Recall: Number of Regions

Lemma 4.28. Let $X$ be a set of clocks, $c_x \in \mathbb{N}_0$ the maximal constant for each $x \in X$, and $c = \max\{c_x | x \in X\}$. Then $(2c + 2)|X| \cdot (4c + 3)$ is an upper bound on the number of regions.

- In the desk lamp controller,
  - $\langle \text{off}, \{0\} \rangle \xrightarrow{\text{press}} \langle \text{light}, \{0\} \rangle$
  - $\langle \text{light}, \{0\} \rangle \xrightarrow{\text{press}} \langle \text{bright}, \{0\} \rangle$
  - $\langle \text{light}, \{0\} \rangle \xrightarrow{\text{press}} \langle \text{bright}, (0, 1) \rangle$
  - $\langle \text{light}, \{0\} \rangle \xrightarrow{\text{press}} \langle \text{bright}, (2, 3) \rangle$
  - $\langle \text{light}, \{0\} \rangle \xrightarrow{\text{press}} \langle \text{bright}, \{3\} \rangle$

Which seems to be a complicated way to write just:

$\langle \text{light}, \{0\} \rangle \xrightarrow{\text{press}} \langle \text{bright}, [0, 3] \rangle$

- Can't we constructively abstract $L$ to:

$\langle \text{off}, \{0\} \rangle \langle \text{light}, \{0\} \rangle \langle \text{bright}, [0, 3] \rangle$

What is a Zone?

**Definition.** A (clock) zone is a set $z \subseteq (\mathbb{X} \rightarrow \text{Time})$ of valuations of clocks $X$ such that there exists $\varphi \in \Phi(X)$ with $\nu \in z$ if and only if $\nu | = \varphi$.

**Example:**

- $x \leq 2 \\
  x > 1 \\
  y \geq 1 \\
  y < 2 \\
  x - y \geq 0$

- Note: Each clock constraint $\varphi$ is a symbolic representation of a zone.

- But: There's no one-on-one correspondence between clock constraints and zones.

The zone $z = \emptyset$ corresponds to $(x > 1 \land x < 1) \land (x > 2 \land x < 2)$, ...
To this end: remove all upper bounds. Algorithm in general are found.

Given a constraint \( \nu \in \ell,\alpha,\phi,Y,\ell \nu \), \( \nu \) can be computed by taking edge \( E \in \nu \). The following operations can be carried out by manipulating \( \nu \):

- Label zone \( \nu \) via the set \( W \) := \{0, \ldots, x> \} \subset \nu \).

- The stocktaking: What's missing?

- More Examples: Zone or Not?

- Zone-based reachability. In other words.
Good News Cont'd

Pros and Cons

\[ \begin{align*}
\text{Difference Bound Matrices} & \quad (x, y) \in \mathbb{R}^2 \\
\text{1) clock hiding} & \quad \forall \ell, \alpha, \varphi, \nu \\
\text{2) reachability analysis usually is explicit wrt. discrete locations:} & \\
\text{3) confined wrt. size of discrete state space} & \\
\end{align*} \]
### Educational Objectives:

- What are TA extended? Why is that useful?
- What's an urgent/committed location? What's the difference?
- What's an urgent channel?
- Where has the notion of “input action” and “output action” correspondences in the formal semantics?

### Content:

- Extended TA:
  - Data-Variables, Structuring Facilities, Restriction of Non-Determinism
  - The Logic of Uppaal

### Data-V ariables

- When modelling controllers as timed automata, it is sometimes desirable to have (local and shared) variables. E.g. count number of open doors, or intermediate positions of gas valve.

