **(Un)Bounded Stability**
  - **(Un)Bounded Initial Stability**
- **Example**:
  A correct controller for the **Gas Burner**
  specified by **DC Implementables**

DC Implementables: Motivation

Requirements vs. Implementations

- **Problem:** in general, a DC requirement doesn't tell **how** to achieve it, how to build a controller/write a program which ensures it.

$$\square((\lceil \neg B \rceil \wedge \ell = 5 \,;\, \lceil B \rceil) \implies (\lceil L = \text{yellow} \rceil \,;\, \mathit{true}))$$

"whenever a pedestrian presses the button **5 time units from now**, then **now** the traffic lights should **already be yellow**"

Plus: road traffic should not see 'yellow' all the time.

$$\square((\lceil B \wedge L = \text{green} \rceil \,;\, \ell = 5) \implies (\mathit{true} \,;\, \lceil L = \text{red} \rceil))$$

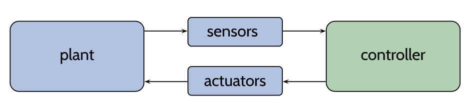
"whenever a pedestrian presses the button **now** while road traffic sees 'green', then **5 time units later** (the latest) road traffic **should see 'red'**"

Requirements vs. Implementations

- **Problem:** in general, a DC requirement doesn't tell **how** to achieve it, how to build a controller/write a program which ensures it.

- What **a controller** (clearly) **can do** is:
  - consider **inputs now**,
  - **change (local) state**, or
  - **wait**,
  - set **outputs now**.

  (But not, e.g., consider future inputs now.)



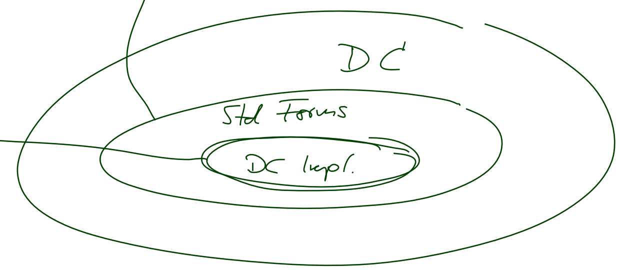
- So, if we have
  - a DC requirement 'Req',
  - a description 'Impl' in DC of the controller behaviour, which "uses" **just these four** operations,

  then
  - proving correctness (still) amounts to proving $\models_0$ Impl $\implies$ Req (**in DC**)
  - and we (more or less) **know how to program** (the correct) 'Impl' in a PLC language, or in C on a real-time OS, or or or…

Approach: Control Automata and DC Implementables

Plan:

- Introduce **DC Standard Forms** (a sub-language of DC)
- Introduce **Control Automata**
- Introduce **DC Implementables** as a subset of **DC Standard Forms**
- **Example**: a correct controller design for the notorious Gas Burner



DC Standard Forms

DC Standard Forms: Followed-by

In the following: F is a DC **formula**, P a **state assertion**, θ a **rigid term**.

- **Followed-by:**

$$F \longrightarrow \lceil P \rceil :\iff \neg\Diamond(F\,;\lceil\neg P\rceil) \iff \Box\neg(F\,;\lceil\neg P\rceil)$$

in other symbols

$$\forall\, x \bullet \Box((F \wedge \ell = x)\,;\ell > 0 \implies (F \wedge \ell = x)\,;\lceil P\rceil\,;\mathit{true})$$

DC Standard Forms: Followed-by

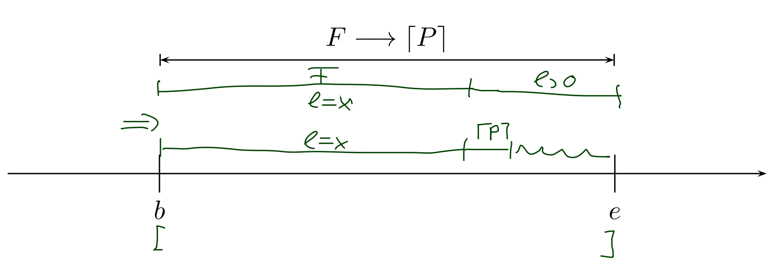
In the following: F is a DC **formula**, P a **state assertion**, θ a **rigid term**.

- **Followed-by:**

$$F \longrightarrow \lceil P \rceil :\iff \neg\Diamond(F\,;\lceil\neg P\rceil) \iff \Box\neg(F\,;\lceil\neg P\rceil)$$

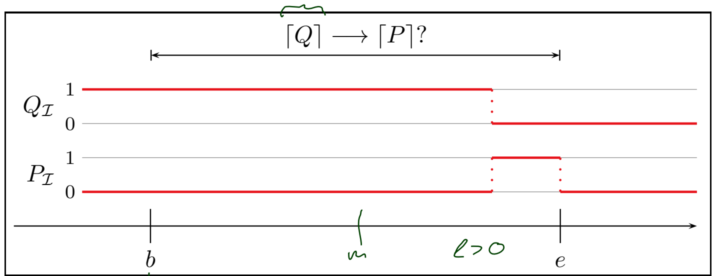
in other symbols

$$\forall\, x \bullet \Box((F \wedge \ell = x)\,;\ell > 0 \implies (F \wedge \ell = x)\,;\lceil P\rceil\,;\mathit{true})$$



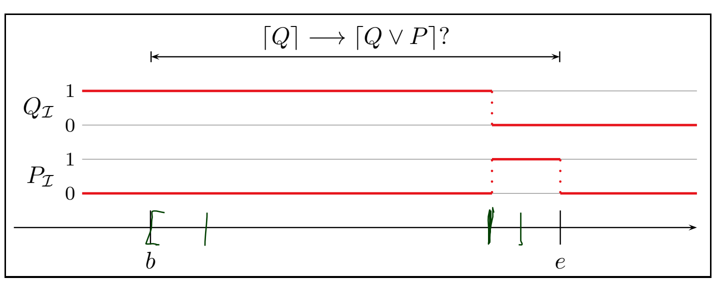
DC Standard Forms: Followed-by Examples

$$\forall x \bullet \Box((F \land \ell = x) \,;\, \ell > 0 \implies (F \land \ell = x) \,;\, \lceil P \rceil \,;\, true)$$



