Real-Time Systems

Lecture 13: (Regions and) Zones

2012-07-05

Dr. Bernd Westphal

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- Location reachability decidability (by region construction)

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - Is constraint reachability decidable?
  - What's a zone? In contrast to a region?
  - Motivation for having zones?
  - What's a DBM? Who needs to know DBMs?
- **Content:**
  - The Constraint Reachability Problem
  - Zones
  - Difference Bound Matrices

The Constraint Reachability Problem

Recall: Putting It All Together

Let $\mathcal{A} = (L, B, X, I, E, \ell_{ini})$ be a timed automaton, $\ell \in L$ a location.

- $\mathcal{R}(\mathcal{A})$ can be constructed effectively.
- There are finitely many locations in L (by definition).
- There are finitely many regions by Lemma 4.28.
- So $Conf(\mathcal{R}(\mathcal{A}))$ is finite (by construction).
- It is decidable whether (C_{init} of $\mathcal{R}(\mathcal{A})$ is empty) or whether there exists a sequence

$$\langle \ell_{ini}, [\nu_{ini}] \rangle \xrightarrow{\alpha}_{R(\mathcal{A})} \langle \ell_1, [\nu_1] \rangle \xrightarrow{\alpha}_{R(\mathcal{A})} \dots \xrightarrow{\alpha}_{R(\mathcal{A})} \langle \ell_n, [\nu_n] \rangle$$

such that $\ell_n = \ell$ (reachability in graphs).

So we have

> **Theorem 4.33.** [*Decidability*]
> The location reachability problem for timed automata is **decidable**.

The Constraint Reachability Problem

- **Given:** A timed automaton $\mathcal{A}$, one of its control locations ℓ, and a clock constraint φ.
- **Question:** Is a configuration $\langle \ell, \nu \rangle$ **reachable** where $\nu \models \varphi$, i.e. is there a transition sequence of the form

$$\langle \ell_{ini}, \nu_{ini} \rangle \xrightarrow{\lambda_1} \langle \ell_1, \nu_1 \rangle \xrightarrow{\lambda_2} \langle \ell_2, \nu_2 \rangle \xrightarrow{\lambda_3} \dots \xrightarrow{\lambda_n} \langle \ell_n, \nu_n \rangle = \langle \ell, \nu \rangle$$

in the labelled transition system $\mathcal{T}(\mathcal{A})$ with $\nu \models \varphi$?

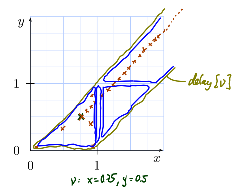
- **Note:** we just observed that $\mathcal{R}(\mathcal{A})$ loses some information about the clock valuations that are possible in/from a region.

> **Theorem 4.34.** The constraint reachability problem for timed automata is decidable.

The Delay Operation

- Let $[\nu]$ be a clock region.
- We set

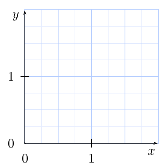
$$delay[\nu] = \{\nu' + t \mid \nu' \cong \nu \text{ and } t \in \mathsf{Time}\}.$$



The Delay Operation

- Let $[\nu]$ be a clock region.
- We set

$$delay[\nu] = \{\nu' + t \mid \nu' \cong \nu \text{ and } t \in \mathsf{Time}\}.$$



- **Note**: $delay[\nu]$ can be represented as a **finite** union of regions.

~~For example, with our two-clock example we have~~

~~$delay[x = y = 0] =$~~

Zones

(Presentation following [Fränzle, 2007])

Recall: Number of Regions

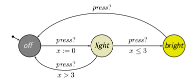
Lemma 4.28. Let X be a set of clocks, $c_x \in \mathbb{N}_0$ the maximal constant for each $x \in X$, and $c = \max\{c_x \mid x \in X\}$. Then

$$(2c+2)^{|X|} \cdot (4c+3)^{\frac{1}{2}|X|\cdot(|X|-1)}$$

is an **upper bound** on the **number of regions**.

$(2c+2)^{|X|} = 2\cdot 3+2 = 8$

- In the desk lamp controller,



all regions are reachable in $\mathcal{R}(\mathcal{L})$, but we convinced ourselves that it's **actually** only important whether $\nu(x) \in [0,3]$ or $\nu(x) \in (3,\infty)$.

So: seems there are even **equivalence classes** of undistinguishable regions.

