Modeling and Analysis of Hybrid Systems

5. Linear hybrid automata I

Prof. Dr. Erika Ábrahám

Informatik 2 - LuFG Theory of Hybrid Systems
RWTH Aachen University

Szeged, Hungary, 27 September - 06 October 2017

Literature

Alur et al.: The algorithmic analysis of hybrid systems

Theoretical Computer Science, 138(1):3–34, 1995

Linear terms, constraints and formulas

- A linear term e over a set Var of variables is of the form

$$e ::= c \mid c \cdot x \mid e + e$$

where $x \in Var$ is a variable and c stays for an integer (rational) constant.

Example: $x_1 + 2x_2 + 5x_3$ is a linear term over $Var = \{x_1, x_2, x_3\}$.

- A linear constraint t over Var is an (in)equality

$$t ::= e \sim 0$$

with $\sim \in \{>, \geq, =, \leq, <\}$ and e a linear term over Var.

Example: $x_1 + 2x_2 + (-2) \geq 0$ is a linear constraint over $Var = \{x_1, x_2\}$. We sometimes deviate from this normal form and write, e.g., $x_1 + 2x_2 \geq 2$.

Linear terms, constraints and formulas

- A conjunctive linear formula φ over Var is defined by the following grammar:

$$\varphi ::= t \mid \varphi \wedge \varphi \mid \exists x.\ \varphi$$

with t a linear constraint over Var and $x \in Var$ a variable. Let Φ_X be the set of all conjunctive linear formulas with free (non-quantified) variables from $X \subseteq Var$.

Example: $\exists t.\ \exists x_1.\ x_1 \geq 0 \wedge x_1' = x_1 + 2t \wedge t \geq 0$ is a conjunctive linear formula over $\{x_1, x_1', t\}$, with free (non-quantified) variables $\{x_1'\}$.

Regions

- A region over Var is a pair $R = (l, \varphi) \in Loc \times \Phi_{Var}$ of a location and a conjunctive linear formula with free variables from Var.

  Example: $(l, \exists t.\ \exists x_1.\ x_1 \geq 0 \wedge x_1' = x_1 + 2t \wedge t \geq 0)$ is a region over $\{x_1'\}$.
- The intersection of two regions $R^1 = (l_1, \varphi_1)$ and $R^2 = (l_2, \varphi_2)$ from $Loc \times \Phi_{Var}$ is defined as $R^1 \hat{\cap} R^2 = \emptyset$ if $l_1 \neq l_2$ and

$$R^1 \hat{\cap} R^2 = (l_1, \varphi_1 \wedge \varphi_2)$$

  otherwise.
- The intersection of two sets of regions $P^1, P^2 \subseteq Loc \times \Phi_{Var}$ is defined as

$$P^1 \hat{\cap} P^2 = \{(l, \varphi_1 \wedge \varphi_2) \mid (l, \varphi_1) \in P^1, (l, \varphi_2) \in P^2\}\ .$$

- We define other operations on regions like union etc. similarly, and extend them to sets of regions the natural way.

Substitution

- Assume a set Var of variables, a linear formula $\varphi \in \Phi_{Var}$ over Var, a variable $x \in Var$, and a linear term e over Var. The substitution $\varphi[e/x]$ replaces each free occurrence of x in φ by e.

  Example: $(x + 2y \leq 0)[5/y] = x + 2 \cdot 5 \leq 0$

  Example: $(\exists y.\ x + 2y \leq 0)[5/y] = x + 2y \leq 0$
- We write $\varphi[e_1, \ldots, e_n/x_1, \ldots, x_n]$ for the simultaneous substitution of e_i for x_i, $i = 1, \ldots, n$.
- For $Var = \{x_1, \ldots, x_n\}$ we will also use a primed variable set $Var' = \{x'_1, \ldots, x'_n\}$ and write short $\varphi[Var'/Var]$ for $\varphi[x'_1, \ldots, x'_n/x_1, \ldots, x_n]$.

Linear terms, constraints and formulas

The semantics of linear terms, constraints and formulas over $Var = \{x_1, \ldots, x_n\}$ in the context of a valuation $\nu \in V_{Var}$ (i.e., $\nu : Var \to \mathbb{R}$) is as usual (we use the same notation for the syntax and the semantics of constants and operators):

$$
\begin{array}{lcl}
\nu(c) & \equiv & c \\
\nu(c \cdot x) & \equiv & c \cdot \nu(x) \\
\nu(e_1 + e_2) & \equiv & \nu(e_1) + \nu(e_2) \\
\nu(e \sim 0) & \equiv & \nu(e) \sim 0 \\
\nu(\varphi_1 \wedge \varphi_2) & \equiv & \nu(\varphi_1) \textit{ and } \nu(\varphi_2) \\
\nu(\exists x.\ \varphi) & \equiv & \textit{exists } v \in \mathbb{R} \textit{ such that } \nu(\varphi[v/x]) \textit{ holds}
\end{array}
$$