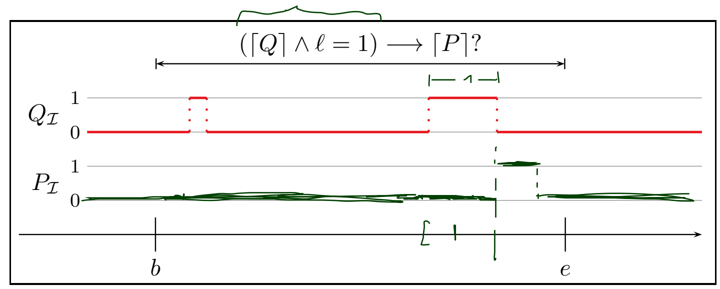
DC Standard Forms: Followed-by Examples

$$\forall x \bullet \Box((F \land \ell = x) \,;\, \ell > 0 \implies (F \land \ell = x) \,;\, \lceil P \rceil \,;\, true)$$



DC Standard Forms: Followed-by Examples

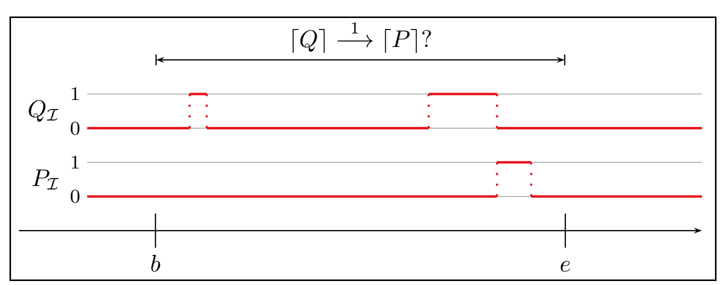
$$\forall x \bullet \Box((F \wedge \ell = x) \,;\, \ell > 0 \implies (F \wedge \ell = x) \,;\, \lceil P \rceil \,;\, \mathit{true})$$



DC Standard Forms: (Timed) leads-to

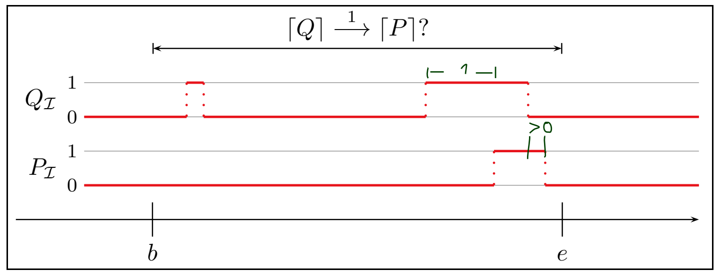
- **(Timed) leads-to:**

$$F \xrightarrow{\theta} \lceil P \rceil :\iff (F \wedge \ell = \theta) \longrightarrow \lceil P \rceil$$



- **(Timed) leads-to:**

$$F \xrightarrow{\theta} \lceil P \rceil :\iff (F \wedge \ell = \theta) \longrightarrow \lceil P \rceil$$

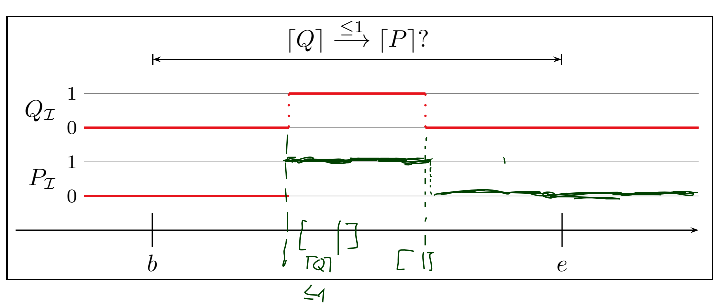


"if F persists for (at least) θ time units from time t,
then there is $\lceil P \rceil$ after $\theta + t$"

$$\forall x \bullet \Box((F \wedge \ell = x) \, ; \ell > 0 \implies (F \wedge \ell = x) \, ; \lceil P \rceil \, ; true)$$

- **(Timed) up-to:**

$$F \xrightarrow{\leq \theta} \lceil P \rceil :\iff (F \wedge \ell \leq \theta) \longrightarrow \lceil P \rceil$$

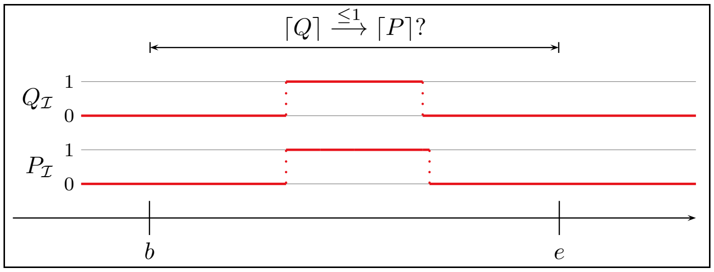


DC Standard Forms: (Timed) up-to

$$\forall x \bullet \Box((F \land \ell = x) \,;\, \ell > 0 \implies (F \land \ell = x) \,;\, \lceil P \rceil \,;\, true)$$

- **(Timed) up-to:**

$$F \xrightarrow{\leq \theta} \lceil P \rceil :\iff (F \land \ell \leq \theta) \longrightarrow \lceil P \rceil$$

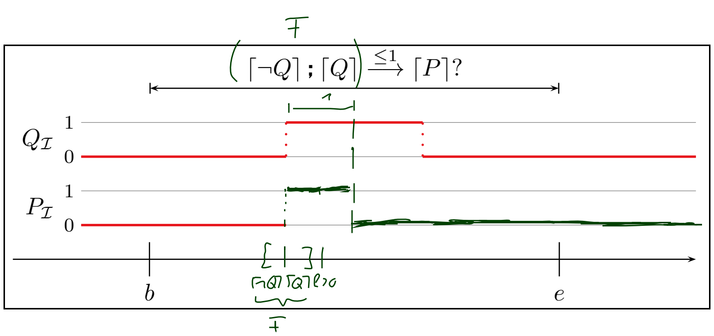


DC Standard Forms: (Timed) up-to

$$\forall x \bullet \Box((F \land \ell = x) \,;\, \ell > 0 \implies (F \land \ell = x) \,;\, \lceil P \rceil \,;\, true)$$

- **(Timed) up-to:**

$$F \xrightarrow{\leq \theta} \lceil P \rceil :\iff (F \land \ell \leq \theta) \longrightarrow \lceil P \rceil$$