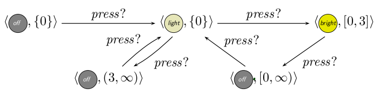
Wanted: Zones instead of Regions

region automaton

- In $\mathcal{R}(\mathcal{L})$ we have transitions:
  - $\langle light, \{0\}\rangle \xrightarrow{press?} \langle bright, \{0\}\rangle$, $\langle light, \{0\}\rangle \xrightarrow{press?} \langle bright, (0,1)\rangle$,
  - ...,
  - $\langle light, \{0\}\rangle \xrightarrow{press?} \langle bright, (2,3)\rangle$, $\langle light, \{0\}\rangle \xrightarrow{press?} \langle bright, \{3\}\rangle$
- Which seems to be a complicated way to write just:

$$\langle light, \{0\}\rangle \xrightarrow{press?} \langle bright, [0,3]\rangle$$

- Can't we **constructively** abstract $\mathcal{L}$ to:



What is a Zone?

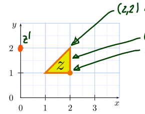
Definition. A (**clock**) **zone** is a set $z \subseteq (X \to \mathsf{Time})$ of valuations of clocks X such that there exists $\varphi \in \Phi(X)$ with

$$\nu \in z \text{ if and only if } \nu \models \varphi.$$

simple clock constraints (for simplicity $c \in \mathbb{N}$)

Example:

$\varphi' = x=0 \wedge y=2$



(2,2) is not in z

(2,1.5) is in z

(2,1) is in z

is a clock zone by

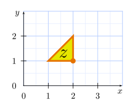
$$\varphi = (x \le 2) \wedge (x > 1) \wedge (y \ge 1) \wedge (y < 2) \wedge (x - y \ge 0)$$

What is a Zone?

Definition. A (**clock**) **zone** is a set $z \subseteq (X \to \mathsf{Time})$ of valuations of clocks X such that there exists $\varphi \in \Phi(X)$ with

$$\nu \in z \text{ if and only if } \nu \models \varphi.$$

Example:

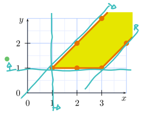


is a clock zone by

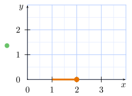
$$\varphi = (x \le 2) \wedge (x > 1) \wedge (y \ge 1) \wedge (y < 2) \wedge (x - y \ge 0)$$

- Note: Each clock constraint φ is a **symbolic representation** of a zone.
- But: There's no one-on-one correspondence between clock constraints and zones. The zone $z = \emptyset$ corresponds to $(x > 1 \wedge x < 1)$, $(x > 2 \wedge x < 2)$, ...

More Examples: Zone or Not?

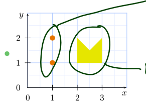


YES $(x \geq 1 \wedge y \geq 1 \wedge x - y \geq 0 \wedge x - y \leq 2)$



YES $(x \geq 1 \wedge x \leq 2 \wedge y = 0)$

not in $\Phi(X)$

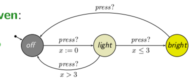


$x = 1 \wedge y = 2 \vee x = 1 \wedge y = 1$ ⇝ NO

NO

Zone-based Reachability

Given:

-  and initial configuration $\langle$off$, \{0\}\rangle$

Assume a function

$$\mathrm{Post}_e : (L \times \text{Zones}) \to (L \times \text{Zones})$$

an edge

such that $\mathrm{Post}_e(\langle \ell, z \rangle)$ yields the configuration $\langle \ell', z' \rangle$ such that

- zone z' denotes exactly those clock valuations ν'
- which are reachable from a configuration $\langle \ell, \nu \rangle$, $\nu \in z$,
- by taking edge $e = (\ell, \alpha, \varphi, Y, \ell') \in E$.

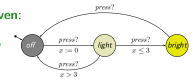
Then $\ell \in L$ is reachable in $\mathcal{A}$ if and only if

$$\mathrm{Post}_{e_n}(\dots(\mathrm{Post}_{e_1}(\langle \ell_{ini}, z_{ini} \rangle))\dots) = \langle \ell, z \rangle$$

for some $e_1, \dots, e_n \in E$.