Linear terms, constraints and formulas

The solution set $Sat(\varphi)$ of a linear formula φ over Var is the set of all valuations $\nu \in V_{Var}$ that make φ true:

$$Sat(\varphi) = \{\nu \in V_{Var} \mid \nu \models \varphi\}$$

The solution set $Sat(R)$ of a region $R = (l, \varphi) \in Loc \times \Phi_{Var}$ over Loc and Var is

$$Sat((l, \varphi)) = \{(l, \nu) \in Loc \times V_{Var} \mid \nu \in Sat(\varphi)\}$$

The solution set $Sat(P)$ of a set of regions P over Loc and Var is

$$Sat(P) = \cup_{R \in P} Sat(R) \ .$$

Two region sets $P_1, P_2 \subseteq Loc \times \Phi_{Var}$ are equivalent, written $P_1 \hat{=} P_2$, iff

$$Sat(P_1) = Sat(P_2) \ .$$

We define similarly the inclusion $P_1 \hat{\subseteq} P_2$ iff

$$Sat(P_1) \subseteq Sat(P_2) \ .$$

Linear hybrid automata I

A linear hybrid automaton is a hybrid automaton $\mathcal{H} = (Loc, Var, Lab, Edge, Act, Inv, Init)$ which can be represented by a tuple $\mathtt{H} = (Loc, Var, Lab, \mathtt{Edge}, \mathtt{Act}, \mathtt{Inv}, \mathtt{Init})$ satisfying the following:

- The finite set $\mathtt{Edge} \subseteq Loc \times \Phi_{Var} \times \Phi_{Var \cup Var'} \times Loc$ defines the set of edges $Edge = \{(l, a, \mu_e, l') \mid e = (l, a, \mathtt{Guard}_e, \mathtt{Reset}_e, l') \in \mathtt{Edge}\}$, where the transition relation μ_e is given as

  $$\mu_e = \{(\nu, \nu') \in V_{Var} \times V_{Var} \mid \nu \in Sat(\mathtt{Guard}_e) \textit{ and } (\nu \oplus \nu') \in Sat(\mathtt{Reset}_e)\}$$

  with $\nu \oplus \nu' \in V_{Var \cup Var'}$ such that $(\nu \oplus \nu')(x) = \nu(x)$ for $x \in Var$ and $(\nu \oplus \nu')(x') = \nu'(x)$ for $x' \in Var'$.

- The set $\mathtt{Act} \subseteq Loc \times \Phi_{Var}$ contains for each location $l \in Loc$ exactly one region $(l, \mathtt{Act}_l) \in \mathtt{Act}$ whose second component is of the form $\mathtt{Act}_l = \bigwedge_{i=1}^{n} x_i + k_{l,i}^{lower} t \leq x_i' \wedge x_i' \leq x_i + k_{l,i}^{upper} t$, defining the activities

  $$Act(l) = \{f : \mathbb{R}_{\geq 0} \to V_{Var} \mid \dot{f}_{x_i} \in [k_{l,i}^{lower}, k_{l,i}^{upper}] \textit{ for all } i \in \{1, \ldots, n\}\} ,$$

  where $f_{x_i} : \mathbb{R}_{\geq 0} \to \mathbb{R}$ with $f_{x_i}(t) = f(t)(x_i)$ for all $t \in \mathbb{R}_{\geq 0}$.

Linear hybrid automata I (cont.)