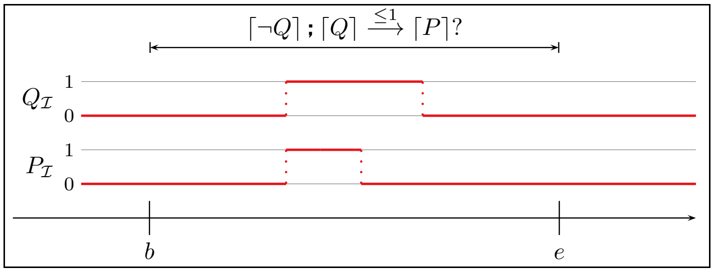


DC Standard Forms: (Timed) up-to

$$\forall x \bullet \Box((F \wedge \ell = x) \,;\, \ell > 0 \implies (F \wedge \ell = x) \,;\, \lceil P \rceil \,;\, true)$$

- **(Timed) up-to:**

$$F \xrightarrow{\leq \theta} \lceil P \rceil \;:\iff\; (F \wedge \ell \leq \theta) \longrightarrow \lceil P \rceil$$

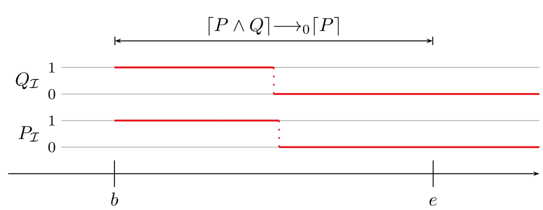


"during all Q-phases of at most θ time units, there needs to be $\lceil P \rceil$ as well"

DC Standard Forms: Initialisation

- **Followed-by-initially:**

$$F \longrightarrow_0 \lceil P \rceil \;:\iff\; \neg(F \,;\, \lceil \neg P \rceil)$$



"after an initial phase with $\lceil P \wedge Q \rceil$, $\lceil P \rceil$ persists for some non-point interval"

- **(Timed) up-to-initially:**

$$F \xrightarrow{\leq \theta}_0 \lceil P \rceil \;:\iff\; (F \wedge \ell \leq \theta) \longrightarrow_0 \lceil P \rceil$$

- **Initialisation:**

$$\lceil\rceil \vee \lceil P \rceil \,;\, true$$

Control Automata

Control Automata

- Let $X_1, \dots, X_k$ be state variables with **finite** domains $\mathcal{D}(X_1), \dots, \mathcal{D}(X_k)$.
- $X_1, \dots, X_k$ together with a DC formula 'Impl' (over $X_1, \dots, X_k$) is called **system of k control automata**.
- 'Impl' is typically a conjunction of **DC implementables**. (→ in a minute)

Example: (Simplified) **traffic lights**: $X : \{\text{red}, \text{green}, \text{yellow}\}$,

state var. $\mathcal{D}(X)$

$$\begin{aligned}\text{Impl} := &(\lceil \text{red} \rceil \longrightarrow \lceil \text{red} \vee \text{green} \rceil) \quad \wedge \quad (\lceil \text{green} \rceil \longrightarrow \lceil \text{green} \vee \text{yellow} \rceil) \\ &\wedge \quad (\lceil \text{yellow} \rceil \longrightarrow \lceil \text{yellow} \vee \text{red} \rceil) \quad \wedge \quad (\lceil\rceil \vee \lceil \text{red} \rceil \,;\, true)\end{aligned}$$

system of 1 control automaton

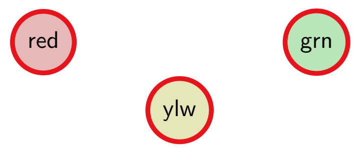
Control Automata

- Let $X_1, \dots, X_k$ be state variables with **finite** domains $\mathcal{D}(X_1), \dots, \mathcal{D}(X_k)$.
- $X_1, \dots, X_k$ together with a DC formula 'Impl' (over $X_1, \dots, X_k$) is called **system of k control automata**.
- 'Impl' is typically a conjunction of **DC implementables**. ($\rightarrow$ in a minute)

Example: (Simplified) **traffic lights**: $X : \{\mathsf{red}, \mathsf{green}, \mathsf{yellow}\}$,

$$\begin{aligned}\mathsf{Impl} := (\lceil \mathsf{red} \rceil \longrightarrow \lceil \mathsf{red} \lor \mathsf{green} \rceil) \quad &\land \quad (\lceil \mathsf{green} \rceil \longrightarrow \lceil \mathsf{green} \lor \mathsf{yellow} \rceil) \\ \land \quad (\lceil \mathsf{yellow} \rceil \longrightarrow \lceil \mathsf{yellow} \lor \mathsf{red} \rceil) \quad &\land \quad (\lceil\rceil \lor \lceil \mathsf{red} \rceil \; ; \; true)\end{aligned}$$

- Where's the **automaton**? Here, look:



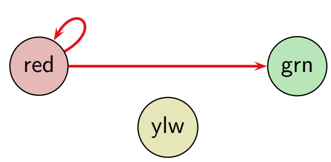
Control Automata

- Let $X_1, \dots, X_k$ be state variables with **finite** domains $\mathcal{D}(X_1), \dots, \mathcal{D}(X_k)$.
- $X_1, \dots, X_k$ together with a DC formula 'Impl' (over $X_1, \dots, X_k$) is called **system of k control automata**.
- 'Impl' is typically a conjunction of **DC implementables**. ($\rightarrow$ in a minute)

Example: (Simplified) **traffic lights**: $X : \{\mathsf{red}, \mathsf{green}, \mathsf{yellow}\}$,

$$\begin{aligned}\mathsf{Impl} := (\lceil \mathsf{red} \rceil \longrightarrow \lceil \mathsf{red} \lor \mathsf{green} \rceil) \quad &\land \quad (\lceil \mathsf{green} \rceil \longrightarrow \lceil \mathsf{green} \lor \mathsf{yellow} \rceil) \\ \land \quad (\lceil \mathsf{yellow} \rceil \longrightarrow \lceil \mathsf{yellow} \lor \mathsf{red} \rceil) \quad &\land \quad (\lceil\rceil \lor \lceil \mathsf{red} \rceil \; ; \; true)\end{aligned}$$

- Where's the **automaton**? Here, look:



