Last Lecture:

- Decidability of the location reachability problem:
  - region automaton & zones
- Extended Timed Automata syntax

This Lecture:

**Educational Objectives:** Capabilities for following tasks/questions.

- What’s an urgent/committed location? What’s the difference? Urgent channel?
- Where has the notion of “input action” and “output action” correspondences in the formal semantics?
- How can we relate TA and DC formulae? What’s a bit tricky about that?
- Can we use Uppaal to check whether a TA satisfies a DC formula?

**Content:**

- Extended TA semantics
- The Logic of Uppaal
- Testable DC
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,
- \( E \subseteq L \times B \times \Phi(X, V) \times R(X, V)^* \times L \): a set of directed edges such that

\[ (\ell, \alpha, \varphi, \vec{r}, \ell') \in E \land \text{chan}(\alpha) \in U \implies \varphi = \text{true}. \]

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.
Definition 4.40. Let $A_{e,i} = (L_i, C_i, B_i, U_i, X_i, V_i, I_i, E_i, \ell_{ini,i})$, $1 \leq i \leq n$, be extended timed automata with pairwise disjoint sets of clocks $X_i$.

The operational semantics of $C(A_{e,1}, \ldots, A_{e,n})$ (closed!) is the labelled transition system

$$T_e(C(A_{e,1}, \ldots, A_{e,n})) = (Conf, \text{Time} \cup \{\tau\}, \{\lambda \mapsto | \lambda \in \text{Time} \cup \{\tau\}\}, C_{ini})$$

where

- $X = \bigcup_{i=1}^{n} X_i$ and $V = \bigcup_{i=1}^{n} V_i$,
- $Conf = \{\langle \vec{l}, \nu \rangle | \vec{l_i} \in L_i, \nu : X \cup V \rightarrow \text{Time}, \nu |\equiv \bigwedge_{k=1}^{n} I_k(\ell_k)\}$,
- $C_{ini} = \{\langle \vec{l_{ini}}, \nu_{ini} \rangle\} \cap Conf$,

and the transition relation consists of transitions of the following three types.
• **Now**: \( \nu: X \cup V \rightarrow \text{Time} \cup \mathcal{D}(V) \)

• Canonically extends to \( \nu: \Psi(V) \rightarrow \mathcal{D} \) (valuation of expression).

• “\( \models \)” extends canonically to expressions from \( \Phi(X, V) \).

• **Extended timeshift** \( \nu + t, t \in \text{Time} \), applies to clocks only:
  • \((\nu + t)(x) := \nu(x) + t, x \in X\),
  • \((\nu + t)(v) := \nu(v), v \in V\).

• **Effect of modification** \( r \in R(X, V) \) on \( \nu \), denoted by \( \nu[r] \):

\[
\nu[x := 0](a) := \begin{cases} 
0, & \text{if } a = x, \\
\nu(a), & \text{otherwise}
\end{cases}
\]

\[
\nu[v := \psi_{\text{int}}](a) := \begin{cases} 
\nu(\psi_{\text{int}}), & \text{if } a = v, \\
\nu(a), & \text{otherwise}
\end{cases}
\]

• We set \( \nu[\langle r_1, \ldots, r_n \rangle] := \nu[r_1] \ldots [r_n] = (((\nu[r_1])[r_2]) \ldots )[r_n] \).
• An **internal transition** \( \langle \vec{\ell}, \nu \rangle \xrightarrow{\tau} \langle \vec{\ell}', \nu' \rangle \) occurs if there is \( i \in \{1, \ldots, n\} \) such that

- there is a \( \tau \)-edge \( (\ell_i, \tau, \varphi, \vec{r}, \ell'_i) \in E_i \),
- \( \nu \models \varphi \),
- \( \vec{\ell}' = \vec{\ell}[\ell_i := \ell'_i] \),
- \( \nu' = \nu[\vec{r}] \),
- \( \nu' \models I_i(\ell'_i) \),
- (♣) if \( \ell_k \in C_k \) for some \( k \in \{1, \ldots, n\} \) then \( \ell_i \in C_i \).
A synchronisation transition \( \langle \vec{\ell}, \nu \rangle \xrightarrow{\tau} \langle \vec{\ell}', \nu' \rangle \) occurs if there are \( i, j \in \{1, \ldots, n\} \) with \( i \neq j \) such that