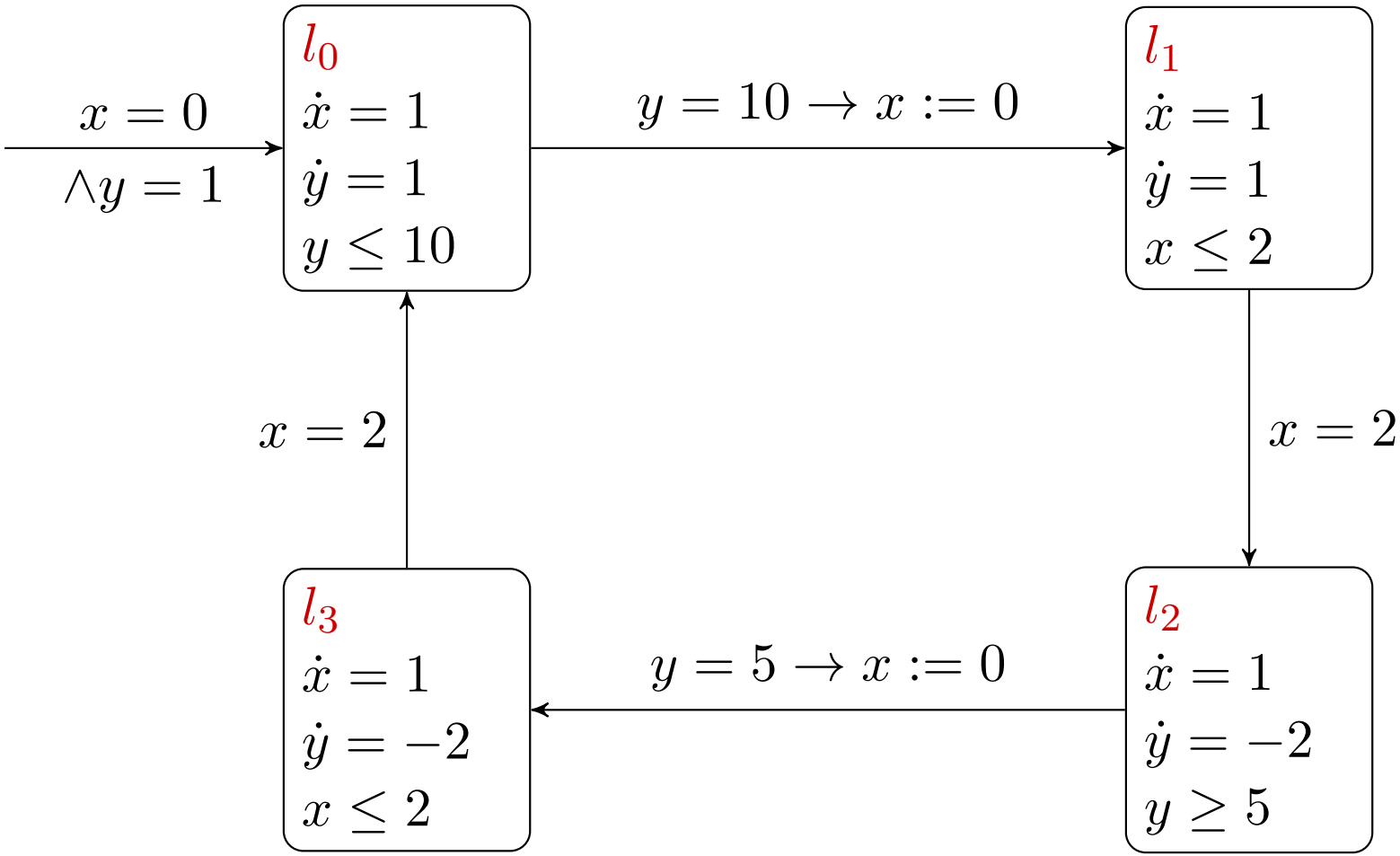
- The set $\texttt{Inv} \subseteq Loc \times \Phi_{Var}$ contains for each location $l \in Loc$ exactly one region $(l, \texttt{Inv}_l) \in \texttt{Inv}$ such that

$$Inv(l) = Sat(\texttt{Inv}_l) \ .$$

- $\texttt{Init} \subseteq Loc \times \Phi_{Var}$ is a finite set of regions specifying the initial states by

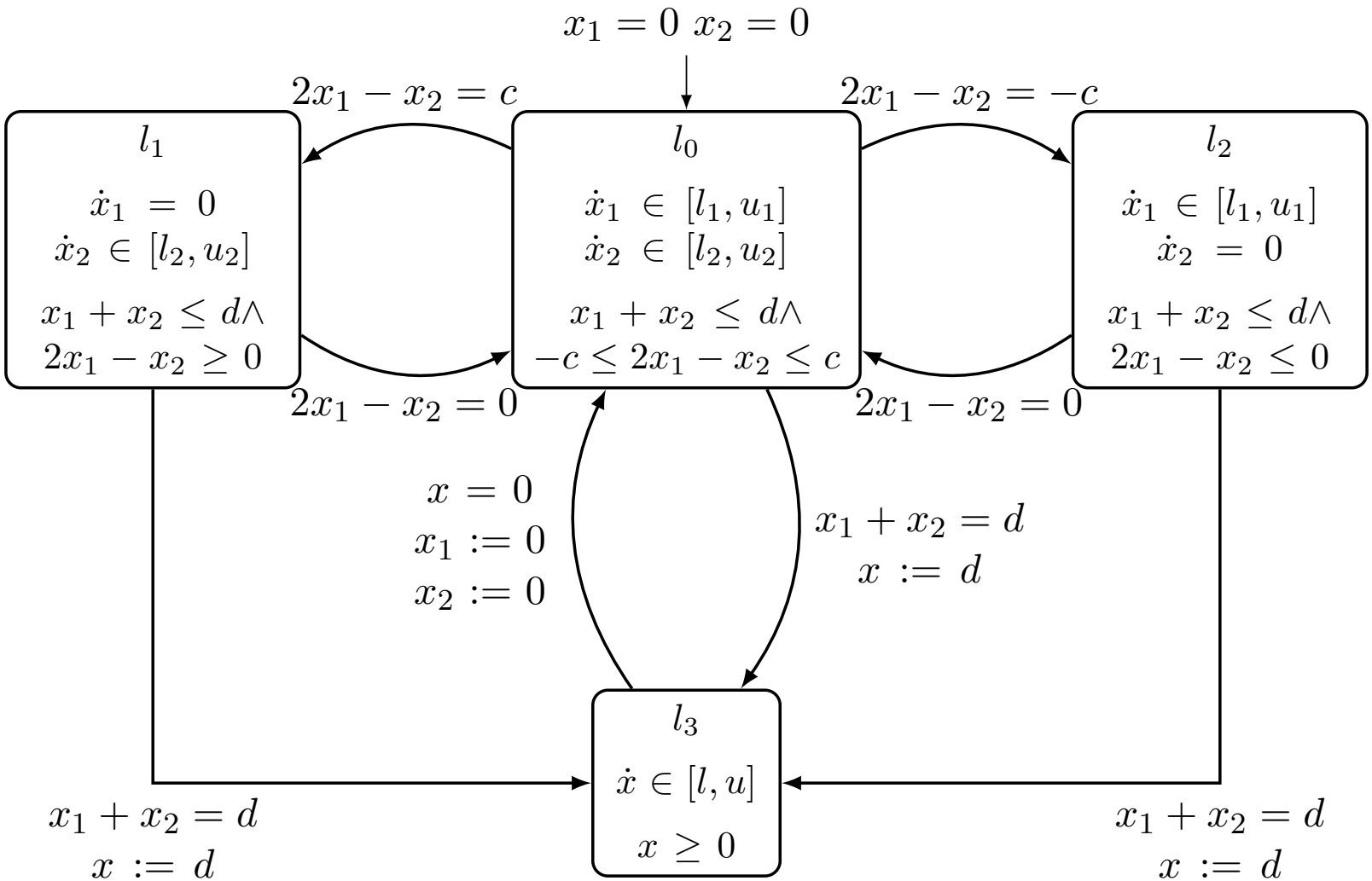
$$Init = \bigcup_{(l,\varphi) \in \texttt{Init}} \{(l, \nu) \mid \nu \in Sat(\varphi)\} \ .$$