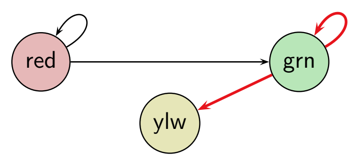
Control Automata

- Let $X_1, \dots, X_k$ be state variables with **finite** domains $\mathcal{D}(X_1), \dots, \mathcal{D}(X_k)$.
- $X_1, \dots, X_k$ together with a DC formula 'Impl' (over $X_1, \dots, X_k$) is called **system of k control automata**.
- 'Impl' is typically a conjunction of **DC implementables**. (→ in a minute)

Example: (Simplified) **traffic lights**: $X : \{\text{red}, \text{green}, \text{yellow}\}$,

$$\text{Impl} := (\lceil \text{red} \rceil \longrightarrow \lceil \text{red} \lor \text{green} \rceil) \quad \land \quad (\lceil \text{green} \rceil \longrightarrow \lceil \text{green} \lor \text{yellow} \rceil) \quad \land \quad (\lceil \text{yellow} \rceil \longrightarrow \lceil \text{yellow} \lor \text{red} \rceil) \quad \land \quad (\lceil\rceil \lor \lceil \text{red} \rceil \, ; \mathit{true})$$

- Where's the **automaton**? Here, look:



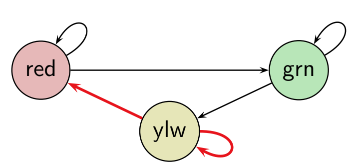
Control Automata

- Let $X_1, \dots, X_k$ be state variables with **finite** domains $\mathcal{D}(X_1), \dots, \mathcal{D}(X_k)$.
- $X_1, \dots, X_k$ together with a DC formula 'Impl' (over $X_1, \dots, X_k$) is called **system of k control automata**.
- 'Impl' is typically a conjunction of **DC implementables**. (→ in a minute)

Example: (Simplified) **traffic lights**: $X : \{\text{red}, \text{green}, \text{yellow}\}$,

$$\text{Impl} := (\lceil \text{red} \rceil \longrightarrow \lceil \text{red} \lor \text{green} \rceil) \quad \land \quad (\lceil \text{green} \rceil \longrightarrow \lceil \text{green} \lor \text{yellow} \rceil) \quad \land \quad (\lceil \text{yellow} \rceil \longrightarrow \lceil \text{yellow} \lor \text{red} \rceil) \quad \land \quad (\lceil\rceil \lor \lceil \text{red} \rceil \, ; \mathit{true})$$

- Where's the **automaton**? Here, look:



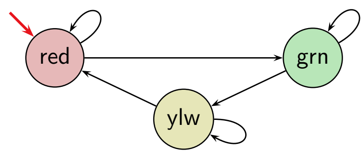
Control Automata

- Let $X_1, \ldots, X_k$ be state variables with **finite** domains $\mathcal{D}(X_1), \ldots, \mathcal{D}(X_k)$.
- $X_1, \ldots, X_k$ together with a DC formula 'Impl' (over $X_1, \ldots, X_k$) is called **system of k control automata**.
- 'Impl' is typically a conjunction of **DC implementables**. ($\rightarrow$ in a minute)

Example: (Simplified) **traffic lights**: $X : \{\mathsf{red}, \mathsf{green}, \mathsf{yellow}\}$,

$$\begin{aligned}\mathsf{Impl} := (\lceil \mathsf{red} \rceil \longrightarrow \lceil \mathsf{red} \vee \mathsf{green} \rceil) \quad &\wedge \quad (\lceil \mathsf{green} \rceil \longrightarrow \lceil \mathsf{green} \vee \mathsf{yellow} \rceil) \\ \wedge \quad (\lceil \mathsf{yellow} \rceil \longrightarrow \lceil \mathsf{yellow} \vee \mathsf{red} \rceil) \quad &\wedge \quad (\lceil\rceil \vee \lceil \mathsf{red} \rceil \,;\, true)\end{aligned}$$

- Where's the **automaton**? Here, look:



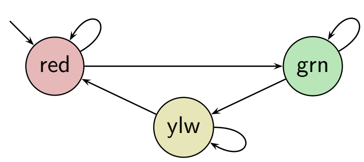
Control Automata

- Let $X_1, \ldots, X_k$ be state variables with **finite** domains $\mathcal{D}(X_1), \ldots, \mathcal{D}(X_k)$.
- $X_1, \ldots, X_k$ together with a DC formula 'Impl' (over $X_1, \ldots, X_k$) is called **system of k control automata**.
- 'Impl' is typically a conjunction of **DC implementables**. ($\rightarrow$ in a minute)

Example: (Simplified) **traffic lights**: $X : \{\mathsf{red}, \mathsf{green}, \mathsf{yellow}\}$,

$$\begin{aligned}\mathsf{Impl} := (\lceil \mathsf{red} \rceil \longrightarrow \lceil \mathsf{red} \vee \mathsf{green} \rceil) \quad &\wedge \quad (\lceil \mathsf{green} \rceil \longrightarrow \lceil \mathsf{green} \vee \mathsf{yellow} \rceil) \\ \wedge \quad (\lceil \mathsf{yellow} \rceil \longrightarrow \lceil \mathsf{yellow} \vee \mathsf{red} \rceil) \quad &\wedge \quad (\lceil\rceil \vee \lceil \mathsf{red} \rceil \,;\, true)\end{aligned}$$

- Where's the **automaton**? Here, look:



Phases

- A state assertion of the form

$$X_i = d_i, \quad d_i \in \mathcal{D}(X_i),$$

which constrains the values of X_i, is called **basic phase** of X_i.

- A **phase** of X_i is a Boolean combination of basic phases of X_i.

- **Abbreviations:**
  - Write X_i instead of $X_i = 1$, if X_i is Boolean.
  - Write d_i instead of $X_i = d_i$, if $\mathcal{D}(X_i)$ is disjoint from $\mathcal{D}(X_j)$, $i \neq j$.

- **Examples**
  - **Basic phases** of X: $(X =$ green$)$ (green) (red) (yellow)
  - **Phases** of X: $(X =$ green $\vee X =$ yellow$)$ (green $\vee$ yellow) ($\neg$red) ...
  - Not a phase: (X = green ∧ B = pressed)
    [two different observables]

DC Implementables

DC Implementables

- ...are special **patterns** of **DC Standard Forms** (due to A.P. Ravn).
- Within one pattern,
  - $\pi, \pi_1, \ldots, \pi_n, n \geq 0$, denote **phases** of **the same** state variable X_i,
  - φ denotes a state assertion **not depending** on X_i.
  - θ denotes a **rigid** term.