- there are edges \((\ell_i, b!, \varphi_i, \vec{r}_i, \ell_i') \in E_i\) and \((\ell_j, b?, \varphi_j, \vec{r}_j, \ell_j') \in E_j\),
- \( \nu \models \varphi_i \land \varphi_j \),
- \( \vec{\ell}' = \vec{\ell}[\ell_i := \ell_i'][\ell_j := \ell_j'] \),
- \( \nu' = \nu[\vec{r}_i][\vec{r}_j] \),
- \( \nu' \models I_i(\ell_i') \land I_j(\ell_j') \),
- (♣) if \( \ell_k \in C_k \) for some \( k \in \{1, \ldots, n\} \) then \( \ell_i \in C_i \) or \( \ell_j \in C_j \).
A delay transition \( \langle \vec{\ell}, \nu \rangle \xrightarrow{t} \langle \vec{\ell}, \nu + t \rangle \) occurs if

- \( \nu + t \models \bigwedge_{k=1}^{n} I_{k}(\ell_{k}) \),
- (♣) there are no \( i, j \in \{1, \ldots, n\} \) and \( b \in U \) with \( (\ell_{i}, b!, \varphi_{i}, \vec{r}_{i}, \ell'_{i}) \in E_{i} \) and \( (\ell_{j}, b?, \varphi_{j}, \vec{r}_{j}, \ell'_{j}) \in E_{j} \),
- (♠) there is no \( i \in \{1, \ldots, n\} \) such that \( \ell_{i} \in C_{i} \).
Restricting Non-determinism: Urgent Location

\[
\begin{align*}
\mathcal{P} & \quad \mathcal{Q} & \quad \mathcal{R} \\
\begin{array}{c}
p_0 \\
p_1 \quad b? \\
p_2 
\end{array} & \quad \begin{array}{c}
q_0 \\
q_1 \quad y := 0, v := 1 \\
q_2 \quad b! , v := 2 \\
q_3 \quad v := 3 
\end{array} & \quad \begin{array}{c}
r_0 \quad w := v \\
r_1 
\end{array}
\end{align*}
\]

<table>
<thead>
<tr>
<th>Property 1</th>
<th>Property 2</th>
<th>Property 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\exists \Diamond w = 1)</td>
<td>(\forall \Box Q.q_1 \implies y \leq 0)</td>
<td>(\forall \Box (\mathcal{P}.p_1 \land Q.q_1 \implies (x \geq y \implies y \leq 0)))</td>
</tr>
<tr>
<td>(\mathcal{N})</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>(\mathcal{N}, q_1) urgent</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>(\mathcal{N}, q_1) comm.</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>(\mathcal{N}, b) urgent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Restricting Non-determinism: Committed Location

\begin{align*}
\mathcal{P} & \quad x := 0 \\
\mathcal{Q} & \quad y := 0, v := 1 \\
\mathcal{R} & \quad w := v
\end{align*}

Property 1
\[ \exists \diamondsuit w = 1 \]
Property 2
\[ \forall \Box \mathcal{Q}.q_1 \implies y \leq 0 \]
Property 3
\[ \forall \Box (\mathcal{P}.p_1 \land \mathcal{Q}.q_1 \implies (x \geq y \implies y \leq 0)) \]

<table>
<thead>
<tr>
<th>Condition</th>
<th>Property 1</th>
<th>Property 2</th>
<th>Property 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{N} )</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>( \mathcal{N}, q_1 \text{ urgent} )</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>( \mathcal{N}, q_1 \text{ comm.} )</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>( \mathcal{N}, b \text{ urgent} )</td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Restricting Non-determinism: Urgent Channel

\[ \begin{align*}
P & \quad x := 0 \\
q_0 & \quad y := 0, v := 1 \\
q_1 & \quad v := 3 \\
r_0 & \quad w := v
\end{align*} \]

- Property 1: \( \exists \Diamond w = 1 \)
- Property 2: \( \forall \Box Q.q_1 \implies y \leq 0 \)
- Property 3: \( \forall \Box (P.p_1 \land Q.q_1 \implies (x \geq y \implies y \leq 0)) \)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Property 1</th>
<th>Property 2</th>
<th>Property 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{N} )</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \mathcal{N}, q_1 ) urgent</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>( \mathcal{N}, q_1 ) comm.</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>( \mathcal{N}, b ) urgent</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
Extended vs. Pure Timed Automata
Extended vs. Pure Timed Automata

\[ \mathcal{A}_e = (L, C, B, U, X, V, I, E, \ell_{ini}) \]
\[ (\ell, \alpha, \varphi, \bar{r}, \ell') \in L \times B!* \times \Phi(X, V) \times R(X, V)^* \times L \]

vs.