- Adding variables with finite range (possibly grouped into finite arrays) to any finite-state automata concept is straightforward:
  - If we have control locations $L_0 = \{ \ell_1, \ldots, \ell_n \}$, and want to model, e.g., the valve range as a variable $v$ with $D(v) = \{0, \ldots, 2\}$, then just use $L = L_0 \times D(v)$ as control locations, i.e. encode the current value of $v$ in the control location, and consider updates of $v$ in the $\lambda \rightarrow$.
  - $L$ is still finite, so we still have a proper TA.
  - But: writing $\lambda \rightarrow$ is tedious.
  - So: have variables as “first class citizens” and let compilers do the work.
  - Interestingly, many examples in the literature live without variables: the more abstract the model is, i.e., the fewer information it keeps track of (e.g. in data variables), the easier the verification task.

### Data-V ariables and Expressions

- Let $(v,w \in V)$ be a set of (integer) variables.
- $(\psi \in \Psi(V))$: integer expressions over $V$ using function symbols $+, -, \ldots$
- $(\phi \in \Phi(V))$: integer (or data) constraints over $V$ using integer expressions, predicate symbols $=, <, \leq, \ldots$, and boolean logical connectives.

- Let $(x,y \in X)$ be a set of clocks.
- $(\phi \in \Phi(X,V))$: (extended) guards, defined by $\phi ::= \phi_{clk} | \phi_{int} | \phi_1 \land \phi_2$ where $\phi_{clk} \in \Phi(X)$ is a simple clock constraint (as defined before) and $\phi_{int} \in \Phi(V)$ an integer (or data) constraint.

### Examples:

- Extended guard or not extended guard? Why?
  - (a) $x < y \land v > 2$
  - (b) $x < y \lor v > 2$
  - (c) $v < 1 \lor v > 2$
  - (d) $x < v$
Modification or Reset Operation

• New: A modification or reset (operation) is $x := 0$, $x \in X$, or $v := \varphi \int V$, $v \in V$, $\varphi \int V \in \Psi(V)$.

• By $R(X,V)$ we denote the set of all resets.

• By $\vec{r}$ we denote a finite list $\langle r_1, \ldots, r_n \rangle$, $n \in \mathbb{N}$, of reset operations $r_i \in R(X,V)$; $\langle \rangle$ is the empty list.

• By $R(X,V)^*$ we denote the set of all such lists of reset operations.

Examples: Modification or not? Why?

(a) $x := y$, (b) $x := v$, (c) $v := x$, (d) $v := w$, (e) $v := 0$.

Structuring Facilities

Global decl.: clocks, data variables, channels, constants

• Binary and broadcast channels: $\text{chan} c$ and broadcast $\text{chan} b$.

• Templates of timed automata.

• Instantiation of templates (instances are called process).

• System definition: list of processes.

Restricting Non-determinism

• Urgent locations — enforce local immediate progress.

• Committed locations — enforce atomic immediate progress.

• Urgent channels — enforce cooperative immediate progress.

• Urgent channel press;

Urgent Locations: Only an Abbreviation...

Replace $\ell \text{ urgent}$ with $\ell \varphi \land z = 0$ where $z$ is a fresh clock:

• reset $z$ on all in-going edges,

• add $z = 0$ to invariant.

Question: How many fresh clocks do we need in the worst case for a network of $N$ extended timed automata?

Extended Timed Automata

Definition 4.39. An extended timed automaton is a structure $A_e = (L, C, B, U, X, V, I, E, \ell_{ini})$ where $L, B, X, I, \ell_{ini}$ are as in Def. 4.3, except that location invariants in $I$ are downward closed, and where

- $C \subseteq L$: committed locations,
- $U \subseteq B$: urgent channels,
- $V$: a set of data variables,
- $E \subseteq L \times B !? \times \Phi(X,V) \times \mathcal{R}(X,V)^* \times L$: a set of directed edges such that $(\ell, \alpha, \varphi, \vec{r}, \ell') \in E \land \text{chan}(\alpha) \in U \Rightarrow \varphi = \text{true}$.

Edges $(\ell, \alpha, \varphi, \vec{r}, \ell')$ from location $\ell$ to $\ell'$ are labelled with an action $\alpha$, a guard $\varphi$, and a list $\vec{r}$ of reset operations.