Contents & Goals

Last Lecture:
- Decidability of the location reachability problem:
  - region automaton
  - zones

This Lecture:
- Educational Objectives:
  - Capabilities for following tasks/questions.
  - By 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

Example (Partly Already Seen in Uppaal Demo)

System:
- $L_U$
- $U_v$
- $U_y$

Data-Variables

- 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 a 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.
reset operations of $\vec{r}$

$\phi$

guard $\alpha$

action are labelled with an $\ell$ to $\ell$ from location $\ell, \alpha, \phi, \vec{r}, \ell$

Extended timed automata?

N of $\phi$

How many fresh clocks do we need in the worst case for a network

true $=$ $\phi$ $\Rightarrow$ $\psi$

$U \in \alpha$ ($\land$ $E \in \alpha$) such that

$\ell, \alpha, \phi, \vec{r}, \ell$

$\gamma$.

$R\times X, V$ $\Phi(\times B \times L \subseteq E \times *)$ to invariant.

$\gamma$

 Replace extended timed automata?

Urgent locations: Only an abbreviation...

Committed locations:

$\ell \subseteq C$

where $\gamma$

committed locations:

$\ell \subseteq C$

$X, V, I, E, \ell$

$\gamma$

Instantiation of templates (instances are called

Templates of timed automata.

Binary and broadcast channels: chan $\gamma$

Global declaration of of clocks, data variables, channels, and constants.

Modification or reset operation?

Examples:

$a x := v$

$y := x$

$w := v$

$0 := v$

$(a)$

$(b)$

$(c)$

$(d)$

$(e)$

$2 v > v$ $\lor$

$2 v > v$ $\land$

$x < y$

$1 v < v$ $\lor$

$1 v > v$ $\lor$

$x = y$

$2 v > v$ $\land$

$x < y$

$w := v$

$x := v$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$

$x < v$

$(d)$

$x := v$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$

$\Phi(\gamma)$
The operational semantics of networks: Transformation Translation

\[ I = \nu \rightarrow D \]

Now, let's focus on the transformation translation aspect of operational semantics. The equation

\[ I \rightarrow D \]

represents the transformation process from the initial state of the network to its final state. This equation is the core of the transformation translation framework, allowing us to analyze and manipulate networks through a series of steps that can be visualized as a process of transformation.

In the context of Operational Semantic Networks, this transformation translation is crucial for understanding how networks evolve over time. The equation:

\[ I \rightarrow D \]

shows the dynamic nature of these networks, where the initial state (\( I \)) is transformed into a final state (\( D \)) through a series of transformations or operations. This process is essential for modeling and analyzing complex systems, where the state of the system changes over time.

The equation:

\[ I \rightarrow D \]

can be interpreted as a sequence of operations that transform the initial state into the final state. Each operation (\( \rightarrow \)) represents a step in the transformation process, and the goal is to understand how these operations work together to achieve the desired transformation from \( I \) to \( D \).

In summary, the transformation translation is a fundamental concept in the study of operational semantics of networks, allowing us to model and analyze the behavior of these systems through a series of transformations.
Theorem 4.41.

If \( A_1, \ldots, A_n \) is a set of pure timed automata, then the operational semantics of \( C(A_1, \ldots, A_n) \) and \( \text{chan} b_1, \ldots, b_m \cdot (A_1 \parallel \ldots \parallel A_n) \), where \( \{b_1, \ldots, b_m\} = \bigcup_{i=1}^{n} B_i \), coincide, i.e.

\[
T(C(A_1, \ldots, A_n)) = T(\text{chan} b_1, \ldots, b_m \cdot (A_1 \parallel \ldots \parallel A_n)).
\]