\[ \mathcal{A} = (L, B, X, I, E, \ell_{ini}) \]
\[ (\ell, \alpha, \varphi, Y, \ell') \in E \subseteq L \times B!* \times \Phi(X) \times 2^X \times L \]

- \( \mathcal{A}_e \) is in fact (or specialises to) a **pure** timed automaton if
  - \( C = \emptyset \),
  - \( U = \emptyset \),
  - \( V = \emptyset \),
  - for each \( \bar{r} = \langle r_1, \ldots, r_n \rangle \), every \( r_i \) is of the form \( x := 0 \) with \( x \in X \).
  - \( I(\ell), \varphi \in \Phi(X) \) is then a consequence of \( V = \emptyset \).
Theorem 4.41. If $A_1, \ldots, A_n$ specialise to pure timed automata, then the operational semantics of

$$C(A_1, \ldots, A_n)$$

and

$$\text{chan } b_1, \ldots, b_m \bullet (A_1 \parallel \ldots \parallel A_n),$$

where $\{b_1, \ldots, b_m\} = \bigcup_{i=1}^{n} B_i$, coincide, i.e.

$$T_e(C(A_1, \ldots, A_n)) = T(\text{chan } b_1, \ldots, b_m \bullet (A_1 \parallel \ldots \parallel A_n)).$$
Reachability Problems for Extended Timed Automata
Theorem 4.33. \[\text{Location Reachability}\] The location reachability problem for pure timed automata is **decidable**.

Theorem 4.34. \[\text{Constraint Reachability}\] The constraint reachability problem for pure timed automata is **decidable**.

- And what about tea \(^\wedge\) ~ extended timed automata?
Extended Timed Automata add the following features:

- **Data-Variables**
  - As long as the domains of all variables in $V$ are finite, adding data variables doesn’t hurt.
  - If they’re infinite, we’ve got a problem (encode two-counter machine).

- **Structuring Facilities**
  - Don’t hurt — they’re merely abbreviations.

- **Restricting Non-determinism**
  - Restricting non-determinism doesn’t affect (or change) the configuration space $Conf$.
  - Restricting non-determinism only removes certain transitions, so makes reachable part of the region automaton even smaller (not necessarily strictly smaller).
The Logic of Uppaal
Consider $N = C(A_1, \ldots, A_n)$ over data variables $V$.

- **basic formula:**
  \[ atom ::= A_i.\ell \mid \varphi \]
  where $\ell \in L_i$ is a location and $\varphi$ a constraint over $X_i$ and $V$.

- **configuration formulae:**
  \[ term ::= atom \mid \neg term \mid term_1 \land term_2 \]

- **existential path formulae:**
  \[ e\text{-}formula ::= \exists\Diamond term \mid \exists\Box term \]

- **universal path formulae:**
  \[ a\text{-}formula ::= \forall\Diamond term \mid \forall\Box term \mid term_1 \rightarrow term_2 \]

- **formulae:**
  \[ F ::= e\text{-}formula \mid a\text{-}formula \]
Configurations at Time $t$

- Recall: **computation path** (or path) starting in $\langle \ell_0, \nu_0 \rangle, t_0$:

  \[
  \xi = \langle \ell_0, \nu_0 \rangle, t_0 \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle, t_1 \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle, t_2 \xrightarrow{\lambda_3} \ldots
  \]

  which is **infinite or maximally finite**.

- Given $\xi$ and $t \in \text{Time}$, we use $\xi(t)$ to denote the set

  \[
  \{ \langle \ell, \nu \rangle \mid \exists i \in \mathbb{N}_0 : t_i \leq t \leq t_{i+1} \land \ell = \ell_i \land \nu = \nu_i + t - t_i \}.
  \]

  of **configurations at time** $t$.

  - Why is it a set?
  - Can it be empty?
Satisfaction of Uppaal-Logic by Configurations

- We define a **satisfaction relation**

\[
\langle \vec{l}_0, \nu_0 \rangle, t_0 \models F
\]

between **time stamped configurations**

\[
\langle \vec{l}_0, \nu_0 \rangle, t_0
\]

of a network \( C(A_1, \ldots, A_n) \) and **formulae** \( F \) of the Uppaal logic.

