Operations on Zones

B. Srivathsan

Chennai Mathematical Institute, India

As we saw in the last lecture, the reachability algorithm works with special sets of valuations called zones. There are two important computations involving zones: successor computation, and inclusion with respect to $\mathit{Closure}_M$. In this lecture, we will see how the successor can be computed efficiently.

Recall from the last lecture that the successor computation $(q, Z) \Rightarrow^t (q', Z')$ for a transition $t = (q, g, R, q')$ proceeds in the following steps.

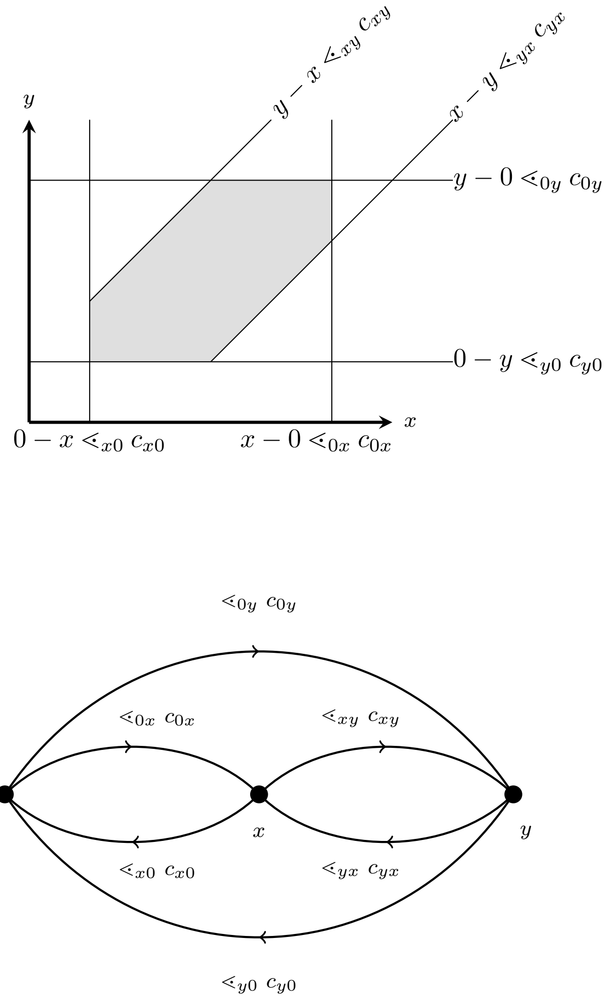
$$(q, Z) \xrightarrow{guard} (q, Z \wedge g) \xrightarrow{reset} (q, [R](Z \wedge g)) \xrightarrow{elapse} (q, Z')$$

In the above, $Z \wedge g$ represents the set of valuations that satisfy the constraints of both Z and g; the set $[R](Z \wedge g)$ represents the set of valuations obtained by resetting clocks in R from every valuation in $Z \wedge g$ and finally Z' is the set of valuations obtained by elapsing an arbitrary amount of time from each valuation in $[R](Z \wedge g)$. We will see each of these operations in detail. But before that, we need a convenient way of representing zones.

1 Representing zones

Consider an arbitrary zone as shown in Figure 1. A zone gets defined by six constraints for every pair of variables as shown in the figure. Each constraint is a half-space. The standard way to represent a zone is to consider a Difference Bound Matrix (DBM) specifying each half space. Instead of considering zones as DBMs, we prefer to work with *distance graphs*, the graph representation of a zone. Figure 1 shows the distance graph representation of the arbitrary zone.

Definition 1 (Distance graph) A *distance graph* G has clocks as vertices, with an additional special vertex x_0 representing constant 0. Between every two vertices there is an edge with a *weight* of the form $(\lessdot, c)$ where $c \in \mathbb{Z}$ and $\lessdot \in \{\leq, <\}$ or $(\lessdot, c) = (<, \infty)$. An edge $x \xrightarrow{\lessdot c} y$ represents a constraint $y - x \lessdot c$: or in words, the distance from x to y is bounded by c. We let $[\![G]\!]$ be the set of valuations of clock variables satisfying all the constraints given by the edges of G with the restriction that the value of x_0 is 0.