Water-level monitor



Mixer of fluids

$$0 < l_1 < u_1,\ 0 < l_2 < u_2,\ l \leq u < 0,\ d > 0,\ c > 0$$



Reminder: Semantics of hybrid automata

$$\frac{(l, a, \mu, l') \in Edge \quad (\nu, \nu') \in \mu \quad \nu' \in Inv(l')}{(l, \nu) \xrightarrow{a} (l', \nu')} \text{ Rule } \texttt{Discrete}$$

$$\frac{\begin{array}{l} f \in Act(l) \quad f(0) = \nu \quad f(t) = \nu' \\ t \geq 0 \quad \forall 0 \leq t' \leq t . f(t') \in Inv(l) \end{array}}{(l, \nu) \xrightarrow{t} (l, \nu')} \text{ Rule } \texttt{Time}$$

Forward analysis

- Given a set of initial states $Init \subseteq \Sigma$, we want to compute the set of all states which are reachable from $Init$:

$$Reach^{+}(Init) = \{\sigma' \in \Sigma \mid \exists \sigma \in Init.\ \sigma \rightarrow^{*} \sigma'\} \ .$$

- Given a set of initial states $Init \subseteq \Sigma$, we want to compute the set of all states which are reachable from $Init$:

$$Reach^{+}(Init) = \{\sigma' \in \Sigma \mid \exists \sigma \in Init.\ \sigma \rightarrow^{*} \sigma'\} \ .$$

- More specifically, we want to check whether the reachable region intersects with a set of bad (unsafe) states.

Forward reachability analysis

```
method forward_reach(
    hybrid automaton representation H = (Loc, Var, Lab, Edge, Act, Inv, Init),
    region set P^bad) {
  //start from the time successors of initial regions
  P^0 := T+(Init ∩̂ Inv);
  if (Sat(P^0 ∩̂ P^bad) ≠ ∅) return unsafe;
  i := 0;
  while P^i ≠ ∅ {
    //compute P^{i+1} := T+(D+(P^i))
    P^{i+1} := ∅;
    for each (R = (l, φ) ∈ P^i) {
      for each e = (l, ..., l') ∈ Edge {
        R' := T+_{l'}(D+_e(R));
        if (Sat({R'} ∩̂ P^bad) ≠ ∅) return unsafe;
        else if (not {R'} ⊆̂ ⋃̂_{j=0}^{i} P^j)
          P^{i+1} := P^{i+1} ∪ {R';
      }
    }
    i := i+1;
  }
  return safe;
}
```

One-step reachability under time steps

- We define the forward time closure $\mathcal{T}_l^+(\varphi)$ of a formula $\varphi \in \Phi_{Var}$ at $l \in Loc$ as

$$\mathcal{T}_l^+(\varphi) = \exists x_{pre}.\ \exists t.\ t \geq 0 \wedge \varphi[x_{pre}/x] \wedge \mathtt{Act}_l[x_{pre}, x/x, x'] \wedge \mathtt{Inv}_l$$

- Region $R = (l, \varphi) \in Loc \times \Phi_{Var}$:

$$\mathcal{T}_l^+(R) = (l, \mathcal{T}_l^+(\varphi))$$

- Set of regions $P \subseteq Loc \times \Phi_{Var}$:

$$\mathcal{T}^+(P) = \{\mathcal{T}_l^+(R) \mid R = (l, \varphi) \in P\}$$

One-step reachability under discrete steps

- We define the postcondition $\mathcal{D}_e^+(\varphi)$ of a formula $\varphi \in \Phi_{Var}$ with respect to an edge $e = (l, \texttt{Guard}_e, \texttt{Reset}_e, l')$ as

$$\mathcal{D}_e^+(\varphi) = \exists x_{pre}.\ \varphi[x_{pre}/x] \wedge \texttt{Guard}_e[x_{pre}/x] \wedge \texttt{Reset}_e[x_{pre}, x/x, x'] \wedge \texttt{Inv}_{l'}.$$