- It is defined inductively as follows:

  - \( \langle \vec{l}_0, \nu_0 \rangle, t_0 \models A_i.\ell \) iff \( \ell_{0,i} = \ell \)
  
  - \( \langle \vec{l}_0, \nu_0 \rangle, t_0 \models \varphi \) iff \( \nu_0 \models \varphi \)
  
  - \( \langle \vec{l}_0, \nu_0 \rangle, t_0 \models \neg \text{term} \) iff \( \langle \vec{l}_0, \nu_0, t_0 \rangle \not\models \text{term} \)
  
  - \( \langle \vec{l}_0, \nu_0 \rangle, t_0 \models \text{term}_1 \land \text{term}_2 \) iff \( \langle \vec{l}_0, \nu_0, t_0 \rangle \models \text{term}_{i}, i=1,2 \)
Exists finally:

- \( \langle \vec{l}_0, \nu_0 \rangle, t_0 \models \exists \Diamond \text{term} \)
  iff \( \exists \text{path } \xi \text{ of } \mathcal{N} \text{ starting in } \langle \vec{l}_0, \nu_0 \rangle, t_0 \)
  \( \exists t \in \text{Time}, \langle \vec{l}, \nu \rangle \in \text{Conf} : \)
  \( t_0 \leq t \land \langle \vec{l}, \nu \rangle \in \xi(t) \land \langle \vec{l}, \nu \rangle, t \models \text{term} \)

Example: \( \exists \Diamond \varphi \)
Satisfaction of Uppaal-Logic by Configurations

**Exists globally:**

- \( \langle \vec{l}, \nu \rangle, t_0 \models \exists \square \text{term} \) iff \( \exists \) path \( \xi \) of \( \mathcal{N} \) starting in \( \langle \vec{l}_0, \nu_0 \rangle, t_0 \)
  
  \( \forall t \in \text{Time}, \langle \vec{l}, \nu \rangle \in \text{Conf} : \)
  
  \( t_0 \leq t \land \langle \vec{l}, \nu \rangle \in \xi(t) \Rightarrow \langle \vec{l}, \nu \rangle, t \models \text{term} \)

**Example:** \( \exists \square \varphi \)
Satisfaction of Uppaal-Logic by Configurations

- **Always finally:**

\[ \langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \forall \Diamond \text{term} \quad \text{iff} \quad \langle \vec{\ell}_0, \nu_0 \rangle, t_0 \not\models \exists \Box \neg \text{term} \]

- **Always globally:**

\[ \langle \vec{\ell}_0, \nu_0 \rangle, t_0 \models \forall \Box \text{term} \quad \text{iff} \quad \langle \vec{\ell}_0, \nu_0 \rangle, t_0 \not\models \exists \Diamond \neg \text{term} \]
Satisfaction of Uppaal-Logic by Configurations

Leads to:

- $\langle \vec{l}_0, \nu_0 \rangle, t_0 \models term_1 \rightarrow term_2$ iff

  $\forall \text{ path } \xi \text{ of } \mathcal{N} \text{ starting in } \langle \vec{l}_0, \nu_0 \rangle, t_0$
  $\forall t \in \text{Time}, \langle \vec{l}, \nu \rangle \in \text{Conf} :$
  $t_0 \leq t \land \langle \vec{l}, \nu \rangle \in \xi(t)$
  $\land \langle \vec{l}, \nu \rangle, t \models term_1$
  implies $\langle \vec{l}, \nu \rangle, t \models \forall \diamond term_2$

Example: $\varphi_1 \rightarrow \varphi_2$
• We write \( \mathcal{N} \models e\text{-formula} \) if and only if

\[
\text{for some } \langle \vec{l}_0, \nu_0 \rangle \in C_{ini}, \langle \vec{l}_0, \nu_0 \rangle, 0 \models e\text{-formula},
\]

and \( \mathcal{N} \models a\text{-formula} \) if and only if

\[
\text{for all } \langle \vec{l}_0, \nu_0 \rangle \in C_{ini}, \langle \vec{l}_0, \nu_0 \rangle, 0 \models a\text{-formula},
\]

where \( C_{ini} \) are the initial configurations of \( \mathcal{T}_e(\mathcal{N}) \).

• If \( C_{ini} = \emptyset \), (1) is a contradiction and (2) is a tautology.