$y - x \lessdot_{xy} c_{xy}$
$x - y \lessdot_{yx} c_{yx}$
y
$y - 0 \lessdot_{0y} c_{0y}$
$0 - y \lessdot_{y0} c_{y0}$
x
$0 - x \lessdot_{x0} c_{x0}$
$x - 0 \lessdot_{0x} c_{0x}$
$\lessdot_{0y}\ c_{0y}$
$\lessdot_{0x}\ c_{0x}$
$\lessdot_{xy}\ c_{xy}$
0
x
y
$\lessdot_{x0}\ c_{x0}$
$\lessdot_{yx}\ c_{yx}$
$\lessdot_{y0}\ c_{y0}$

Figure 1: An arbitrary zone and its distance graph

For readability, we will often write 0 instead of x_0. A concrete example of a zone and its distance graph is given in Figure 2.

Arithmetic over weights

An arithmetic over the weights $(\lessdot, c)$ can be defined as follows [BY04].

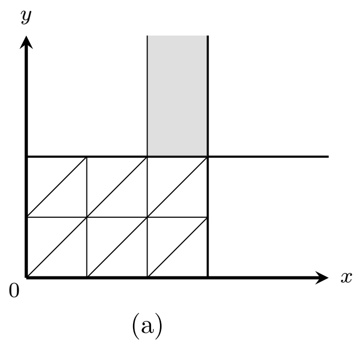
Equality $(\lessdot_1, c_1) = (\lessdot_2, c_2)$ if $c_1 = c_2$ and $\lessdot_1 = \lessdot_2$.

Addition $(\lessdot_1, c_1) + (\lessdot_2, c_2) = (\lessdot, c_1 + c_2)$ where $\lessdot =<$ iff either $\lessdot_1$ or $\lessdot_2$ is $<$.

Minus $-(\lessdot, c) = (\lessdot, -c)$.

Order $(\lessdot_1, c_1) < (\lessdot_2, c_2)$ if either $c_1 < c_2$ or ($c_1 = c_2$ and $\lessdot_1 =<$ and $\lessdot_2 =\leq$).

This arithmetic lets us talk about the weight of a path as a weight of the sum of its edges.



$x < 3$ $\quad$ $y < \infty$

$x > 2$ $\quad$ $y > 2$

(b)

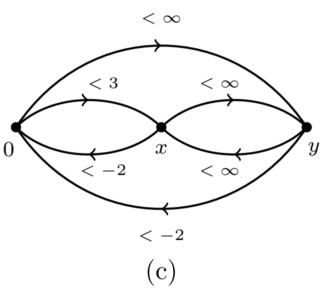


Figure 2: A concrete zone represented pictorially by the shaded portion in (a); by constraints defining it in (b); and by its distance graph in (c)

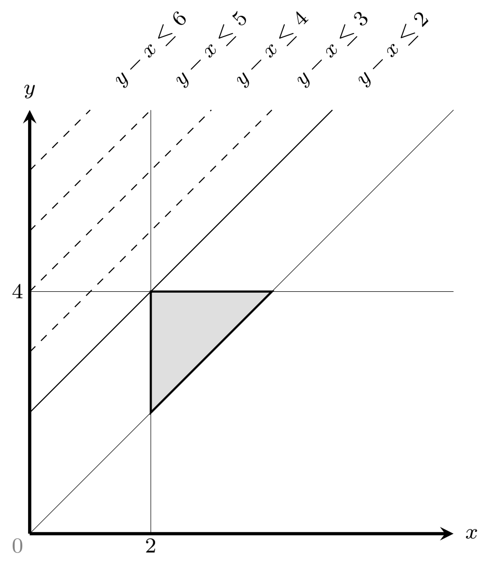


Figure 3: The $y - x \lessdot_{xy} c_{xy}$ constraint for the zone gets defined by the constraints $x \geq 2$ and $y \leq 4$. Adding the constraint $y - x \leq c$ for any $c \geq 2$ still gives the same zone

A cycle in a distance graph G is said to be *negative* if the sum of the weights of its edges is at most $(<, 0)$; otherwise the cycle is *positive*.