- **Initialisation**: $\lceil\rceil \vee \lceil\pi\rceil \,;\, true$

  "initially, the control automaton is in phase π"

- **Sequencing**: $\lceil\pi\rceil \longrightarrow \lceil\pi \vee \pi_1 \vee \cdots \vee \pi_n\rceil$

  "when the control automaton is in π, it subsequently stays in π or moves to one of $\pi_1, \ldots \pi_n$"

- **Progress**: $\lceil\pi\rceil \xrightarrow{\theta} \lceil\neg\pi\rceil$

  "after the control automaton stayed in phase π for θ time units,
  is subsequently leaves this phase, thus progresses"

DC Implementables Cont'd

- **Synchronisation**: $\lceil\pi \wedge \varphi\rceil \xrightarrow{\theta} \lceil\neg\pi\rceil$

  "after the control automaton stayed for θ time units in phase π
  with the condition φ being true, it subsequently leaves this phase"

- **Bounded Stability**:

$$\lceil\neg\pi\rceil \,;\, \lceil\pi \wedge \varphi\rceil \xrightarrow{\leq\theta} \lceil\pi \vee \pi_1 \vee \cdots \vee \pi_n\rceil$$

  "if the control automaton changed its phase to π with the condition φ being true
  and the time since this change does not exceed θ time units,
  it subsequently stays in π or moves to one of $\pi_1, \ldots, \pi_n$"

- **Unbounded Stability**:

$$\lceil\neg\pi\rceil \,;\, \lceil\pi \wedge \varphi\rceil \longrightarrow \lceil\pi \vee \pi_1 \vee \cdots \vee \pi_n\rceil$$

  "if the control automaton changed its phase to π with the condition φ being true, it
  subsequently stays in π or moves to one of $\pi_1, \ldots, \pi_n$"

DC Implementables Cont'd

- **Bounded initial stability**:

$$\lceil \pi \wedge \varphi \rceil \xrightarrow{\leq \theta}_0 \lceil \pi \vee \pi_1 \vee \cdots \vee \pi_n \rceil$$

"when the control automaton initially is in phase π with condition φ being true and the current time does not exceed θ time units, the control automaton subsequently stays in π or moves to one of $\pi_1, \ldots, \pi_n$"

- **Unbounded initial stability**:

$$\lceil \pi \wedge \varphi \rceil \longrightarrow_0 \lceil \pi \vee \pi_1 \vee \cdots \vee \pi_n \rceil$$

"when the control automaton initially is in phase π with condition φ being true, the control automaton subsequently stays in π or moves to one of $\pi_1, \ldots, \pi_n$"

– 8 – 2017-11-23 – Simpl –

Using DC Implementables for (Controller) Specifications

- Let $X_1, \ldots, X_k$ be a **system of k control automata.**
- Let 'Impl' be a conjunction of **DC implementables**.
- Then 'Impl' **specifies** / **denotes** all interpretations $\mathcal{I}$ of $X_1, \ldots, X_k$ and all valuations $\mathcal{V}$ such that $\mathcal{I}, \mathcal{V} \models_0$ Impl
- In other words: 'Impl' denotes the set $\{(\mathcal{I}, \mathcal{V}) \mid \mathcal{I}, \mathcal{V} \models_0 \text{Impl}\}$ of **interpretations** and **valuations** which **realise** 'Impl' **from** 0.
- **Controller Verification**:
  If 'Impl' describes (exactly or over-approximating) the behaviour of a controller, then proving the controller correct wrt. requirements 'Req' amounts to showing

$$\models_0 \text{Impl} \implies \text{Req}$$

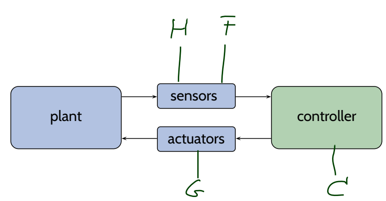
- **Controller Specification**: Dear programmers,
  'Impl' describes my design idea (and I have shown $\models_0$ Impl $\implies$ Req),
  please provide a controller program whose behaviour is a subset of 'Impl';
  that is: a correct implementation of my design.

– 8 – 2017-11-23 – Simpl –

Example: Gas Burner

– 8 – 2017-11-23 – main –

Control Automata for the Gas Burner



A **gas burner controller** can be modelled as a **system of four control automata**:

- **inputs** / sensors:
  - $H : \{0, 1\}$ – heating request
  - $F : \{0, 1\}$ – flame sensor

  implementables constraining phases of H, F express **environment assumptions**; H, F in controller implementables correspond to **reading sensor values**,

- **outputs** / actuators:
  - $G : \{0, 1\}$ – gas valve

  implementables constraining phases of G describe the connection between **controller states and actuators**.

- **local state** / controller:
  - $C : \{\text{idle}, \text{purge}, \text{ignite}, \text{burn}\}$,

  to produce the desired behaviour, the controller makes use of four **local states**.

– 8 – 2017-11-23 – Sexa –

Gas Burner Controller: Control State Changes

C : {idle, purge, ignite, burn}

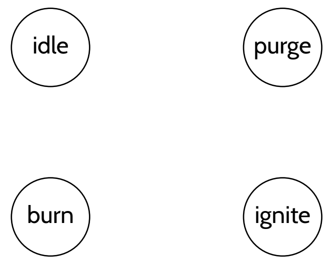
$$\lceil\rceil \vee \lceil \mathsf{idle} \rceil \,;\, \mathit{true} \qquad \text{(Init-1)}$$

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

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

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

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



Gas Burner Controller: Control State Changes

C : {idle, purge, ignite, burn}

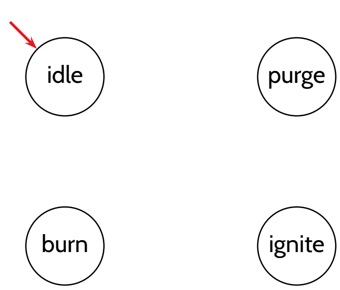
$$\lceil\rceil \vee \lceil \mathsf{idle} \rceil \,;\, \mathit{true} \qquad \text{(Init-1)}$$

