Specification Language

Having described the computational model of clocked transition systems and the implementation language of timed SPL, it only remains to fix the specification language. Here we take good old LTL with the additional license of referring to all clocks in the system.

It is obvious how we can use LTL in order to specify untimed properties of timed systems. For timed properties, we will illustrate some of the most important ones:

- **Bounded Response:** Every $p$ should be followed by an occurrence of $q$, not later than $d$ time units.

This can be specified by each of the two following LTL formulas:

$$p \land (T = t_0) \Rightarrow \Box (q \land T \leq t_0 + d)$$

$$p \land (T = t_0) \Rightarrow (T \leq t_0 + d) \mathcal{W} q$$

In the case of model checking, the reference to the free variable $t_0$ is not convenient. Instead, we can augment the system with the following observer:

$$¬ q \land t < d$$

and then verify $\square ¬ Error$.

**Minimal Separation**

Another important timed property is:

- **Minimal Separation:** No $q$ can occur earlier than $d$ time units after an occurrence of $p$.

This can be specified by the following LTL formula:

$$p \land (T = t_0) \Rightarrow (¬ q) \mathcal{W} (T \geq t_0 + d)$$

Again, in the context of model checking, we can construct the following observer:

Consider the following program **UP-DOWN**:

$$x, y: \text{integer where } x = y = 0$$

$$P_1:: \begin{cases} \ell_0: \text{while } x = 0 \text{ do} \\ \ell_1: y := y + 1 \\ \ell_2: \text{while } y > 0 \text{ do} \\ \ell_3: y := y - 1 \\ \ell_4: \end{cases}$$

$$P_2:: \begin{cases} m_0: x := 1 \\ m_1: \end{cases}$$

Assume we assign to it the time bounds $[1, 5]$. We wish to prove for this program two properties:

$$\square (y + at_\ell_1 \leq 3)$$

$$\Diamond (at_\ell_4 \land at_{m_1} \land T \leq 50)$$

What is special about this program is that it contradicts the naive assumption that, in order to generate a behavior with the worst execution time, every process should proceed at the slowest pace possible. Here, in the initial steps, $P_2$ should proceed at its slowest pace, while $P_1$ should rush forward at maximal speed.

Files `updn.smv` and `updn.pf` are available on the course web page.
Dense Time

Obviously, the use of integer time may lead to distortions which can be sensed even with integer constraints. The system $\Phi_1$

\[ t > 1/2 \implies 0 \quad t > 1/2 \implies 0 \]

satisfies the property $\Sigma (T \leq 3 \implies at_0, t_0)$ under the integer-time model. However, under a dense-time model, the system can reach location $\ell_2$ at time $T = 3$.

We conclude that, to reach a better precision, we must use dense time. The main problem is that the dense-time model no longer leads to finite-state systems.

Therefore, we will develop special methods which will enable us to deal with systems whose discrete part is finite state while its clocks vary over a dense domain. This leads us into the model of timed automata of [Alur & Dill].

Symbolic Representation

Recall that the state variables are partitioned into $V = D \cup C$. We assume that the discrete variables $D$ range over finite domains. Let $D = \{d_1, \ldots, d_n\}$ be the set of different valuations that the variables in $D$ can assume. For example, for system $\Phi_1$, $D = \{0, 1, 2\}$ are the three possible values that the single discrete variable $\pi$ can assume. We can represent the transition relation as

\[ \rho(D, C, D', C') = \bigvee_{d_i, d_j \in D} D = d_i \land D' = d_j \land \rho_{ij}(C, C'), \]

where, for each $d_i, d_j \in D$,

\[ \rho_{ij}(C, C') = g_{ij}(C) \land C' = r_{ij}(C) \]

In this presentation, $g_{ij}(C)$ is a guard specifying a condition on the current values of the clocks under which a transition from $d_i$ to $d_j$ is allowed. The function $r_{ij}$ is a reset function ensuring that, for each $t_k \in C$ either $r_{ij}(t_k) = 0$ or $r_{ij}(t_k) = t_k$. For example, for $\Phi_1$,

\[ \rho_{01} = \left( \frac{t \geq 1}{0_1} \land \left( t', T' = (0, T) \right) \right)_{C' = r_{01}(C)} \]

Timed Automata

A timed automaton is a CTS with the following restrictions:

- The discrete variables range over finite domains.
- The time dependent component of the transition relations and the progress conditions, are formed as boolean combinations of inequalities of the form $t_i < c_i \lor t_i - t_j < c_{ij}$ where $\sim \in \{<, \leq, >, \geq\}$ and $c_i, c_{ij}$ are natural numbers.
- The only modifications to clocks by non-tick transitions are resets to 0.