Zone-based Reachability: In Other Words

Given:

-  and initial configuration $\langle$off$, \{0\}\rangle$

Wanted: A procedure to compute the set

- $\langle$light$, \{0\}\rangle$
- $\langle$bright$, [0, 3]\rangle$
- $\langle$off$, [0, \infty)\rangle$

- Set $R := \{\langle \ell_{ini}, z_{ini} \rangle\} \subset L \times$ Zones
- Repeat
  - pick
    - a pair $\langle \ell, z \rangle$ from R and
    - an edge $e \in E$ with source ℓ

    such that $\mathrm{Post}_e(\langle \ell, z \rangle)$ is not already subsumed by R
  - add $\mathrm{Post}_e(\langle \ell, z \rangle)$ to R

  until no more such $\langle \ell, z \rangle \in R$ and $e \in E$ are found.

Stocktaking: What's Missing?

- Set $R := \{\langle \ell_{ini}, z_{ini} \rangle\} \subset L \times$ Zones
- Repeat
  - pick
    - a pair $\langle \ell, z \rangle$ from R and
    - an edge $e \in E$ with source ℓ

    such that $\mathrm{Post}_e(\langle \ell, z \rangle)$ is not already **subsumed** by R
  - add $\mathrm{Post}_e(\langle \ell, z \rangle)$ to R

  until no more such $\langle \ell, z \rangle \in R$ and $e \in E$ are found.

Missing:

- Algorithm to effectively compute $\mathrm{Post}_e(\langle \ell, z \rangle)$ for given configuration $\langle \ell, z \rangle \in L \times$ Zones and edge $e \in E$.
- Decision procedure for whether configuration $\langle \ell', z' \rangle$ is **subsumed** by a given subset of $L \times$ Zones.

Note: Algorithm in general **terminates only if** we apply **widening** to zones, that is, roughly, to take maximal constants c_x into account (not in lecture).

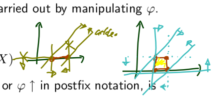
What is a Good "Post"?

- If z is given by a constraint $\varphi \in \Phi(X)$, then the zone component z' of $\mathrm{Post}_e(\ell, z) = \langle \ell', z' \rangle$ should also be a constraint from $\Phi(X)$. (Because sets of clock valuations are soo unhandily...)

Good news: the following operations can be carried out by manipulating φ.

- The **elapse time** operation:

  $$\uparrow : \Phi(X) \to \Phi(X)$$

  Given a constraint φ, the constraint $\uparrow(\varphi)$, or $\varphi \uparrow$ in postfix notation, is supposed to denote the set of clock valuations

  $$\{\nu + t \mid \nu \models \varphi, t \in \text{Time}\}.$$

  In other symbols: we **want**

  zone denoted by φ

  $$[\![\uparrow(\varphi)]\!] = [\![\varphi \uparrow]\!] = \{\nu + t \mid \nu \in [\![\varphi]\!], t \in \text{Time}\}.$$

  To this end: remove all upper bounds $x \leq c$, $x < c$ from φ and add diagonals.

Good News Cont'd

Good news: the following operations can be carried out by manipulating φ.

- **elapse time** $\varphi \uparrow$ with

  $$[\![\varphi \uparrow]\!] = \{\nu + t \mid \nu \models \varphi, t \in \text{Time}\}$$

- **zone intersection** $\varphi_1 \wedge \varphi_2$ with

  $$[\![\varphi_1 \wedge \varphi_2]\!] = \{\nu \mid \nu \models \varphi_1 \text{ and } \nu \models \varphi_2\}$$

- **clock hiding** $\exists x.\varphi$ with

  $$[\![\exists x.\varphi]\!] = \{\nu \mid \text{there is } t \in \text{Time such that } \nu[x := t] \models \varphi\}$$

- **clock reset** $\varphi[x := 0]$ with

  $$[\![\varphi[x := 0]]\!] = [\![x = 0 \wedge \exists x.\varphi]\!]$$

This is Good News...

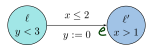
...because given $\langle \ell, z \rangle = \langle \ell, \varphi_0 \rangle$ and $e = (\ell, \alpha, \varphi, \{y_1, \dots, y_n\}, \ell') \in E$ we have