Canonical form

Consider the zone showed in Figure 3. It is defined by the constraints:

$$(\ x \geq 2 \ \wedge \ y \leq 4 \ \wedge \ x - y \leq 0 \ )$$

If one adds $y - x \leq 6$ to the above system of constraints, the solution set does not change. In fact, one could add any constraint $y - x \leq c$ for $c \geq 2$ and the solution set remains the same. This is because the diagonal constraint gets defined by $x \geq 2$ and $y \leq 4$. This is

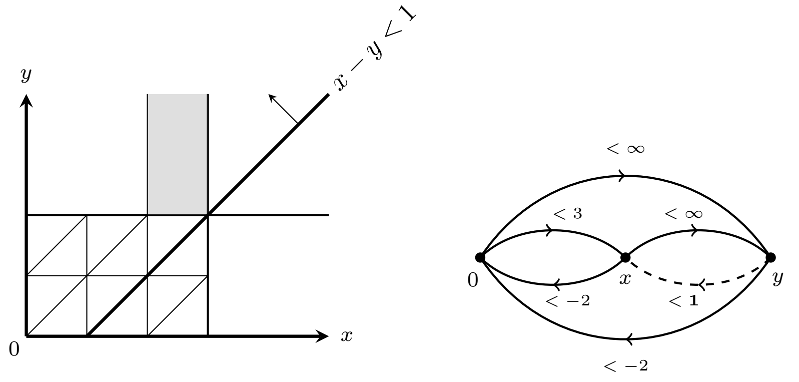


Figure 4: Tightening the constraints: the thick gray line in the picture on the left shows the tight diagonal bordering the shaded area, and the arrow marks the half space given by the inequality; the corresponding constraint obtained by tightening the distance graph is shown in by a dashed line in the graph.

depicted in Figure 3. Adding the constraint given by the dashed diagonals to the set of constraints given above does not change the solution set. However, one cannot go below the diagonal $y - x \leq 2$, as doing so would alter the solution set. It is a *tight* constraint.

Every zone can be represented by a distance graph in *canonical form.* A distance graph is in *canonical form* if the weight of the edge from x to y is the lower bound of the weights of paths from x to y. For instance, the distance graph given in Figure 2 (c) is not canonical. Consider the edge from y to x. The weight of the edge is $(<, \infty)$. However there is a path $y \xrightarrow{<-2} 0 \xrightarrow{<3} x$ that has weight $(<, 1)$. The tightened edge should therefore be $y \xrightarrow{<1} x$. This corresponds to the diagonal $x - y < 1$. We depict this in Figure 4.

Given a distance graph, its canonical form can be computed by using an all-pairs shortest paths algorithm like Floyd-Warshall's [BY04] in time $\mathcal{O}(|X|^3)$ where $|X|$ is the number of clocks. Note that the number of vertices in the distance graph is $|X| + 1$.

Distance graphs with empty solution set

We examine when is the solution set $[\![G]\!]$ of a distance graph G empty, in other words, when is the system of inequalities represented by the distance graph inconsistent. This is crucial in designing the algorithm for $Z \subseteq \mathit{Closure}_M(Z')$.

Consider the distance graph of Figure 4. The solution set represented by this graph is not empty. Now suppose we change the value of the edge $x \to 0$ to $(<, -4)$. This will correspond to a solution set where $x < 3$ and $x > 4$ and hence this solution set will be empty. Let us now see how this gets reflected in the distance graph. Look at the cycle $0 \xrightarrow{<3} x \xrightarrow{<-4} 0$. The sum of the weights of edges in the cycle is $(<, -1)$ which is a negative value. We will now recall the folklore result that a distance graph G has a non-empty solution set iff all cycles in G are positive.

Proposition 2 A distance graph G has only positive cycles iff $[\![G]\!] \neq \emptyset$.

Proof

If there is a valuation $v \in [\![G]\!]$ then we replace every edge $x \xrightarrow{\lessdot_{xy} c_{xy}} y$ by $x \xrightarrow{\leq d} y$ where $d = v_y - v_x$. We have $d \lessdot_{xy} c_{xy}$. Since every cycle in the new graph has value 0, every cycle in G is positive.

For the other direction suppose that every cycle in G is positive. Let $\overline{G}$ be the canonical form of G. Clearly $[\![G]\!] = [\![\overline{G}]\!]$, i.e., the constraints defined by G and by $\overline{G}$ are equivalent. It is also evident that all the cycles in $\overline{G}$ are positive.