The tick Transition

In a similar way, we can decompose the tick transition into the disjunction

\[ \rho_{\text{tick}}(D, C, D', C') = \bigcup_{d_i \in D} D = d_i \land D' = d_i \land \rho_{\text{tick}}(C, C'), \]

where, for each $d_i \in D$,

\[ \rho_{\text{tick}}(C, C') = \exists \Delta \geq 0 : p_i(C + \Delta) \land C' = C + \Delta. \]

For example, for $\Phi_1$,

\[ \rho_{\text{tick}}^{0}(C, C'): \exists \Delta \geq 0 : (t + \Delta \leq 2)_{p_0} \land \left( t', T' = (t + \Delta, T + \Delta) \right)_{C' = C + \Delta} \]

A formula is called $k$-polyhedral if it is a boolean combination of atomic formulas of the forms $t_i \# c$ or $t_i - t_j \# c$, where the relation $\# \in \{<, \leq, >, \geq\}$ and $c \in \{0, \ldots, k\}$.

We restrict our attention to systems such that, for some $k \geq 0$, and each $d_i, d_j \in D$, the guards $g_{ij}(C)$ and the progress conditions $p_i(C)$ are $k$-polyhedral.

An assertion $\varphi(D, C)$ is called $k$-admissible if there exists a decomposition $\varphi(D, C) : \bigcup_{d_i \in D} D = d_i \land \psi_i(C)$ such that each $\psi_i(C)$ is $k$-polyhedral.
The Main Result

The main result which is the basis for symbolic model-checking of dense-time systems is stated by

**Claim 13. Closure of k-admissible Assertions**

If $\varphi$ is a $k$-admissible assertion, then so is its $\rho \vee p_{\text{tick}}$-predecessor.

In order to prove the claim, it is sufficient to show that if $\psi(C)$ is $k$-polyhedral, then so are its $\rho_{ij}$- and $p_{\text{tick}}$-predecessors, for every $d_i, d_j \in \mathcal{D}$.

The general computation of a predecessor is based on the formula:

$$\exists C^\prime : \rho(C, C^\prime) \land \psi(C^\prime).$$

By expanding all formulas into DNF form and observing that existential quantification distributes over disjunctions, we see that it is sufficient to consider the case that $\rho$ and $\psi$ are conjunctions of $k$-atomic formulas.

Consider first the case that $\rho = \rho_{ij}(C, C^\prime)$. In that case, the predecessor is given by

$$\exists C^\prime : g_{ij}(C) \land \psi(C^\prime) \land \bigwedge_{t_i \in C} t'_i = r_{ij}(t_i),$$

which can be simplified to

$$g_{ij}(C) \land \psi(r_{ij}(C)).$$

Proof Continued

A formula of the form $t_i \# c$ is changed into $t_i + \Delta \# c$, which can be rewritten as either $\Delta < c - t_i$ or $c - t_i < \Delta$, for $\preceq \in \{<, \leq\}$. To obtain a uniform representation, we rewrite $\Delta \geq 0$ as $t_0 \leq \Delta$, where $t_0$ is an artificial clock having the constant value 0.

We form a new set of constraints $S$ as follows:

- Each original constraint $t_i - t_j \# c$ is placed in $S$.
- For each pair of constraints $c_i - t_i \prec_i \Delta$ and $\Delta \prec_j c_j - t_j$, we place in $S$ the constraint $c_i - c_j \prec t_j - t_i$ if $c_i \geq c_j$ or the constraint $t_j - t_i \prec c_j - c_i$ if $c_i < c_j$. In both cases, $\prec$ is taken to be strict ($<$) if one of $\prec_i$ or $\prec_j$ is strict.