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

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

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

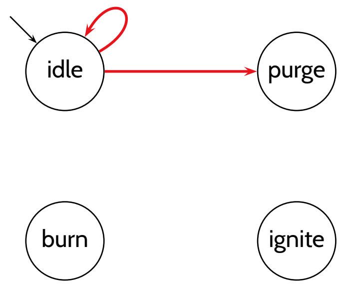
$$\lceil \mathsf{burn} \rceil \longrightarrow \lceil \mathsf{burn} \vee \mathsf{idle} \rceil \qquad \text{(Seq-4)}$$



Gas Burner Controller: Control State Changes

C : {idle, purge, ignite, burn}

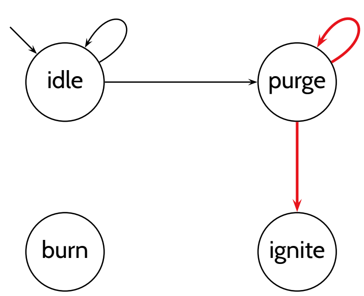
$\lceil\rceil \vee \lceil \text{idle} \rceil \,;\, \mathit{true}$	(Init-1)
$\lceil \text{idle} \rceil \longrightarrow \lceil \text{idle} \vee \text{purge} \rceil$	(Seq-1)
$\lceil \text{purge} \rceil \longrightarrow \lceil \text{purge} \vee \text{ignite} \rceil$	(Seq-2)
$\lceil \text{ignite} \rceil \longrightarrow \lceil \text{ignite} \vee \text{burn} \rceil$	(Seq-3)
$\lceil \text{burn} \rceil \longrightarrow \lceil \text{burn} \vee \text{idle} \rceil$	(Seq-4)



Gas Burner Controller: Control State Changes

C : {idle, purge, ignite, burn}

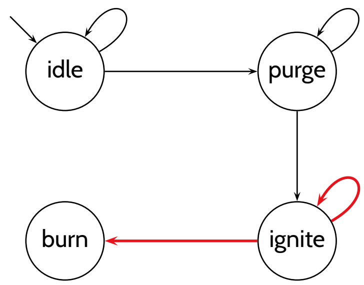
$\lceil\rceil \vee \lceil \text{idle} \rceil \,;\, \mathit{true}$	(Init-1)
$\lceil \text{idle} \rceil \longrightarrow \lceil \text{idle} \vee \text{purge} \rceil$	(Seq-1)
$\lceil \text{purge} \rceil \longrightarrow \lceil \text{purge} \vee \text{ignite} \rceil$	(Seq-2)
$\lceil \text{ignite} \rceil \longrightarrow \lceil \text{ignite} \vee \text{burn} \rceil$	(Seq-3)
$\lceil \text{burn} \rceil \longrightarrow \lceil \text{burn} \vee \text{idle} \rceil$	(Seq-4)



Gas Burner Controller: Control State Changes

C : {idle, purge, ignite, burn}

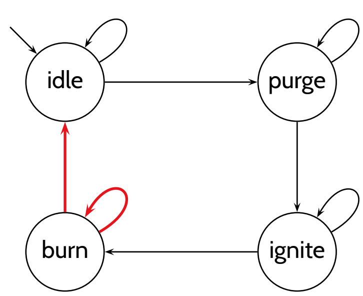
$\lceil\rceil \vee \lceil \text{idle} \rceil \, ; \, \mathit{true}$	(Init-1)
$\lceil \text{idle} \rceil \longrightarrow \lceil \text{idle} \vee \text{purge} \rceil$	(Seq-1)
$\lceil \text{purge} \rceil \longrightarrow \lceil \text{purge} \vee \text{ignite} \rceil$	(Seq-2)
$\lceil \text{ignite} \rceil \longrightarrow \lceil \text{ignite} \vee \text{burn} \rceil$	(Seq-3)
$\lceil \text{burn} \rceil \longrightarrow \lceil \text{burn} \vee \text{idle} \rceil$	(Seq-4)



Gas Burner Controller: Control State Changes

C : {idle, purge, ignite, burn}

$\lceil\rceil \vee \lceil \text{idle} \rceil \, ; \, \mathit{true}$	(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)



Gas Burner Controller: Control State Changes

C : {idle, purge, ignite, burn}

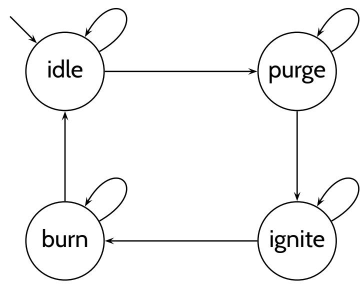
$$\lceil\rceil \vee \lceil \text{idle} \rceil \,;\, true \qquad \text{(Init-1)}$$

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

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

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

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

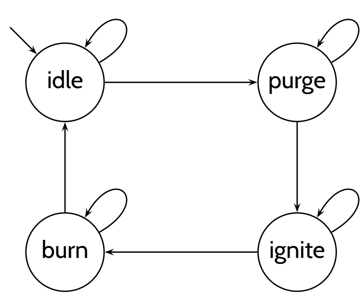


Gas Burner Controller: Timing Constraints

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

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

"after changing to 'purge', **stay there for at least** 30 time units (or: leave after 30 the earliest); you may **stay** in 'purge' **for at most** $30 + \varepsilon$ time units"

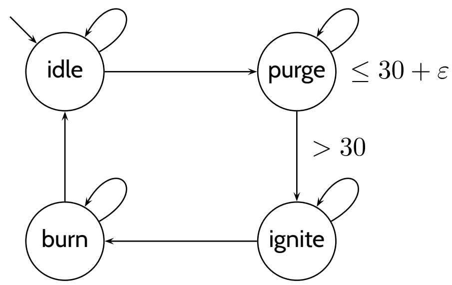


Gas Burner Controller: Timing Constraints

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

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

"after changing to 'purge', **stay there for at least** 30 time units (or: leave after 30 the earliest); you may **stay** in 'purge' **for at most** $30 + \varepsilon$ time units"



