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 $\mathcal{D}$ with m clocks, a DBM for $\mathcal{D}$ 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 $\succ c_{ij}$, where $\succ \in \{>, \geq\}$, 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 $\succ c_{ij}$ represents the constraint

$$t_i - t_j \succ c_{ij}$$

The special clock t_0 represents the constant 0.

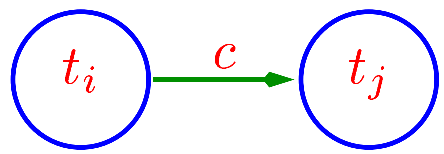
All the operations needed in order to compute predecessors and the set of reachable states, can be presented as operations on DBM's. We will illustrate this in the following slides.

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

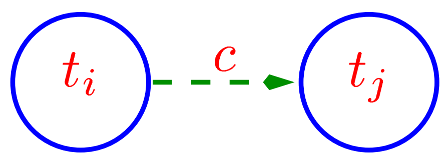
Bound Graphs

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 $t_i \# c$ (for $i > 0$) by $t_i - t_0 \# c$.
- 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



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



- 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.

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.

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.

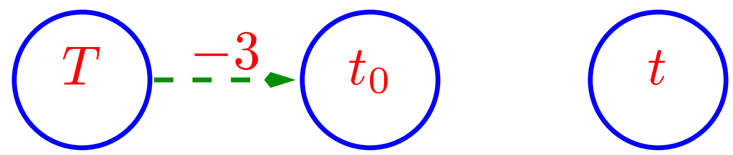
Computing a $tick$ Predecessor

Let G_ψ be a graph representing the formula ψ. The graph corresponding to the formula $\exists\Delta \geq 0 : \psi(C + \Delta)$ can be obtained from G_ψ 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.

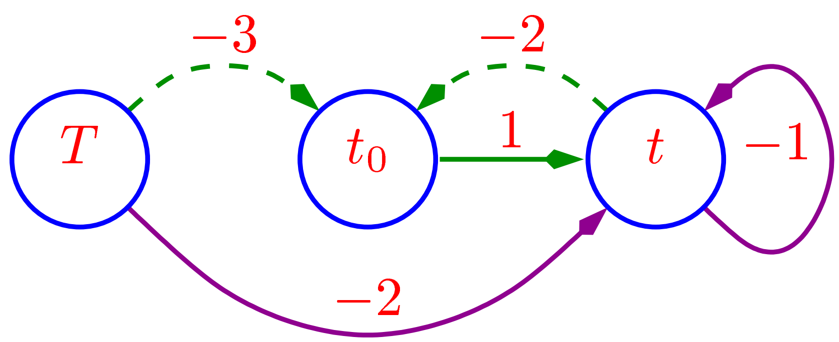
Example

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

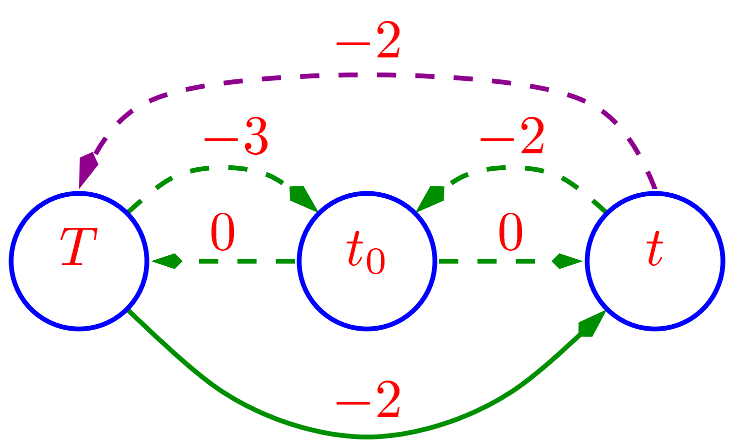
The goal assertion $\varphi_2 : at\_\ell_2 \ \wedge \ T \leq 3$ is presented by



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



Taking the $tick$-predecessor, we obtain
$\varphi_1 : at\_\ell_1 \ \wedge \ 0 \leq T \leq 3 \ \wedge \ 0 \leq t \leq 2 \ \wedge \ -2 \leq T - t < 2$



Example Continued

To Be Completed!!!

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.

There are, though, other useful operations which do not 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 $\widetilde{e_{ij}}$ connecting t_j to t_i with weight $-c_{ij}$. The type of $\widetilde{e_{ij}}$ is opposite to that of e_{ij}, that is $\widetilde{e_{ij}}$ is solid (representing strict inequality) iff 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

$$G_1 - G_2 : \qquad G_1 \cap G(\neg e_1) \quad \cup \quad \cdots \quad \cup \quad G_1 \cap G(\neg e_m)$$

Checking Inclusion Between Polyhedral Sets

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

$$\begin{array}{lll} S_1 \subseteq S_2 & \text{iff} & G_i \subseteq S_2 \text{ for each } i = 1, \ldots, m \\ G_i \subseteq S_2 & \text{iff} & (\cdots((G_i - H_1) - H_2) - \cdots) - H_n = \emptyset \end{array}$$