Finally, we substitute 0 for all occurrences of $t_0$. The conjunction of all constraints within $S$ is the $p_{\text{tick}}$-predecessor of $\psi$. It is not difficult to see that this conjunction is $k$-polyhedral.

**Proof Continued:** $\rho = \rho_{\text{tick}}^i(C, C^\prime)$

Next, consider the case that $\rho = \rho_{\text{tick}}^i(C, C^\prime)$. In this case, the predecessor is given by

$$\exists C^\prime : \exists \Delta \geq 0 : p_i(C + \Delta) \land C^\prime = C + \Delta \land \psi(C^\prime).$$

which can be simplified into

$$\exists \Delta \geq 0 : p_i(C + \Delta) \land \psi(C + \Delta)$$

Let us examine the effect that the replacement of $C$ by $C + \Delta$ has on the various types of atomic formulas.

For formulas of the form $t_i - t_j \# c$, this replacement has no effect, because the addition of $\Delta$ is canceled.

A Simplified Presentation

Assume that the time-progress condition $p_i$ is a conjunction of constraints of the form $t_j < E_j$ or $t_j \leq E_j$. We denote by $p_i^*$ the conjunction obtained by replacing every strict inequality $t_j < E_j$ in $p_i$ by its weaker version $t_j \leq E_j$. In this case, we can simplify further the computation of the $\rho$-predecessor and $p_{\text{tick}}$-predecessor into:

$$\rho_{ij} \land \psi : p_i^*(C) \land g_{ij}(C) \land \psi(r_{ij}(C))$$

$$p_{\text{tick}}^i \land \psi : \exists \Delta \geq 0 : \psi(C + \Delta)$$

Thus, the time-progress condition $p_i^*$ is moved from the computation of the $p_{\text{tick}}$-predecessor to the computation of the $\rho_{ij}$-predecessor.

Usually, we compute first $\varphi_i = \rho_{ij} \land \psi_j$ and then compute $\psi_i = p_{\text{tick}}^i \land \varphi_i$. This can be combined into a single computation $\psi_i = p_{\text{tick}}^i \land (\rho_{ij} \land \psi_j)$, given by

$$\psi_i : \exists \Delta \geq 0 : p_i^*(C + \Delta) \land g_{ij}(C + \Delta) \land \psi_j(r_{ij}(C + \Delta))$$

This presupposes that, being in discrete state $d_i$, we let first time elapse for $\Delta$ time units, and then take the transition to discrete state $d_j$. 

Timed and Hybrid Systems, NYU, Spring, 2007
Simplifying the Computation of $p_{\text{tick}}^i \diamond \psi$

The computation of $p_{\text{tick}}^i \diamond \psi$ can be further simplified, bypassing the Fourier Motzkin elimination. The process can be described as follows:

- First, we infer all constraints of the form $t_i - t_j \neq \sigma_{ij}$ which are implied by single-clock lower and upper bounds. Thus, from the two constraints
  \[ L_j \prec_i t_i \succ_i U_i \]
  we infer
  \[ L_j - U_i \prec_i t_i - t_j \prec_i U_i - L_j \]
  if $U_i - L_j \geq 0$
  otherwise
  where the ordering $\prec_i$ is strict iff one of $\{\prec_i, \succ_i\}$ is.

- Next, we impose the inequalities $t_i \geq 0$, for every $i > 1$, overriding any previous single-clock lower bound constraint.

Working an Example

Let us apply this approach in order to check whether system $\Phi_1$ can reach location $\ell_2$ at time $T \leq 3$, thus violating the property $\square (T \leq 3 \rightarrow \text{at} \cdot \ell_{0,1})$ which is valid for $\Phi_1$ under the integer-time model.