Gas Burner Controller: Timing Constraints

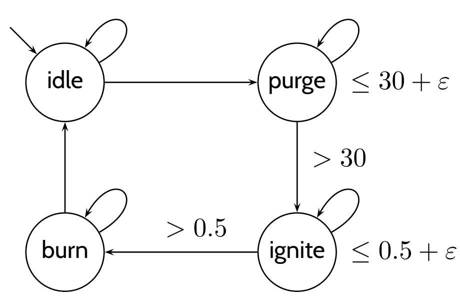
$$\lceil \neg \text{purge} \rceil \,;\, \lceil \text{purge} \rceil \xrightarrow{\le 30} \lceil \text{purge} \rceil \qquad \text{(Stab-2)}$$

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

"after changing to 'purge', **stay there for at least** 30 time units (or: leave after 30 the earliest); you may **stay** in 'purge' **for at most** $30 + \varepsilon$ time units"

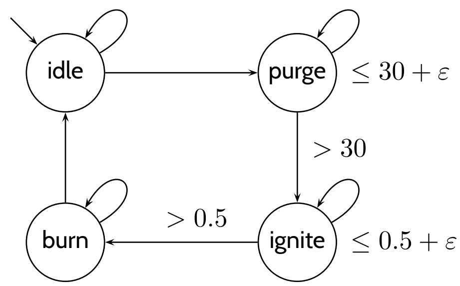
$$\lceil \neg \text{ignite} \rceil \,;\, \lceil \text{ignite} \rceil \xrightarrow{\le 0.5} \lceil \text{ignite} \rceil \qquad \text{(Stab-3)}$$

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



Gas Burner Controller: Inputs

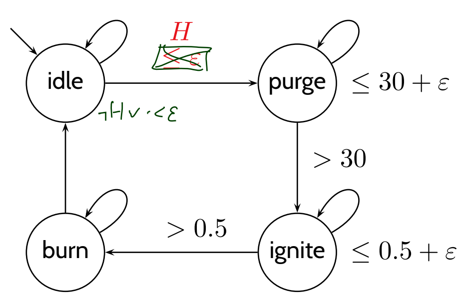
$\lceil \mathsf{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{idle} \rceil$	(Syn-1)
$\lceil \mathsf{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{burn} \rceil$	(Syn-2)
$\lceil \neg \mathsf{idle} \rceil \,;\, \lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow \lceil \mathsf{idle} \rceil$	(Stab-1)
$\lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \mathsf{idle} \rceil$	(Stab-1-init)
$\lceil \neg \mathsf{burn} \rceil \,;\, \lceil \mathsf{burn} \wedge H \wedge F \rceil \longrightarrow \lceil \mathsf{burn} \rceil$	(Stab-4)



– 8 – 2017-11-23 – Sexa –

Gas Burner Controller: Inputs

$\lceil \mathsf{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{idle} \rceil$	(Syn-1)
$\lceil \mathsf{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{burn} \rceil$	(Syn-2)
$\lceil \neg \mathsf{idle} \rceil \,;\, \lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow \lceil \mathsf{idle} \rceil$	(Stab-1)
$\lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \mathsf{idle} \rceil$	(Stab-1-init)
$\lceil \neg \mathsf{burn} \rceil \,;\, \lceil \mathsf{burn} \wedge H \wedge F \rceil \longrightarrow \lceil \mathsf{burn} \rceil$	(Stab-4)



– 8 – 2017-11-23 – Sexa –

Gas Burner Controller: Inputs

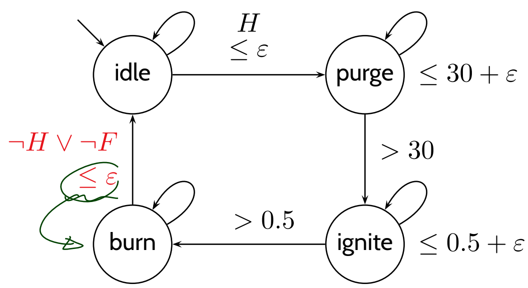
$$\lceil \mathsf{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{idle} \rceil \qquad \text{(Syn-1)}$$

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

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

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

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



Gas Burner Controller: Inputs

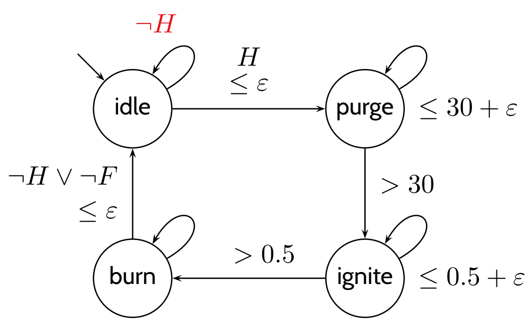
$$\lceil \mathsf{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{idle} \rceil \qquad \text{(Syn-1)}$$

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

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

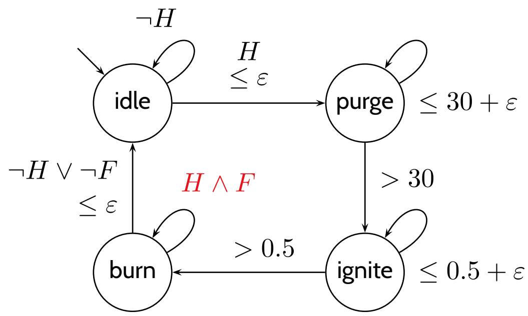
$$\lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \mathsf{idle} \rceil \qquad \text{(Stab-1-init)}$$

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



Gas Burner Controller: Inputs

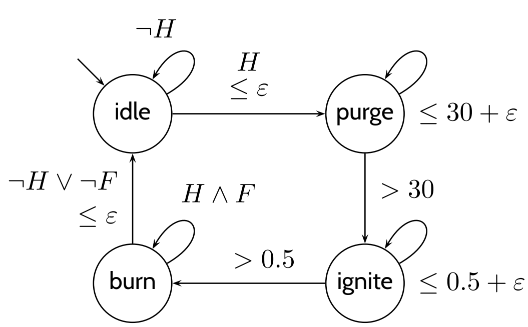
$\lceil \mathsf{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{idle} \rceil$	(Syn-1)
$\lceil \mathsf{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{burn} \rceil$	(Syn-2)
$\lceil \neg \mathsf{idle} \rceil \,;\, \lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow \lceil \mathsf{idle} \rceil$	(Stab-1)
$\lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \mathsf{idle} \rceil$	(Stab-1-init)
$\lceil \neg \mathsf{burn} \rceil \,;\, \lceil \mathsf{burn} \wedge H \wedge F \rceil \longrightarrow \lceil \mathsf{burn} \rceil$	(Stab-4)