- Region $R = (l, \varphi) \in Loc \times \Phi_{Var}$:

$$\mathcal{D}_e^+(R) = (l', \mathcal{D}_e^+(\varphi))$$

- Set of regions $P \subseteq Loc \times \Phi_{Var}$:

$$\mathcal{D}^+(P) = \{\mathcal{D}_e^+(R) \mid R = (l, \varphi) \in P,\ e = (l, \texttt{Guard}_e, \texttt{Reset}_e, l') \in \texttt{Edge}\}$$

Backward analysis

- Given a set of target states $B \subseteq \Sigma$, we want to compute the set of all states from which a state in B is reachable:

$$Reach^{-}(B) = \{\sigma \in \Sigma \mid \exists \sigma' \in B.\ \sigma \rightarrow^{*} \sigma'\}\ .$$

- Given a set of target states $B \subseteq \Sigma$, we want to compute the set of all states from which a state in B is reachable:

$$Reach^{-}(B) = \{\sigma \in \Sigma \mid \exists \sigma' \in B.\ \sigma \rightarrow^{*} \sigma'\}\ .$$

- More specifically, we want to check whether the set of backward reachable states intersects with a set of initial states.

Backward reachability analysis

```
method backward_reach() {
  i := 0;
  P^0 := {T_l^-(R) | R = (l, φ) ∈ P^bad ∩̂ Inv}; //time predec. of bad regions
  if (Sat(P^0 ∩̂ Init) ≠ ∅) return unsafe;
  while P^i ≠ ∅ {
    //compute time predecessors of
    //discrete predecessors of all regions from P^i
    for each (R = (l, φ) ∈ P^i) {
      for each e = (l', ..., l) ∈ Edge {
        R' := T_l'^-(D_e^-(R));
        if (Sat({R'} ∩̂ Init) ≠ ∅) return unsafe;
        else if (not {R'} ⊆̂ ⋃_{j=0}^{i} P^j)
          P^{i+1} := P^{i+1} ∪ {R'};
      }
    }
    i := i+1;
  }
  return safe;
}
```

One-step reachability under time steps

- We define the backward time closure $\mathcal{T}_l^-(\varphi)$ of a formula $\varphi \in \Phi_{Var}$ at $l \in Loc$ as

$$\mathcal{T}_l^-(\varphi) = \exists x_{post}.\ \exists t.\ t \geq 0 \wedge \varphi[x_{post}/x] \wedge \mathtt{Act}_l[x, x_{post}/x, x'] \wedge \mathtt{Inv}_l\ .$$

- Region $R = (l, \varphi) \in Loc \times \Phi_{Var}$:

$$\mathcal{T}_l^-(R) = (l, \mathcal{T}_l^-(\varphi))$$

- Set of regions $P \subseteq Loc \times \Phi_{Var}$:

$$\mathcal{T}^-(P) = \{\mathcal{T}_l^-(R) \mid R = (l, \varphi) \in P\}$$

One-step reachability under discrete steps

- We define the precondition $\mathcal{D}_e^-(\varphi)$ of a formula $\varphi \in \Phi_{Var}$ with respect to an edge $e = (l, \texttt{Guard}_e, \texttt{Reset}_e, l')$ as

$$\mathcal{D}_e^-(\varphi) = \exists x_{post}.\ \varphi[x_{post}/x] \wedge \texttt{Guard}_e \wedge \texttt{Reset}_e[x, x_{post}/x, x'] \wedge \texttt{Inv}_l.$$

- Region $R = (l', \varphi) \in Loc \times \Phi_{Var}$:

$$\mathcal{D}_e^-(R) = (l, \mathcal{D}_e^-(\varphi))$$

- Set of regions $P \subseteq Loc \times \Phi_{Var}$:

$$\mathcal{D}^-(P) = \{\mathcal{D}_e^-(R) \mid R = (l', \varphi) \in P,\ e = (l, \texttt{Guard}_e, \texttt{Reset}_e, l') \in \texttt{Edge}\}$$

Quantifier elimination

- Problem 1: Formula size increases steeply
- Problem 2: In the presence of quantifiers, computing operations (inclusion, intersection etc.) on regions is non-trivial
- Solution: Therefore, we eliminate quantifiers: given a conjunctive linear formula $\exists x.\ \varphi \in \Phi_X$, we compute another conjunctive linear formula $\varphi' \in \Phi_{X \setminus \{x\}}$ that does not contain x such that

$$\mathit{Sat}(\exists x.\ \varphi) = \mathit{Sat}(\varphi') \ .$$

- Technique: Gauß and Fourier-Motzkin variable elimination

Linear real arithmetic: Gauß method for equations

- Assume that φ is of the form $\exists x_n.\ \varphi'' \wedge \sum_{k=1}^{n} a_k \cdot x_k = b$ with $a_n \neq 0$.