Note that the discrete transitions we have to consider are $ij \in \{12, 01\}$. For both of them,

$\rho_{ij} \wedge \gamma_{ij} : 1 < t \leq 2$

The goal state set is given by

$\varphi_2 : \text{at} \cdot \ell_2 \wedge T \leq 3$

The computation of $\psi_1$ proceeds along the following steps:

Starting assertion

 Undoing the reset $t := 0$

 Conjuncting $1 < t \leq 2$

 Inferring clock differences

 Imposing non-negativity and drawing all conclusions

From $\ell_1$ Back To $\ell_0$

Let us compute the counter example:

(\ell_0, t : 0, T : 0) satisfies $\varphi_0 : 0 \leq T < 2 \wedge 0 \leq t \leq 2 \wedge -2 \leq T - t < 1$

(\ell_1, t : 0, T : 1.1) satisfies $\varphi_1 : 0 \leq T \leq 3 \wedge 0 \leq t \leq 2 \wedge -2 \leq T - t < 2$

(\ell_2, t : 0, T : 2.2) satisfies $\varphi_2 : T \leq 3$

We continue the computation by computing the predecessor of $\varphi_1$ at $\ell_0$.

$\varphi_1 : 0 \leq T \leq 3 \wedge 0 \leq t \leq 2 \wedge -2 \leq T - t < 2$

Starting point

Undo reset and conclude

Conjuncting $1 < t \leq 2$

Inferring clock differences

Imposing non-negativity and drawing all conclusions

Since the initial condition $\Theta : \text{at} \cdot \ell_0 \wedge t = 0$ has a non-empty intersection with $\varphi_0$, we conclude that $\Phi_4$ has a computation reaching location $\ell_2$ with $T \leq 3$.

It follows that the property $\square (T \leq 3 \rightarrow \text{at} \cdot \ell_{0,1})$ is not valid for $\Phi_4$ under the dense-time model.
**Difference Bounds Matrix**

Observing that clock differences are very central to the analysis of timed automata, David Dill in his first paper on the topic, introduced the notion of difference bounds matrix (DBM). Assuming a system \( D \) with \( m \) clocks, a DBM for \( D \) is a \((m+1) \times (m+1)\) matrix \( M \). For each \( i, j \in [0..m] \), \( M[i,j] \) is an entry of the form \( \geq c_{ij} \), where \( \geq \in \{\geq, >\} \), and \( c_{ij} \) is an integer. For a timed automata whose constants are bounded by \( k \), it is required that \( |c_{ij}| \leq k \). The entry \( \geq c_{ij} \) represents the constraint

\[
t_j - t_i \geq c_{ij}
\]

The special clock \( t_0 \) represents the constant \( 0 \).

All the operations needed in order to compute predecessors and the set of reachable states, can be presented as operations on DBM’s. We will illustrate this in the following slides.

Instead of representing DBM’s in their tabular form, we prefer their graphical presentation as bounds graph (BG).

**Tightening the Constraints**

Whenever there is an edge \( e_{ij} \) labeled by \( c_{ij} \) from node \( t_i \) to \( t_j \) and an edge \( e_{jk} \) labeled by \( c_{jk} \) from node \( t_j \) to \( t_k \), draw a new edge \( e_{ik} \) from node \( t_i \) to \( t_k \), and label it by \( c_{ik} = c_{ij} + c_{jk} \).

If there already exists an edge from node \( t_i \) to \( t_k \) labeled by \( d_{ik} \), retain the edge with the larger label. If \( d_{ik} = c_{ik} \) but the edges are of different types, retain the solid edge.

A graph is inconsistent if it contains a solid self loop with a non-negative label, or a dashed self loop with a positive label.