$$\mathrm{Post}_e(\langle \ell, z \rangle) = \langle \ell', \varphi_5 \rangle$$

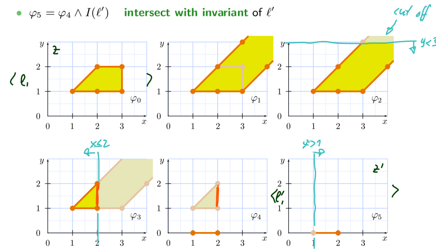
where

- $\varphi_1 = \varphi_0 \uparrow$
  let **time elapse** starting from φ_0: φ_1 represents all valuations reachable by waiting in ℓ for an arbitrary amount of time.
- $\varphi_2 = \varphi_1 \wedge I(\ell)$
  intersect with invariant of ℓ: φ_2 represents the reachable "good" valuations.
- $\varphi_3 = \varphi_2 \wedge \varphi$
  intersect with guard: φ_3 are the reachable good valuations where e is enabled.
- $\varphi_4 = \varphi_3[y_1 := 0] \dots [y_n := 0]$
  reset clocks: φ_4 are all possible outcomes of taking e from φ_3
- $\varphi_5 = \varphi_4 \wedge I(\ell')$
  intersect with invariant of ℓ': φ_5 are the "good" outcomes of taking e from φ_3

Example

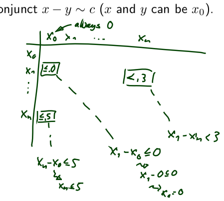


- $\varphi_1 = \varphi_0 \uparrow$ let **time elapse.**
- $\varphi_2 = \varphi_1 \wedge I(\ell)$ **intersect with invariant** of ℓ
- $\varphi_3 = \varphi_2 \wedge \varphi$ **intersect with guard**
- $\varphi_4 = \varphi_3[y_1 := 0] \dots [y_n := 0]$ **reset clocks**
- $\varphi_5 = \varphi_4 \wedge I(\ell')$ **intersect with invariant** of ℓ'



Difference Bound Matrices

- Given a finite set of clocks X, a **DBM** over X is a mapping
$$M : (X \dot\cup \{x_0\} \times X \dot\cup \{x_0\}) \to (\{<, \le\} \times \mathbb{Z}) \cup \{(<, \infty)\})$$
special symbol
- $M(x, y) = (\sim, c)$ encodes the conjunct $x - y \sim c$ (x and y can be x_0).



Difference Bound Matrices

- Given a finite set of clocks X, a **DBM** over X is a mapping
$$M : (X \dot\cup \{x_0\} \times X \dot\cup \{x_0\}) \to (\{<, \le\} \times \mathbb{Z} \cup \{(<, \infty)\})$$
- $M(x, y) = (\sim, c)$ encodes the conjunct $x - y \sim c$ (x and y can be x_0).
- If M and N are DBM encoding φ_1 and φ_2 (representing zones z_1 and z_2), then we can efficiently compute $M \uparrow$, $M \wedge N$, $M[x := 0]$ such that
  - all three are again DBM,
  - $M \uparrow$ encodes $\varphi_1 \uparrow$,
  - $M \wedge N$ encodes $\varphi_1 \wedge \varphi_2$, and
  - $M[x := 0]$ encodes $\varphi_1[x := 0]$.
- And there is a **canonical form** of DBM — canonisation of DBM can be done in cubic time (**Floyd-Warshall** algorithm).
- Thus: we can define our 'Post' on DBM, and let our algorithm run on DBM.

Pros and cons

- **Zone-based** reachability analysis usually is explicit wrt. discrete locations:
  - maintains a list of location/zone pairs or
  - maintains a list of location/DBM pairs
  - **confined wrt. size of discrete state space** (*)
  - **avoids blowup by number of clocks and size of clock constraints through symbolic representation of clocks**
- **Region-based** analysis provides a finite-state abstraction, amenable to finite-state symbolic MC
  - **less dependent on size of discrete state space**
  - **exponential in number of clocks**

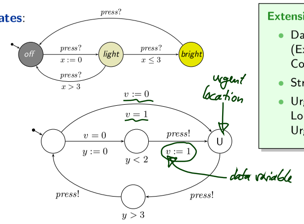
- H.-J. Peter et al.: fight (*)
- Sipi, Scholl: fully symbolic

Extended Timed Automata

Example (Partly Already Seen in Uppaal Demo)

Templates:

- $\mathcal{L}$:
- $\mathcal{U}$:



System:

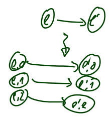
Extensions:

- Data Variables (Expressions, Constraints, Updates)
- Structuring
- Urgent/Committed Location, Urgent Channel

Data-Variables

- When modelling controllers as timed automata, it is sometimes desirable to have (local and shared) variables.
  E.g. count number of open doors, or intermediate positions of gas valve.
- Adding variables with **finite** range (possibly grouped into **finite** arrays) to any finite-state automata concept is straighforward:
  - If we have control locations $L_0 = \{\ell_1, \dots, \ell_n\}$,
  - and want to model, e.g., the valve range as a variable v with $\mathcal{D}(v) = \{0, \dots, 2\}$,
  - then just use $L = L_0 \times \mathcal{D}(v)$ as control locations, i.e. encode the current value of v in the control location, and consider updates of v in the $\xrightarrow{\lambda}$.