We say that a variable x is *fixed* in $\overline{G}$ if in this graph we have edges $0 \xrightarrow{\leq c_x} x$ and $x \xrightarrow{\leq -c_x} 0$ for some constant c_x. These edges mean that every valuation in $[\![\overline{G}]\!]$ should assign c_x to x.

If all the variables in $\overline{G}$ are fixed then the value of every cycle in $\overline{G}$ is 0, and the valuation assigning c_x to x for every variable x is the unique valuation in $[\![\overline{G}]\!]$. Hence, $[\![\overline{G}]\!]$, and in consequence $[\![G]\!]$ are not empty.

Otherwise there is a variable, say y, that is not fixed in $\overline{G}$. We will show how to fix it. Let us multiply all the constraints in $\overline{G}$ by 2. This means that we change each arrow $x_1 \xrightarrow{\lessdot c} x_2$ to $x_1 \xrightarrow{\lessdot 2c} x_2$. Let us call the resulting graph H. Clearly H is in canonical form since $\overline{G}$ is. Moreover $[\![H]\!]$ is not empty iff $[\![\overline{G}]\!]$ is not empty. The gain of this transformation is that for our chosen variable y we have in H edges $0 \xrightarrow{\lessdot_{0y} c_{0y}} y$ and $y \xrightarrow{\lessdot_{y0} c_{y0}} 0$ with $c_{y0} + c_{0y} \geq 2$. This means that there is a natural number d such that $(\leq, d) \leq (\lessdot_{0y}, c_{0y})$ and $(\leq, -d) \leq (\lessdot_{y0}, c_{y0})$. Let H_d be H with edges to and from y changed to $0 \xrightarrow{\leq d} y$ and $y \xrightarrow{\leq -d} 0$, respectively. This is a distance graph where y is fixed. We need to show that there is no negative cycle in this graph.

Suppose that there is a negative cycle in H_d. Clearly it has to pass through 0 and y since there was no negative cycle in H. Suppose that it uses the edge $0 \xrightarrow{\leq d} y$, and suppose that the next used edge is $y \xrightarrow{\lessdot_{yx} c_{yx}} x$. The cycle cannot come back to y before ending in 0 since then we could construct a smaller negative cycle. Hence all the other edges in the cycle come from H. Since H is in the canonical form, a path from x to 0 can be replaced by the edge from x to 0, and the value of the path will not increase. This means that our hypothetical negative cycle has the form $0 \xrightarrow{\leq d} y \xrightarrow{\lessdot_{yx} c_{yx}} x \xrightarrow{\lessdot_{x0} c_{x0}} 0$. By canonicity of H we have $(\lessdot_{yx}, c_{yx}) + (\lessdot_{x0}, c_{x0}) \geq (\lessdot_{y0}, c_{y0})$. Putting these two facts together we get

$$(\leq, 0) > (\leq, d) + (\lessdot_{yx}, c_{yx}) + (\lessdot_{x0}, c_{x0}) \geq (\leq, d) + (\lessdot_{y0}, c_{y0})$$

but this contradicts the choice of d which supposed that $(\leq, d) + (\lessdot_{y0}, c_{y0})$ is positive. The proof when the hypothetical negative cycle passes through the edge $y \xrightarrow{\leq -d} 0$ is analogous.

Summarizing, starting from G that has no negative cycles we have constructed a graph H_d that has no negative cycles, and has one more variable fixed. We also know that if $[\![H_d]\!]$ is not empty then $[\![G]\!]$ is not empty. Repeatedly applying this construction we get a graph where all the variables are fixed and no cycle is negative. As we have seen above the semantics of such a graph is not empty. □

Intersection of two distance graphs

For two distance graphs G_1, G_2 which are not necessarily in canonical form, we denote by $\min(G_1, G_2)$ the distance graph where each edge has the weight equal to the minimum of