**Bounds Graphs**

Conjunctions of \( k \)-polyhedral atomic formulas can conveniently be represented by a graph constructed as follows:

- Introduce a special timer \( t_0 \) intended to represent \( 0 \). Then replace all inequalities of the form \( t_i \neq c \) (for \( i > 0 \)) by \( t_i - t_0 \neq c \).
- Place in the graph a node for each timer \( t_i, i \geq 0 \).
- For each constraint \( t_j - t_i > c \), draw a solid edge

\[
\begin{array}{c}
t_i \\ \downarrow c \\
\end{array} \rightarrow 
\begin{array}{c}
t_j \\ \downarrow c \\
\end{array}
\]

- For each constraint \( t_j - t_i \geq c \), draw a dashed edge

\[
\begin{array}{c}
t_i \\ \downarrow c \\
\end{array} \rightarrow 
\begin{array}{c}
t_j \\ \downarrow c \\
\end{array}
\]

- For constraints of the form \( t_j - t_i < c \) or \( t_j - t_i \leq c \), draw the edges corresponding to the constraints \( t_i - t_j > -c \) or \( t_i - t_j \geq -c \), respectively.

**Operations on Bounds Graphs**

**Undoing a Reset**

Let \( G \) be graph representing a \( k \)-polyhedron. To undo the reset operation \( t_i := 0 \), \( i > 0 \). We redirect all edges \( t_i \rightarrow t_j \) to depart from \( t_0 \) and redirect all edges \( t_k \rightarrow t_i \) to arrive to \( t_0 \). This causes an intersection of the original assertion with \( t_i = 0 \) and also removes \( t_i \) from any constraint.

**Intersecting Two Graphs**

Let \( G_1 \) and \( G_2 \) be two graphs representing two convex \( k \)-polyhedra. The graph \( G \) corresponding to their intersection (conjunction) can be obtained by placing in \( G \) all the edges contained in either \( G_1 \) or \( G_2 \), and then tightening.

**Computing a Tick Predecessor**

Let \( G_\psi \) be a graph representing the formula \( \psi \). The graph corresponding to the formula \( \exists \Delta \geq 0 : \psi(C + \Delta) \) can be obtained from \( G_\psi \) by tightening first, removing all edges departing from \( t_0 \), drawing new 0-labeled edges from \( t_0 \) to each \( t_i, i > 0 \), and finally tightening again.
Example: From $\ell_2$ Back to $\ell_1$

We will repeat the process of computing the set of states from which $\varphi_2 : \text{at}_-\ell_2 \land T \leq 3$ is reachable, using the graphical representation.

The goal assertion $\varphi_2 : \text{at}_-\ell_2 \land T \leq 3$ is presented by

```
T  -3  t_0  t
```

Undoing the $t$-reset, intersecting with $1 < t \leq 2$, and tightening, we obtain $\psi_1 : \text{at}_-\ell_1 \land T \leq 3 \land 1 < t \leq 2 \land T - t < 2$, representable as:

```
T  -3  t_0  t  -1
```

Example Continued

Undoing the $t$-reset, intersecting with $1 < t \leq 2$, and tightening we obtain $\psi_0 : \text{at}_-\ell_0 \land 0 \leq T < 2 \land 1 < t \leq 2 \land -2 \leq T - t < 1$, representable as:

```
T  -2  t_0  t
```

Taking the $\text{tick}$-predecessor, we obtain $\psi_0 : \text{at}_-\ell_0 \land 0 \leq T < 2 \land 0 \leq t \leq 2 \land -2 \leq T - t < 1$

```
T  -2  t_0  t  -1
```

Winding Up at $\ell_1$

Taking the $\text{tick}$-predecessor, we obtain

$$\varphi_1 : \text{at}_-\ell_1 \land 0 \leq T \leq 3 \land 0 \leq t \leq 2 \land -2 \leq T - t < 2$$

representable as:

```
T  -3  t_0  t  -2
```

Operations Leading to Non-Convex Polyhedra

A bounds graph represent a convex polyhedron. The operation of tightening does not change the semantics (geometry) of the graph. The operations of reset reversal, intersection, and tick reversal all preserve convexity.

Union

Given two polyhedra represented by graphs $G_1$ and $G_2$, their union $G_1 \cup G_2$ is in general non-convex and, therefore, cannot be represented by a single bounds graph. Often, non-convex polyhedra are represented as a set of bound graphs. We refer to such a set as polyhedral set.