  L is still finite, so we still have a proper TA.



Data-Variables

- When modelling controllers as timed automata, it is sometimes desirable to have (local and shared) variables.
  E.g. count number of open doors, or intermediate positions of gas valve.
- Adding variables with **finite** range (possibly grouped into **finite** arrays) to any finite-state automata concept is straighforward:
  - If we have control locations $L_0 = \{\ell_1, \dots, \ell_n\}$,
  - and want to model, e.g., the valve range as a variable v with $\mathcal{D}(v) = \{0, \dots, 2\}$,
  - then just use $L = L_0 \times \mathcal{D}(v)$ as control locations, i.e. encode the current value of v in the control location, and consider updates of v in the $\xrightarrow{\lambda}$.

  L is still finite, so we still have a proper TA.
- But: writing $\xrightarrow{\lambda}$ is tedious.
- So: have variables as "first class citizens" and let compilers do the work.

- **Interestingly**, many examples in the literature live without variables: the more abstract the model is, i.e., the fewer information it keeps track of (e.g. in data variables), the easier the verification task.

Data Variables and Expressions

- Let $(v, w \in)\ V$ be a set of (integer) variables.
  $(\psi_{int} \in)\ \Psi(V)$: **integer expressions** over V using func. symb. $+, -, \dots$
  $(\varphi_{int} \in)\ \Phi(V)$: **integer** (or **data**) **constraints** over V
  using **integer expressions**, predicate symbols $=, <, \leq, \dots$, and boolean logical connectives.
- Let $(x, y \in)\ X$ be a set of clocks.
  $(\varphi \in)\ \Phi(X, V)$: (**extended**) **guards**, defined by

  $$\varphi ::= \varphi_{clk} \mid \varphi_{int} \mid \varphi_1 \wedge \varphi_2$$

  where $\varphi_{clk} \in \Phi(X)$ is a simple clock constraint (as defined before) and $\varphi_{int} \in \Phi(V)$ an **integer (or data) constraint**.

Examples: Extended guard or not extended guard? Why?

(a) $x < y \wedge v > 2$, (b) $x < y \vee v > 2$, (c) $v < 1 \vee v > 2$, (d) $x < v$

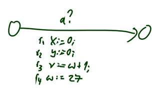
Modification or Reset Operation

- **New**: a **modification** or **reset operation** is

  $$x := 0, \qquad x \in X,$$

  or

  $$v := \psi_{int}, \qquad v \in V, \quad \psi_{int} \in \Psi(V).$$
- By $R(X, V)$ we denote the set of all resets.
- By $\vec{r}$ we denote a finite list $\langle r_1, \dots, r_n \rangle$, $n \in \mathbb{N}_0$, of reset operations $r_i \in R(X, V)$;
  $\langle\rangle$ is the empty list.
- By $R(X, V)^*$ we denote the set of all such lists of reset operations.



Modification or Reset Operation

- **New**: a **modification** or **reset operation** is

  $$x := 0, \qquad x \in X,$$

  or

  $$v := \psi_{int}, \qquad v \in V, \quad \psi_{int} \in \Psi(V).$$
- By $R(X, V)$ we denote the set of all resets.
- By $\vec{r}$ we denote a finite list $\langle r_1, \dots, r_n \rangle$, $n \in \mathbb{N}_0$, of reset operations $r_i \in R(X, V)$;
  $\langle\rangle$ is the empty list.
- By $R(X, V)^*$ we denote the set of all such lists of reset operations.

Examples: Modification or not? Why?

(a) $x := y$, (b) $x := v$, (c) $v := x$, (d) $v := w$, (e) $v := 0$

References

References

[Fränzle, 2007] Fränzle, M. (2007). Formale methoden eingebetteter systeme. Lecture, Summer Semester 2007, Carl-von-Ossietzky Universität Oldenburg.

[Olderog and Dierks, 2008] Olderog, E.-R. and Dierks, H. (2008). Real-Time Systems - Formal Specification and Automatic Verification. Cambridge University Press.