– 8 – 2017-11-23 – Sexa –

Gas Burner Controller: Inputs

$\lceil \mathsf{idle} \wedge H \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{idle} \rceil$	(Syn-1)
$\lceil \mathsf{burn} \wedge (\neg H \vee \neg F) \rceil \xrightarrow{\varepsilon} \lceil \neg \mathsf{burn} \rceil$	(Syn-2)
$\lceil \neg \mathsf{idle} \rceil \,;\, \lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow \lceil \mathsf{idle} \rceil$	(Stab-1)
$\lceil \mathsf{idle} \wedge \neg H \rceil \longrightarrow_0 \lceil \mathsf{idle} \rceil$	(Stab-1-init)
$\lceil \neg \mathsf{burn} \rceil \,;\, \lceil \mathsf{burn} \wedge H \wedge F \rceil \longrightarrow \lceil \mathsf{burn} \rceil$	(Stab-4)

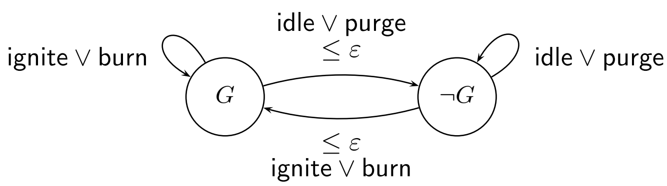


– 8 – 2017-11-23 – Sexa –

Gas Burner Controller: Outputs

$G : \{0, 1\}$

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



Gas Burner Controller: Environment Assumptions

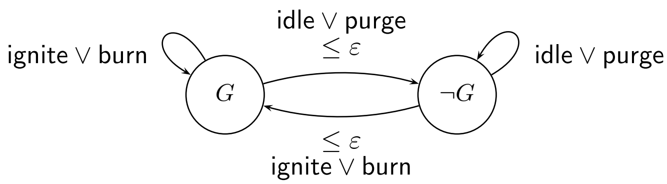
$G : \{0, 1\}$

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

Gas Burner Controller: Environment Assumptions

$G : \{0, 1\}$

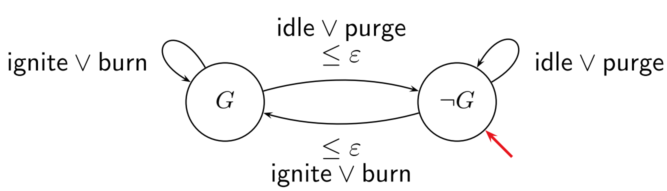
$$\lceil\rceil \vee \lceil \neg G \rceil \, ; \, true \qquad \text{(Init-4)}$$



Gas Burner Controller: Environment Assumptions

$G : \{0, 1\}$

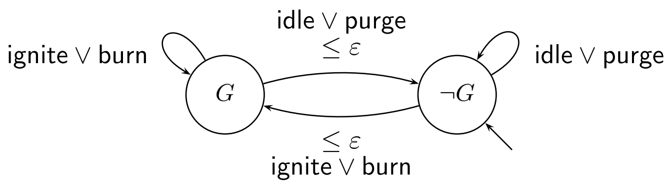
$$\lceil\rceil \vee \lceil \neg G \rceil \, ; \, true \qquad \text{(Init-4)}$$



Gas Burner Controller: Environment Assumptions

$G : \{0, 1\}$

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



Gas Burner Controller: Environment Assumptions

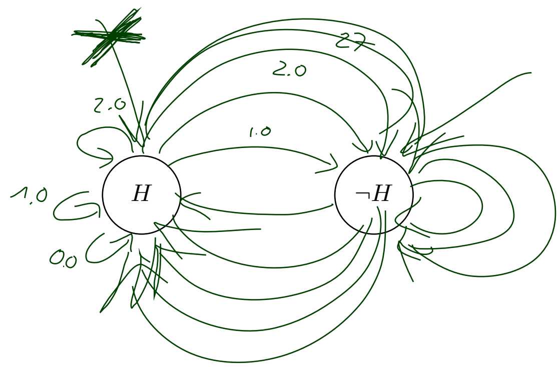
$H : \{0, 1\}$

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

Gas Burner Controller: Environment Assumptions

$H : \{0, 1\}$

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



Gas Burner Controller: Environment Assumptions

$H : \{0, 1\}$

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

Gas Burner Controller: Environment Assumptions

$F : \{0, 1\}$

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

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

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

Gas Burner Controller: Environment Assumptions

$F : \{0, 1\}$

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

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

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

Gas Burner Controller: Environment Assumptions

$F : \{0, 1\}$

$\lceil\rceil \vee \lceil \neg F \rceil ; true$ (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)

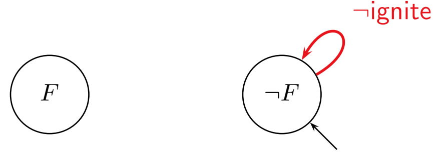
Gas Burner Controller: Environment Assumptions

$F : \{0, 1\}$

$\lceil\rceil \vee \lceil \neg F \rceil ; true$ (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)



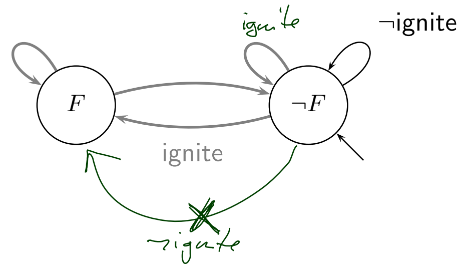
Gas Burner Controller: Environment Assumptions

$$F : \{0, 1\}$$

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

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

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



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**)

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

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

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

- Controller hardware platforms can
  - **read inputs**, **change local state**,
  - **wait**, **write outputs**.
- If we limit **controller behaviour descriptions** to these "operations", there's (at least) no principle **obstacle** to **implement** the design.
- One such **limited specification language**:
  - **DC Implementables**,
  - a set of patterns of **DC Standard Forms**.
- **DC Implementables** basically contrain:
  - local state changes, synchronisation with inputs
  - and outputs, timed stability and progress
- This is sufficient to formalise a **correct** (safe) **Gas Burner** controller design specification.

References

References

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