Timed automata with diagonal constraints

B. Srivathsan

Chennai Mathematical Institute, India

In this lecture, we will consider timed automata that can additionally take guards of the form $x - y \leq 5$, $z - x > 20$, etc. Such guards are called *diagonal guards* or *diagonal constraints*.

Definition 1 (Guards with diagonal constraints) Let X be a set of clocks. The set of *guards with diagonal constraints* $\Phi(X)$ is given by the following grammar:

$$\Phi(X) \ := \ x \lessdot c \mid x \gtrdot c \mid x - y \lessdot c \mid x - y \gtrdot c \mid \Phi(X) \ \wedge \ \Phi(X)$$

where $x, y \in X$, $\lessdot \in \{\leq, <\}$, $\gtrdot \in \{\geq, >\}$ and $c \in \mathbb{Z}$.

As seen from the above definition, the above set contains the normal guards of the form $x \leq 3 \ \wedge \ y > 4$, in addition to the diagonal constraints like $x - y = 5$.

Definition 2 (Timed automata with diagonal constraints) A timed automaton with diagonal constraints is a normal timed automaton whose guards come from the set given in Definition 1. Let us call such timed automata as d-timed automata.

Results: In this lecture, we will see two results:

1. Diagonal constraints do not add expressive power to timed automata [BPDG98].
2. d-timed automata are exponentially more succinct than timed automata [BC05].

We will see both of them in detail below.

Remark 3 In literature, timed automata without diagonal constraints are sometimes referred to as *diagonal free* timed automata.

1 Diagonals to diagonal-free

Diagonal constraints do not add expressive power. This is the substance of the following theorem.

Theorem 4 *[BPDG98] For every d-timed automaton $\mathcal{A}$, there exists a (diagonal-free) timed automaton $\mathcal{A}_{df}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{df})$.*

Proof
Let $\mathcal{A} = (Q, \Sigma, X, T, q_0, F)$. Suppose the automaton $\mathcal{A}$ has only one diagonal constraint $x - y \leq c$. The idea is to use the states of the automaton to maintain if the constraint is true or false. In the initial state q_0 when the value of both x and y are 0, the constraint $x - y \leq c$ is true if $0 \leq c$, otherwise it is false. Note that when time elapses, the value of the diagonal $x - y$ *does not change*. The only time when the diagonal changes is during a reset of either of the clocks x, y. We should modify the transitions to take care of this change.

The states of $\mathcal{A}_{df}$ would be $Q \times \{0, 1\}$. A state $(q, 0)$ denotes the fact that whenever the automaton reaches this state the constraint $x - y \leq c$ evaluates to false. Similarly when the automaton reaches the state $(q, 1)$ the constraint $x - y \leq c$ evaluates to true. Let us call this additional bit as the *attribute* of the state.

Consider an arbitrary transition of $\mathcal{A}$:

$$q \xrightarrow[R]{g} q'$$

For each such transition, we have the following transitions in $\mathcal{A}_{df}$ depending on whether g contains the diagonal constraint $x - y \leq c$ or not.

g does not contain $x - y \leq c$: In this case, we need to only take care of passing the attribute correctly from q to q'.

1. If $x \notin R$ and $y \notin R$, then we have in $\mathcal{A}_{df}$:

   $$(q, 0) \xrightarrow[R]{g} (q', 0) \qquad \text{and} \qquad (q, 1) \xrightarrow[R]{g} (q', 1)$$

   This is because if both x and y are not reset, the truth value of $x - y \leq c$ remains the same after the transition.

2. If $x \in R$ and $y \in R$, then we have in $\mathcal{A}_{df}$:

   $$(q, b) \xrightarrow[R]{g} (q', 1) \text{ if } 0 \leq c$$
   $$(q, b) \xrightarrow[R]{g} (q', 0) \text{ if } 0 > c$$

   for $b \in \{0, 1\}$.

3. If $x \in R$ and $y \notin R$, then we need to check if $-y \leq c$ or not. This can be done by modifying the guard, giving the following transitions:

   $$(q, b) \xrightarrow[R]{g \wedge -y \leq c} (q', 1) \qquad \text{and} \qquad (q, b) \xrightarrow[R]{g \wedge -y > c} (q', 0)$$

   for $b \in \{0, 1\}$.