Subtraction

Let $G_1$ and $G_2$ be two bound graphs. We wish to compute the polyhedron which is the subtraction $G_1 - G_2$. Let $e_{ij}$ be an edge in $G_2$ connecting node $t_i$ to $t_j$ with weight $c_{ij}$. We denote by $G(\neg e_{ij})$ the bounds graph which has a single edge $\tilde{e}_{ij}$ connecting $t_j$ to $t_i$ with weight $-c_{ij}$. The type of $\tilde{e}_{ij}$ is opposite to that of $e_{ij}$, that is $\tilde{e}_{ij}$ is solid (representing strict inequality) if $e_{ij}$ is dashed (representing weak inequality).

Assume that $G_2$ contains the edges $e_1, \ldots, e_m$. Then the graph subtraction is given by

$$G_1 - G_2 : \quad G_1 \cap G(\neg e_1) \cup \cdots \cup G_1 \cap G(\neg e_m)$$
Checking Inclusion Between Polyhedral Sets

Assume that \( S_1 = G_1 \cup \cdots \cup G_m \) and \( S_1 = H_1 \cup \cdots \cup H_n \). We wish to check that \( S_1 \subseteq S_2 \). Obviously,

\[
S_1 \subseteq S_2 \quad \text{iff} \quad G_i \subseteq S_2 \quad \text{for each} \quad i = 1, \ldots, m \\
G_i \subseteq S_2 \quad \text{iff} \quad (\cdots ((G_i - H_1) - H_2) - \cdots) - H_n = \emptyset
\]

Let \( [\bar{d}, \bar{t}] \) denote the equivalence class of all states which are equivalent to \( (\bar{d}, \bar{t}) \). Obviously there is a finite number of equivalence classes which is bounded by \( |D|K^m n! \), where \( |D| \) is the number of all the different valuations of the discrete part of the state and \( n \) is the number of clocks.

Region Graphs

Another approach to the analysis of dense-time finite-state CTS’s is based on the definition of an equivalence relation among the states. Let \( D \) be a CTS whose timed guards and progress conditions are boolean combinations of inequalities of the forms \( t_i < c_i \) and \( t_j > d_j \), where all constants \( c_i, d_j \) are non-negative and strictly smaller than \( K \). We denote by \( |t_i| \) the integer part of \( t_i \), i.e., the largest integer not greater than \( t_i \), and by \( fr(t_i) \) the fractional part of \( t_i \) given by \( fr(t_i) = t_i - |t_i| \). Obviously, \( 0 \leq fr(t_i) < 1 \).

We say that states \( (\bar{d}, \bar{t}) \) and \( (\bar{d}', \bar{t}') \) are equivalent, denoted \( (\bar{d}, \bar{t}) \sim (\bar{d}', \bar{t}') \), if

- \( \bar{d} = \bar{d}' \). That is, the two states have an identical discrete part.
- For every \( t_i \in C \), either \( t_i > K \) and \( t_i^* > K \), or \( 0 \leq |t_i| = |t_i^*| \leq K \).
- For every \( t_i \geq t_j \in C \), either \( t_i - t_j > K \) and \( t_i^* - t_j^* > K \), or \( 0 \leq |t_i - t_j| = |t_i^* - t_j^*| \leq K \) and \( fr(t_i - t_j) > 0 \Rightarrow fr(t_i^* - t_j^*) > 0 \).
- For every \( t_i, t_j \in C \) such that \( t_i \leq K \) and \( t_j \leq K \), \( sign(fr(t_i) - fr(t_j)) = sign(fr(t_i^*) - fr(t_j^*)) \), including the case that \( t_j = t_j^* = t_0 = 0 \).

Properties of Regions

Following is the partition of timed states into regions for the case of two clocks \( C = \{t, \bar{T} \} \) and \( K = 3 \).

In this diagram, there are 25 2-dimensional regions, 41 = 16 + 16 + 9 1-dimensional regions, and 16 0-dimensional regions.

The equivalence underlying the region definition is a bi-simulation relation. This is established by the following claims.