• If \( C_{ini} \neq \emptyset \), then

\[
\mathcal{N} \models F \text{ if and only if } \langle \vec{l}_{ini}, \nu_{ini} \rangle, 0 \models F.
\]
Example

A simple state diagram:

- Off
  - Press? \( x := 0 \)
  - Press? \( x > 3 \)

- Light
  - Press? \( x \leq 3 \)

- Bright
Example

\[
x := 0 \quad \tau \\
x > 3 \quad \tau \\
x \leq 3 \quad \tau
\]
Example

- $\mathcal{N} \models \exists \Diamond \mathcal{L}.\text{bright}?$
- $\mathcal{N} \models \exists \Box \mathcal{L}.\text{bright}?$
- $\mathcal{N} \models \exists \Box \mathcal{L}.\text{off}?$
- $\mathcal{N} \models \forall \Diamond \mathcal{L}.\text{light}?$
- $\mathcal{N} \models \forall \Box \mathcal{L}.\text{bright} \implies x \geq 3?$
- $\mathcal{N} \models \mathcal{L}.\text{bright} \rightarrow \mathcal{L}.\text{off}?$
Observer-based Automatic Verification of DC Properties for TA
Model-Checking DC Properties with Uppaal

\[
\begin{align*}
&\text{press?} \\
&x := 0 \\
&x \leq 3 \\
&x > 3
\end{align*}
\]

\[
\begin{align*}
&\text{press?} \\
&y := 0 \\
y < 2 \\
&y > 3
\end{align*}
\]

\[
\begin{align*}
&\mathcal{N} \models 0_\tau \models 3 \diamond 0_\tau . \text{bad}
\end{align*}
\]
• **First Question**: what is the “|=” here?

• **Second Question**: what kinds of DC formulae can we check with Uppaal?

  • **Clear**: Not every DC formula. (Otherwise contradicting undecidability results.)

  • **Quite clear**: $F = \Box[\text{off}]$ or $F = \neg \Diamond[\text{light}]

    (Use Uppaal’s fragment of TCTL, something like $\forall \Box \text{off}$, but not exactly (see later).)

  • **Maybe**: $F = \ell > 5 \implies \Diamond[\text{off}]^5$

  • **Not so clear**: $F = \neg \Diamond([\text{bright}] ; [\text{light}])$
Testable DC Properties
**Testability**

**Definition 6.1.** A DC formula $F$ is called **testable** if an observer (or test automaton (or monitor)) $A_F$ exists such that for all networks $\mathcal{N} = C(A_1, \ldots, A_n)$ it holds that

$$\mathcal{N} \models F \iff C(A'_1, \ldots, A'_n, A_F) \models \forall \Box \neg (A_F . q_{bad})$$

Otherwise it’s called **untestable**.

**Proposition 6.3.** There exist untestable DC formulae.

**Theorem 6.4.** DC implementables are testable.
“Whenever we observe a change from $A$ to $\neg A$ at time $t_A$, the system has to produce a change from $B$ to $\neg B$ at some time $t_B \in [t_A, t_A + 1]$ and a change from $C$ to $\neg C$ at time $t_B + 1$.

**Sketch of Proof:** Assume there is $A_F$ such that, for all networks $\mathcal{N}$, we have

$$\mathcal{N} \models F \text{ iff } C(A'_1, \ldots, A'_n, A_F) \models \forall \square \neg (A_F \cdot q_{bad})$$

Assume the number of clocks in $A_F$ is $n \in \mathbb{N}_0$. 
Consider the following time points:

- $t_A := 1$
- $t^i_B := t_A + \frac{2i-1}{2(n+1)}$ for $i = 1, \ldots, n + 1$
- $t^i_C \in ]t^i_B + 1 - \frac{1}{4(n+1)}, t^i_B + 1 + \frac{1}{4(n+1)}[ $ for $i = 1, \ldots, n + 1$

with $t^i_C - t^i_B \neq 1$ for $1 \leq i \leq n + 1$.

**Example:** $n = 3$
Example: \( n = 3 \)

- The shown interpretation \( \mathcal{I} \) satisfies **assumption** of property.
- It has \( n + 1 \) candidates to satisfy **commitment**.
- By choice of \( t_i^C \), the commitment is not satisfied; so \( F \) not satisfied.
- Because \( A_F \) is a test automaton for \( F \), is has a computation path to \( q_{bad} \).
- Because \( n = 3 \), \( A_F \) can not save all \( n + 1 \) time points \( t_i^B \).
- Thus there is \( 1 \leq i_0 \leq n \) such that all clocks of \( A_F \) have a valuation which is not in \( 2 - t_i^0_B + \left( -\frac{1}{4(n+1)}, \frac{1}{4(n+1)} \right) \)
Example: \( n = 3 \)