Linear real arithmetic: Gauß method for equations

- Assume that φ is of the form $\exists x_n.\ \varphi'' \wedge \sum_{k=1}^{n} a_k \cdot x_k = b$ with $a_n \neq 0$.

$$\Rightarrow \quad a_n \cdot x_n \;=\; b - \sum_{k \in \{1,\ldots,n-1\}} a_k \cdot x_k$$

Linear real arithmetic: Gauß method for equations

- Assume that φ is of the form $\exists x_n.\ \varphi'' \wedge \sum_{k=1}^{n} a_k \cdot x_k = b$ with $a_n \neq 0$.

$$\Rightarrow \quad a_n \cdot x_n \;=\; b - \sum_{k \in \{1,\ldots,n-1\}} a_k \cdot x_k$$

$$\Rightarrow \quad x_n \;=\; \frac{b}{a_n} - \sum_{k \in \{1,\ldots,n-1\}} \frac{a_k}{a_n} \cdot x_k := \beta$$

Linear real arithmetic: Gauß method for equations

- Assume that φ is of the form $\exists x_n.\ \varphi'' \wedge \sum_{k=1}^{n} a_k \cdot x_k = b$ with $a_n \neq 0$.

$$\Rightarrow \quad a_n \cdot x_n \;=\; b - \sum_{k \in \{1,\ldots,n-1\}} a_k \cdot x_k$$

$$\Rightarrow \quad x_n \;=\; \frac{b}{a_n} - \sum_{k \in \{1,\ldots,n-1\}} \frac{a_k}{a_n} \cdot x_k := \beta$$

- Replace x_n by β in φ''.

Linear real arithmetic: Gauß method for equations

- Assume that φ is of the form $\exists x_n.\ \varphi'' \wedge \sum_{k=1}^{n} a_k \cdot x_k = b$ with $a_n \neq 0$.

$$\Rightarrow \quad a_n \cdot x_n \;=\; b - \sum_{k \in \{1,\ldots,n-1\}} a_k \cdot x_k$$

$$\Rightarrow \quad x_n \;=\; \frac{b}{a_n} - \sum_{k \in \{1,\ldots,n-1\}} \frac{a_k}{a_n} \cdot x_k := \beta$$

- Replace x_n by β in φ''.
- This substitutiton leads to an equisatisfiable problem in $n-1$ variables:

$$\mathit{Sat}(\exists x_n.\ \varphi'' \wedge \sum_{k=1}^{n} a_k \cdot x_k = b) \quad = \quad \mathit{Sat}(\varphi''[\beta/x_n]) \quad (\text{for } a_n \neq 0).$$

Linear real arithmetic: Fourier-Motzkin for inequalities

- Discovered in 1826 by Fourier, re-discovered by Motzkin in 1936
- Given:

$$\exists x_n. \bigwedge_{1 \leq i \leq m} \sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i$$

Linear real arithmetic: Fourier-Motzkin for inequalities

- Discovered in 1826 by Fourier, re-discovered by Motzkin in 1936
- Given:

$$\exists x_n. \bigwedge_{1 \leq i \leq m} \sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i$$

- For a variable x_n, we can partition the constraints according to the coefficient a_{in}:
  - $a_{in} > 0$: upper bound β_i on x_n
  - $a_{in} < 0$: lower bound β_i on x_n

Linear real arithmetic: Fourier-Motzkin for inequalities

- Discovered in 1826 by Fourier, re-discovered by Motzkin in 1936
- Given:

$$\exists x_n. \bigwedge_{1 \leq i \leq m} \sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i$$

- For a variable x_n, we can partition the constraints according to the coefficient a_{in}:
  - $a_{in} > 0$: upper bound β_i on x_n
  - $a_{in} < 0$: lower bound β_i on x_n

- Idea: Exists satisfying value for variable x_j iff none of the intervals defined by lower-upper-bound-pairs on x_j is empty

Linear real arithmetic: Fourier-Motzkin for inequalities

- Discovered in 1826 by Fourier, re-discovered by Motzkin in 1936
- Given:

$$\exists x_n. \bigwedge_{1 \leq i \leq m} \sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i$$

- For a variable x_n, we can partition the constraints according to the coefficient a_{in}:
  - $a_{in} > 0$: upper bound β_i on x_n
  - $a_{in} < 0$: lower bound β_i on x_n

- Idea: Exists satisfying value for variable x_j iff none of the intervals defined by lower-upper-bound-pairs on x_j is empty

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i$$

Linear real arithmetic: Fourier-Motzkin for inequalities

- Discovered in 1826 by Fourier, re-discovered by Motzkin in 1936
- Given:

$$\exists x_n. \bigwedge_{1 \leq i \leq m} \sum_{1 \leq j \leq n} a_{ij} x_j \leq b_i$$

- For a variable x_n, we can partition the constraints according to the coefficient a_{in}:
  - $a_{in} > 0$: upper bound β_i on x_n
  - $a_{in} < 0$: lower bound β_i on x_n

- Idea: Exists satisfying value for variable x_j iff none of the intervals defined by lower-upper-bound-pairs on x_j is empty

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \leq b_i \quad \Rightarrow \quad a_{in} \cdot x_n \leq b_i - \sum_{j=1}^{n-1} a_{ij} \cdot x_j$$

Linear real arithmetic: Fourier-Motzkin for inequalities

- Discovered in 1826 by Fourier, re-discovered by Motzkin in 1936
- Given:

$$\exists x_n. \bigwedge_{1 \le i \le m} \sum_{1 \le j \le n} a_{ij} x_j \le b_i$$