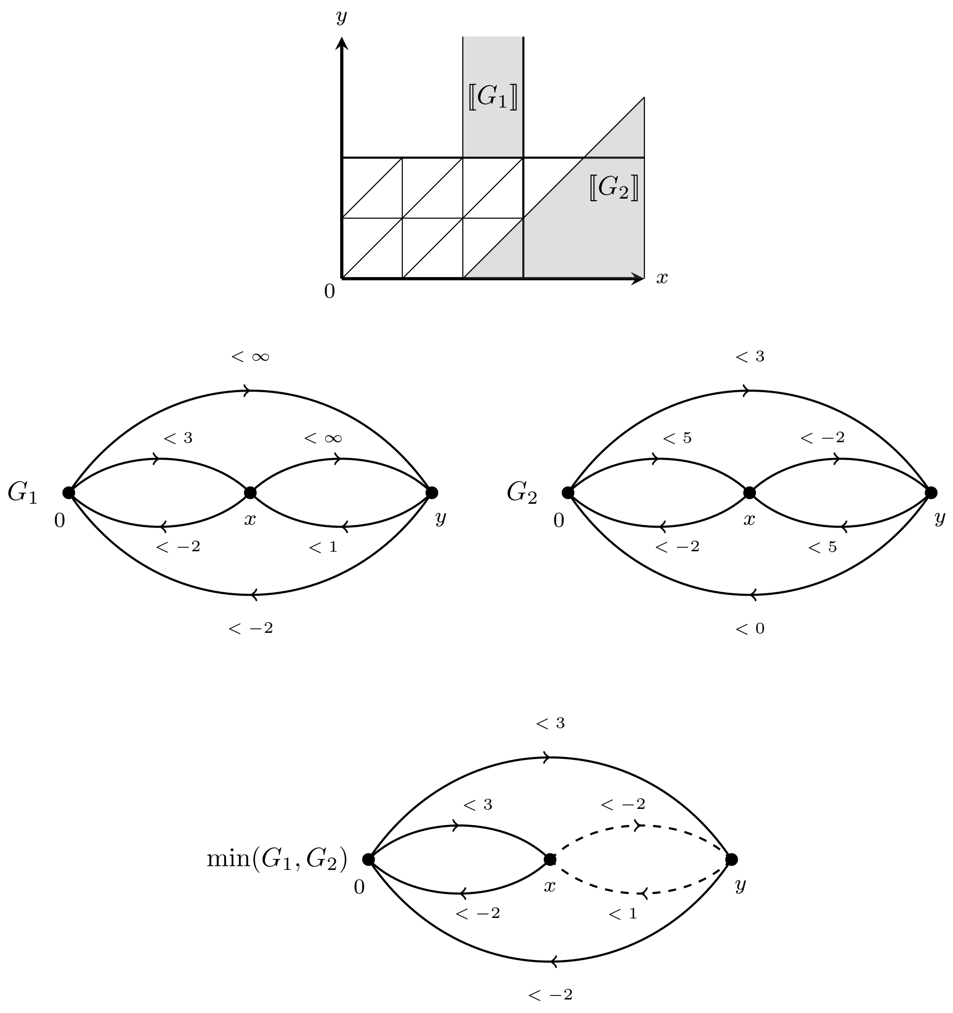


Figure 5: Two zones $[\![G_1]\!]$ and $[\![G_2]\!]$ are shown in the topmost figure. The corresponding distance graphs are given below. Note that $[\![G_1]\!]$ does not intersect $[\![G_2]\!]$. This is captured by the distance graph $\min(G_1, G_2)$ which has a negative cycle. Additionally, the $\min(G_1, G_2)$ graph shows exactly which constraints are responsible for non-intersection.

the corresponding weights in G_1 and G_2. Even though this graph may be not in canonical form, it should be clear that it represents intersection of the two arguments, that is, $[\![\min(G_1, G_2)]\!] = [\![G_1]\!] \cap [\![G_2]\!]$; in other words, the valuations satisfying the constraints given by $\min(G_1, G_2)$ are exactly those satisfying all the constraints from G_1 as well as G_2.

From Proposition 2, the intersection $[\![G_1]\!] \cap [\![G_2]\!]$ is empty iff the distance graph $\min(G_1, G_2)$ has a negative cycle. Figure 5 shows an example.

We are now ready to give the different steps involved in the successor computation from a zone.

2 Guard intersection

3 Reset

4 Time elapse

5 Zone inclusion

References

[BY04] Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, pages 87–124. Springer, 2004.