- Because \( \mathcal{A}_F \) is a test automaton for \( F \), it has a computation path to \( q_{bad} \).
- Thus there is \( 1 \leq i_0 \leq n \) such that all clocks of \( \mathcal{A}_F \) have a valuation which is not in \( 2 - t^{i_0}_B + \left( -\frac{1}{4(n+1)}, \frac{1}{4(n+1)} \right) \).
- Modify the computation to \( \mathcal{I}' \) such that \( t^{i_0}_C := t^{i_0}_B + 1 \).
- Then \( \mathcal{I}' \models F \), but \( \mathcal{A}_F \) reaches \( q_{bad} \) via the same path.
- That is: \( \mathcal{A}_F \) claims \( \mathcal{I}' \nmodels F \).
- Thus \( \mathcal{A}_F \) is not a test automaton. **Contradiction.**
Testable DC Formulae

Theorem 6.4. DC implementables are testable.

- **Initialisation:** \([\top \lor \top] ; \text{true}\)
- **Sequencing:** \([\top] \rightarrow [\top \lor \top_1 \lor \cdots \lor \top_n]\)
- **Progress:** \([\top] \xrightarrow{\theta} [\neg \top]\)
- **Synchronisation:** \([\top \land \varphi] \xrightarrow{\theta} [\neg \top]\)
- **Bounded Stability:** \([\neg \top] ; [\top \land \varphi] \xrightarrow{\leq \theta} [\top \lor \top_1 \lor \cdots \lor \top_n]\)
- **Unbounded Stability:** \([\neg \top] ; [\top \land \varphi] \rightarrow [\top \lor \top_1 \lor \cdots \lor \top_n]\)
- **Bounded initial stability:** \([\pi \land \varphi] \xrightarrow{\leq \theta_0} [\pi \lor \pi_1 \lor \cdots \lor \pi_n]\)
- **Unbounded initial stability:** \([\pi \land \varphi] \xrightarrow{\gamma_0} [\pi \lor \pi_1 \lor \cdots \lor \pi_n]\)

**Proof Sketch:**
- For each implementable \(F\), construct \(A_F\).
- Prove that \(A_F\) is a test automaton.
**Proof of Theorem 6.4: Preliminaries**

- **Note:** DC does not refer to communication between TA in the network, but only to data variables and locations.

**Example:**

\[ \Diamond ([v = 0] ; [v = 1]) \]

- **Recall:** transitions of TA are only triggered by synchronisation, not by changes of data-variables.

- **Approach:** have auxiliary *step* action.

Technically, replace each

![Diagram](image)

by

![Diagram](image)
Proof of Theorem 6.4: Sketch

- Example: $[\pi] \xrightarrow{\theta} [\neg \pi]$
Defnition 6.5.

- A **counterexample formula** (CE for short) is a DC formula of the form:

\[ \text{true} ; ([\pi_1] \land \ell \in I_1) \ldots ([\pi_k] \land \ell \in I_k) ; \text{true} \]

where for **1 ≤ i ≤ k**, 

- \( \pi_i \) are state assertions,
- \( I_i \) are non-empty, and open, half-open, or closed time intervals of the form
  - \((b, e)\) or \([b, e)\) with \(b \in \mathbb{Q}_0^+\) and \(e \in \mathbb{Q}_0^+ \cup \{\infty\}\),
  - \((b, e]\) or \([b, e]\] with \(b, e \in \mathbb{Q}_0^+\).

\((b, \infty)\) and \([b, \infty)\) denote unbounded sets.

- Let \( F \) be a DC formula. A DC formula \( F_{CE} \) is called **counterexample formula for** \( F \) if \( \models F \iff \neg (F_{CE}) \) holds.
Definition 6.5.

- A **counterexample formula** (CE for short) is a DC formula of the form:

\[ \text{true} ; ([\pi_1] \land \ell \in I_1) ; \ldots ; ([\pi_k] \land \ell \in I_k) ; \text{true} \]

where for \( 1 \leq i \leq k \),

- \( \pi_i \) are state assertions,

- \( I_i \) are non-empty, and open, half-open, or closed time intervals of the form
  - \((b, e)\) or \([b, e)\) with \( b \in \mathbb{Q}_0^+ \) and \( e \in \mathbb{Q}_0^+ \cup \{\infty\} \),
  - \((b, e]\) or \([b, e]\) with \( b, e \in \mathbb{Q}_0^+ \).

- \((b, \infty)\) and \([b, \infty)\) denote unbounded sets.

- Let \( F \) be a DC formula. A DC formula \( F_{CE} \) is called **counterexample formula for** \( F \) if \( \models F \iff \neg (F_{CE}) \) holds.
References