- For a variable x_n, we can partition the constraints according to the coefficient a_{in}:
  - $a_{in} > 0$: upper bound β_i on x_n
  - $a_{in} < 0$: lower bound β_i on x_n

- Idea: Exists satisfying value for variable x_j iff none of the intervals defined by lower-upper-bound-pairs on x_j is empty

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \le b_i \quad \Rightarrow \quad a_{in} \cdot x_n \le b_i - \sum_{j=1}^{n-1} a_{ij} \cdot x_j$$

$$(a) \quad \overset{a_{in}>0}{\Rightarrow} \quad x_n \le \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j \quad =: \beta_l \quad \text{upper bound}$$

$$(b) \quad \overset{a_{in}<0}{\Rightarrow} \quad x_n \ge \frac{b_i}{a_{in}} - \sum_{j=1}^{n-1} \frac{a_{ij}}{a_{in}} \cdot x_j \quad =: \beta_u \quad \text{lower bound}$$

Example for upper and lower bounds

Category for x_1?

$$
\begin{aligned}
&(1) \quad x_1 - x_2 \leq 0 \\
&(2) \quad x_1 - x_3 \leq 0 \\
&(3) \quad -x_1 + x_2 + 2x_3 \leq 0 \\
&(4) \quad -x_3 \leq -1
\end{aligned}
$$

Example for upper and lower bounds

$$
\begin{aligned}
&(1) \quad x_1 - x_2 \leq 0 \\
&(2) \quad x_1 - x_3 \leq 0 \\
&(3) \quad -x_1 + x_2 + 2x_3 \leq 0 \\
&(4) \quad -x_3 \leq -1
\end{aligned}
$$

Category for x_1?

Upper bound

Example for upper and lower bounds

		Category for x_1?
(1)	$x_1 - x_2 \leq 0$	Upper bound
(2)	$x_1 - x_3 \leq 0$	Upper bound
(3)	$-x_1 + x_2 + 2x_3 \leq 0$	
(4)	$-x_3 \leq -1$	

Example for upper and lower bounds

		Category for x_1?
(1)	$x_1 - x_2 \leq 0$	Upper bound
(2)	$x_1 - x_3 \leq 0$	Upper bound
(3)	$-x_1 + x_2 + 2x_3 \leq 0$	Lower bound
(4)	$-x_3 \leq -1$	

Example for upper and lower bounds

		Category for x_1?
(1)	$x_1 - x_2 \leq 0$	Upper bound
(2)	$x_1 - x_3 \leq 0$	Upper bound
(3)	$-x_1 + x_2 + 2x_3 \leq 0$	Lower bound
(4)	$-x_3 \leq -1$	No bound

Eliminating unbounded variables

- Iteratively remove variables that are not bounded in both ways (and all the constraints that use them).
- The new problem has a solution iff the old problem has one!

$$\begin{aligned} 8x &\geq 7y \\ x &\geq 3 \\ y &\geq z \\ z &\geq 10 \\ 20 &\geq z \end{aligned}$$

Eliminating unbounded variables

- Iteratively remove variables that are not bounded in both ways (and all the constraints that use them).
- The new problem has a solution iff the old problem has one!

$$
\begin{array}{rcl}
\cancel{8x} & \cancel{\geq} & \cancel{7y} \\
\cancel{x} & \cancel{\geq} & \cancel{3} \\
y & \geq & z \\
z & \geq & 10 \\
20 & \geq & z
\end{array}
$$

Eliminating unbounded variables

- Iteratively remove variables that are not bounded in both ways (and all the constraints that use them).
- The new problem has a solution iff the old problem has one!

$$
\begin{array}{rcl}
\cancel{8x} & \cancel{\geq} & \cancel{7y} \\
\cancel{x} & \cancel{\geq} & \cancel{3} \\
y & \geq & z \\
z & \geq & 10 \\
20 & \geq & z
\end{array}
\quad \longrightarrow \quad
\begin{array}{rcl}
y & \geq & z \\
z & \geq & 10 \\
20 & \geq & z
\end{array}
$$

Eliminating unbounded variables

- Iteratively remove variables that are not bounded in both ways (and all the constraints that use them).
- The new problem has a solution iff the old problem has one!

$$
\begin{array}{rcl}
\cancel{8x} & \cancel{\geq} & \cancel{7y} \\
\cancel{x} & \cancel{\geq} & \cancel{3} \\
y & \geq & z \\
z & \geq & 10 \\
20 & \geq & z
\end{array}
\quad \longrightarrow \quad
\begin{array}{rcl}
\cancel{y} & \cancel{\geq} & \cancel{z} \\
z & \geq & 10 \\
20 & \geq & z
\end{array}
$$

Eliminating unbounded variables

- Iteratively remove variables that are not bounded in both ways (and all the constraints that use them).
- The new problem has a solution iff the old problem has one!