4. The case of $x \notin R$ and $y \in R$ is similar to the above case.

g is of the form $x - y \leq c \ \wedge \ g_1$: In this case, we will consider only $(q, 1)$ for this transition and add in $\mathcal{A}_{df}$:

$$(q, 1) \xrightarrow[R]{g_1 \ \wedge \ g_2} (q', b)$$

where b and g_2 depend on the reset set and are given by the conditions for passing the attribute mentioned in the previous case.

The initial state of $\mathcal{A}_{df}$ would be $(q, 1)$ if $0 \leq c$ and $(q, 0)$ otherwise. The set of final states would be $F \times \{0, 1\}$.

It can be proved by induction that for every accepting run in $\mathcal{A}$ there is a corresponding accepting run in $\mathcal{A}_{df}$.

The above construction described the situation when there is a single diagonal constraint $x - y \leq c$. Given $\mathcal{A}$, there can be only a finite number of diagonal constraints, say k. Then the states of $\mathcal{A}_{df}$ would be $Q \times \{0, 1\}^k$ maintaining an attribute for each constraint. The transitions are then given by passing each attribute correctly depending on the reset set. □

As we see, the above construction induces an exponential blowup on the size of the diagonal free timed automaton. Is this blowup necessary or there could be some other construction that can avoid this blowup? This is answered in the next section.

2 Succinctness of diagonal constraint timed automata

We will show that d-timed automata are exponentially more succinct than diagonal free timed automata. To this regard, we will construct a sequence of languages $L_1, L_2, \ldots, L_n, \ldots$ such that for each n there exists a d-timed automaton $\mathcal{A}_n$ of size polynomial in n such that $\mathcal{L}(\mathcal{A}_n) = L_n$ and every diagonal free timed automaton that accepts L_n necessarily has exponentially many states in n. This construction is due to [BC05].

The language sequence: Define L_n to be following language over the singleton alphabet $\{a\}$:

$$\{ \ (a^{2^n}, \tau) \mid 0 < \tau_1 < \tau_2 < \cdots < \tau_{2^n} < 1\}$$

Lemma 5 [BC05] For every L_n, there exists a d-timed automaton $\mathcal{A}_n$ of size polynomial in n such that $\mathcal{L}(\mathcal{A}_n) = L_n$.

Proof
We need an automaton that can count from 1 till 2^n. Suppose we had a counter c that can store n bits and its initial value was 0.

Automaton in Figure 1.1 shows how one can then accept the string a^{2^n}. Each time an a is seen, the counter c is incremented. When it reaches $2^n - 1$ the last letter a is read.

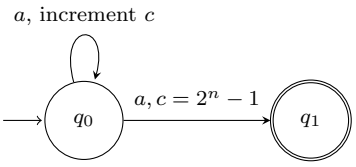

Figure 1.1

Additionally in L_n we required that all these letters should occur within 1 time unit and no two letters occur at the same time. For this, we can add 2 clocks y, z. Add the guard $z < 1$ in the transition from q_0 to q_1 and add the guard and reset $y > 0$, $\{y\}$ in the transition from q_0 to q_0.

However what about the counter? We need a timed automaton. We will now see how we can use clocks to achieve the same result as the counter.

Encoding counter by clocks: To encode the counter of n bits, introduce $2n$ clocks $\{x_i, x_i'$ for $i \in \{1, \ldots, n\}$. Let $c = b_1 b_2 \ldots b_n$ where b_1 denotes the least significant bit and b_n the most significant bit. The bit b_i is simulated by the pair $\{x_i, x_i'\}$: if i^{th} bit is 1, then it is denoted by $x_i - x_i' > 0$, and if i^{th} bit is 0, then $x_i - x_i' = 0$.

For example, if c has the value 10110, then $b_1 = 0, b_2 = 1, b_3 = 1, b_4 = 0, b_5 = 1$ and the clock constraints that encode c are given by:

$$\begin{aligned} x_1 - x_1' &= 0 \\ x_2 - x_2' &> 0 \\ x_3 - x_3' &> 0 \\ x_4 - x_4' &= 0 \\ x_5 - x_5' &> 0 \end{aligned}$$