Claim 14. Let \( (\bar{d}, \bar{t}) \) and \( (\bar{d}', \bar{t}') \) be two states such that \( (\bar{d}, \bar{t}) \sim (\bar{d}', \bar{t}') \), and let \( \varphi \) be an assertion formed as a positive boolean combination of inequalities \( t_i - t_j < c_i \) and \( t_i - t_j > d_i \), for \( 0 \leq c_i, d_j \leq K \), \( i > 0 \), \( j \geq 0 \). Then

\[
(\bar{d}, \bar{t}) \models \varphi \quad \text{iff} \quad (\bar{d}', \bar{t}') \models \varphi
\]
Proving the correctness of the region graph automaton:

Continuation of Proof that $\sim$ is a Bi-Simulation

**Claim 15.** For every states $(\bar{d}, \bar{t}), (\bar{d}', \bar{t}')$, and $(\bar{d}, \bar{t})$ is a $\rho_2$-successor of $(\bar{d}, \bar{t})$, there exists a state $(\bar{d}', \bar{t}')$, such that $(\bar{d}, \bar{t}) \sim (\bar{d}', \bar{t}')$ and $(\bar{d}', \bar{t}')$ is a $\rho_2$-successor of $(\bar{d}', \bar{t}')$.

$$\forall (\bar{d}, \bar{t}) \rho_2 \exists (\bar{d}', \bar{t}') \sim (\bar{d}, \bar{t})$$

For the proof we consider separately the different types of transitions.

Consider the case that $(\bar{d}, \bar{t})$ is obtained by applying a discrete transition to $(\bar{d}, \bar{t})$, moving from discrete state $d_i$ to discrete state $d_j$. In this case, $\bar{d} = d_i$, $\bar{d}' = d_j$, $\bar{t}$ satisfies $g_{ij}$ and $\bar{t}' = r_{ij}(\bar{t})$. We take $(\bar{d}', \bar{t}')$ to be $(d_j, r_{ij}(\bar{t}))$ and claim that, due to the equivalence $(\bar{d}, \bar{t}) \sim (\bar{d}', \bar{t}')$, $\bar{d}' = d_i$ and $\bar{t}'$ satisfies $g_{ij}$. It follows that $(d_j, r_{ij}(\bar{t}'))$ is a $\rho_2$-successor of $(\bar{d}', \bar{t}')$ and is equivalent to $(\bar{d}, \bar{t})$.

The Region Graph Automaton

For every finite-state CTS $\mathcal{D}$, we can construct a region-graph automaton $\mathcal{R}_\mathcal{D}$ whose locations are the different $K$-regions corresponding to the clock space of $\mathcal{D}$. There exists a transition from region $r_i$ to region $r_j$ iff there exist states $(\bar{d}, \bar{t})$, $(\bar{d}', \bar{t}')$ such that $(\bar{d}, \bar{t}) \in r_i$, $(\bar{d}', \bar{t}') \in r_j$, and $(\bar{d}', \bar{t}')$ is a $\rho_2$-successor of $(\bar{d}, \bar{t})$.

For example, $\mathcal{R}_{\sigma_1}$ is an automaton consisting of 82 regions. Below are all the reachable states of $\mathcal{R}_{\sigma_1}$.
In most cases, \( \sigma \) is time-divergent iff it contains infinitely many states \((d, t_i)\) such that \( f_r(t_i) = 0 \) but \( 0 < t_i \leq K \) for some \( t_i \in C \). For these cases, it is possible to define a set of accepting regions, which are the regions corresponding to such states. A computation of \( \mathcal{R}_D \) is then required to visit accepting regions infinitely often.

**Region Equivalence Between Systems**

A region-observation corresponding to a computation \( s_0, s_1, \ldots \) is the infinite sequence of regions \([s_0], [s_1], \ldots\).

The finite-state CTS \( D_1 \) is said to be region-equivalent to the finite-state CTS \( D_2 \) if every observation \( r_0, r_1, r_2, \ldots \) is equal, up to stuttering, to an observation of \( D_2 \), and vice versa.

Obviously, if \( D_1 \) is region-equivalent to \( D_2 \), and \( \varphi \) is a \( K \)-bounded next-free LTL formula, then

\[
D_1 \models \varphi \quad \text{iff} \quad D_2 \models \varphi
\]