$$\begin{array}{rcl} \cancel{8x} & \cancel{\geq} & \cancel{7y} \\ \cancel{x} & \cancel{\geq} & \cancel{3} \\ y & \geq & z \\ z & \geq & 10 \\ 20 & \geq & z \end{array} \longrightarrow \begin{array}{rcl} \cancel{y} & \cancel{\geq} & \cancel{z} \\ z & \geq & 10 \\ 20 & \geq & z \end{array} \longrightarrow \begin{array}{rcl} z & \geq & 10 \\ 20 & \geq & z \end{array}$$

Fourier-Motzkin variable elimination

- For each pair of a lower bound β_l and an upper bound β_u, we have

$$\beta_l \quad \leq \quad x_n \quad \leq \quad \beta_u$$

Fourier-Motzkin variable elimination

- For each pair of a lower bound β_l and an upper bound β_u, we have

$$\beta_l \quad \leq \quad x_n \quad \leq \quad \beta_u$$

- For each such pair, add the constraint

$$\beta_l \quad \leq \quad \beta_u$$

Fourier-Motzkin: Example

Category for x_1?

$$
\begin{aligned}
&(1) \quad x_1 - x_2 \leq 0 \\
&(2) \quad x_1 - x_3 \leq 0 \\
&(3) \quad -x_1 + x_2 + 2x_3 \leq 0 \\
&(4) \quad -x_3 \leq -1
\end{aligned}
$$

Fourier-Motzkin: Example

(1)	$x_1 - x_2 \leq 0$	Upper bound
(2)	$x_1 - x_3 \leq 0$	Upper bound
(3)	$-x_1 + x_2 + 2x_3 \leq 0$	Lower bound
(4)	$-x_3 \leq -1$	
		eliminate x_1

Category for x_1?

Fourier-Motzkin: Example

		Category for x_1?
(1)	$x_1 - x_2 \leq 0$	Upper bound
(2)	$x_1 - x_3 \leq 0$	Upper bound
(3)	$-x_1 + x_2 + 2x_3 \leq 0$	Lower bound
(4)	$-x_3 \leq -1$	
	---	eliminate x_1
(5)	$2x_3 \leq 0$ (from 1,3)	

Fourier-Motzkin: Example

			Category for x_1?
(1)	$x_1 - x_2 \leq 0$		Upper bound
(2)	$x_1 - x_3 \leq 0$		Upper bound
(3)	$-x_1 + x_2 + 2x_3 \leq 0$		Lower bound
(4)	$-x_3 \leq -1$		
	---		eliminate x_1
(5)	$2x_3 \leq 0$	(from 1,3)	
(6)	$x_2 + x_3 \leq 0$	(from 2,3)	

Fourier-Motzkin: Example

Category for x_1?

(1) ~~$x_1 \quad x_2 \leq 0$~~

(2) ~~$x_1 \quad x_3 \leq 0$~~

(3) ~~$x_1 + x_2 + 2x_3 \leq 0$~~

(4) $-x_3 \leq -1$

---

eliminate x_1

(5) $2x_3 \leq 0$ (from 1,3)

(6) $x_2 + x_3 \leq 0$ (from 2,3)

Fourier-Motzkin: Example

Category for x_1?

~~(1) $x_1 \quad x_2 \leq 0$~~

~~(2) $x_1 \quad x_3 \leq 0$~~

~~(3) $x_1 + x_2 + 2x_3 \leq 0$~~

(4) $-x_3 \leq -1$

---

eliminate x_1

(5) $2x_3 \leq 0$ (from 1,3)

(6) $x_2 + x_3 \leq 0$ (from 2,3)

---

we eliminate x_3

Fourier-Motzkin: Example

Category for x_1?

(1) ~~$x_1 \quad x_2 \leq 0$~~

(2) ~~$x_1 \quad x_3 \leq 0$~~

(3) ~~$x_1 + x_2 + 2x_3 \leq 0$~~

(4) $-x_3 \leq -1$ — Lower bound

---

eliminate x_1

(5) $2x_3 \leq 0$ (from 1,3) — Upper bound

(6) $x_2 + x_3 \leq 0$ (from 2,3) — Upper bound

---

we eliminate x_3

Fourier-Motzkin: Example

Category for x_1?

(1) ~~$x_1 \quad x_2 \leq 0$~~

(2) ~~$x_1 \quad x_3 \leq 0$~~

(3) ~~$x_1 + x_2 + 2x_3 \leq 0$~~

(4) $-x_3 \leq -1$ — Lower bound

---

eliminate x_1

(5) $2x_3 \leq 0$ (from 1,3) — Upper bound

(6) $x_2 + x_3 \leq 0$ (from 2,3) — Upper bound

---

we eliminate x_3

(7) $1 \leq 0$ (from 4,5)

$\rightarrow$ Contradiction (the system is UNSAT)

Complexity

- Worst-case complexity:

$$m \rightarrow m^2$$

Complexity

- Worst-case complexity:

$$m \rightarrow m^2 \rightarrow (m^2)^2$$

Complexity

- Worst-case complexity:

$$m \rightarrow m^2 \rightarrow (m^2)^2 \rightarrow \ldots \rightarrow m^{2^n}$$

Complexity

- Worst-case complexity:

$$m \rightarrow m^2 \rightarrow (m^2)^2 \rightarrow \ldots \rightarrow m^{2^n}$$

- Heavy!