Incrementing the counter can be done in the following way: if the first $j-1$ bits are 1 and the j^{th} bit is 0, then set all $(b_i)_{1 \le i \le j-1}$ to 0 and set b_j to 1. This gives us a set of n instructions which can be translated to $n-1$ transitions (q_0, a, g_j, R_j, q_0) for $j = \{1, \ldots, n\}$ where

$$\begin{cases} g_j \text{ is } \bigwedge_{i=1}^{j-1} \ (x_i - x_i' > 0) \ \wedge \ (x_j - x_j' = 0) \\ R_j \text{ is } \bigcup_{i=1}^{j-1} \ \{x_i, x_i'\} \ \cup \ \{x_j'\} \end{cases}$$

Moreover we need to add the transition $(q_0, a, g, \{\}, q_1)$ where g checks if $x_i - x_i' > 0$ for all $i \in \{1, \ldots, n\}$.

Initially all clocks are 0 and hence all differences are 0 too. At any point, only one of the n transitions at q_0 will be satisfied. During each transition, the reset simulates the increment of the clock. When all differences are > 0, that is, when $2^n - 1$ a's have been read, the automaton takes the transition to q_1.

Call the above automaton be $\mathcal{A}_n$. It can be easily checked that the language of $\mathcal{A}_n$ is L_n. As there are only 2 states and $n+1$ transitions, the size of $\mathcal{A}_n$ is polynomial in n.

□

The next lemma will show that there is every diagonal free timed automaton that accepts L_n should have atleast 2^n states.

Lemma 6 [BC05] Let $\mathcal{B}_n$ be a diagonal free timed automaton such that $\mathcal{L}(\mathcal{B}_n) = L_n$. Then $\mathcal{B}_n$ has at least 2^n states.

Proof
Suppose $\mathcal{B}_n$ has strictly less than 2^n states. Consider the timed word w defined below:

$$(a^{2^n}, \tau) \qquad \text{s. t.} \qquad \tau_i = \frac{i}{2^n + 1}$$

Clearly, $w \in L_n$. Hence there is an accepting run of $\mathcal{B}_n$ on w:

$$(q_0, v_0) \xrightarrow{\delta_0, \theta_0} (q_1, v_1) \xrightarrow{\delta_1, \theta_1} \dots (q_{2^n}, v_{2^n}) \tag{1.1}$$

where $\delta_i = \frac{1}{2^n+1}$ and θ_i is a transition $(q_i, a, g_i, R_i, q_{i+1})$.

Note that there is a strict elapse of time at each step. However the total time spent during the run is strictly less than 1. Therefore, $v_i + \delta_i$ should be a function that maps all clocks to the open interval $(0, 1)$. From the above run, we know that $v_i + \delta_i$ satisfies g_i, that is, g_i allows the valuation $v_i + \delta_i$. But the guards contain only integer constants and are additionally diagonal free. Hence we can infer that each g_i allows all valuations in the unit hypercube $(0, 1)^X$.

Now, as the number of states in $\mathcal{B}_n$ is strictly less than 2^n there exist two indices i, j such that $q_i = q_j$. This will then give an accepting run for the word obtained by removing the part of w between the i^{th} and j^{th} a's. This would give a contradiction as this word with less than 2^n a's cannot belong to L_n.

In more detail, consider the timed word $(a^{2^n-(j-i)}, \tau)$ where

$$\tau_k = \begin{cases} \frac{k}{2^n+1} & \text{if } k \leq i \\ \\ \frac{k+(j-i)}{2^n+1} & \text{if } k > i \end{cases}$$

Using the fact that every guard g_i allows valuations in the unit hypercube $(0, 1)^X$, we can get an accepting run for the above word by modifying (1.1):

$$(q_0, v_0) \xrightarrow{\delta_0, \theta_0} (q_1, v_1) \xrightarrow{\delta_1, \theta_1} \dots (q_i, v_i) \xrightarrow{\Delta, \theta_j} (q_{j+1}, v'_{j+1}) \xrightarrow{\delta_{j+1}, \theta_{j+1}} \dots (q_{2^n}, v'_{2^n})$$

where $\Delta = \frac{(j-i)+1}{2^n+1}$. This gives us the contradiction. □

References

[BC05] Patricia Bouyer and Fabrice Chevalier. On conciseness of extensions of timed automata. *Journal of Automata, Languages and Combinatorics*, 10(4):393–405, 2005.

[BPDG98] Béatrice Bérard, Antoine Petit, Volker Diekert, and Paul Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2):145–182, 1998.