Better abstractions for timed automata

B. Srivathsan

Joint work with F. Herbreteau, D. Kini and I. Walukiewicz

LaBRI, Université Bordeaux 1
IIT Bombay / UIUC

LSV-ENS Cachan

February 2012
Run: finite sequence of transitions,

\[
(s_0, 0, 0) \xrightarrow{0.4,a} (s_1, 0.4, 0) \xrightarrow{0.5,c} (s_3, 0.9, 0.5)
\]

▶ A run is accepting if it ends in a green state.
The problem we are interested in ...

Given a TA, does there exist an accepting run?
The problem we are interested in ... 

Given a TA, does there exist an accepting run?

Theorem [AD94, CY92] 
This problem is PSPACE-complete
First solution to this problem

Key idea: Partition the space of valuations into a finite number of regions

- Region: set of valuations satisfying the same guards w.r.t. time
- Finiteness: Parametrized by maximal constant

Sound and complete [AD94]

Region graph preserves state reachability
First solution to this problem

**Key idea:** Partition the space of valuations into a **finite** number of **regions**

- **Region:** set of valuations satisfying the **same** guards w.r.t. time
- **Finiteness:** Parametrized by **maximal constant**

\[ O(|X|! \cdot M|X|) \] many regions!

---

**Sound and complete [AD94]**

**Region graph** preserves state **reachability**
A more efficient solution...

Key idea: Maintain **all valuations** reachable along a path

\[ x = y \geq 0 \quad x = y \geq 0 \quad y - x \geq 7 \quad y - x \geq 7 \]

\[ x := 0 \]

\[ (x \leq 5) \quad (y \geq 7) \quad x := 0 \]
Zones and zone graph

Zone: set of valuations defined by conjunctions of constraints:
- $x \sim c$
- $x - y \sim c$
- e.g. $(x - y \geq 1) \land y < 2$

Representation: by DBM

Sound and complete [DT98]

Zone graph preserves state reachability
But the zone graph could be infinite ...
Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner

\[
(q_0, Z_0) \quad \rightarrow \quad (q_1, Z_1) \quad \rightarrow \quad (q_2, Z_2)
\]

\[
(q_0, a(Z_0)) \quad \rightarrow \quad (q_1, a(Z')) \quad \rightarrow \quad (q_2, a(Z''))
\]

- Number of **abstracted zones** is **finite**
Use finite abstractions

Key idea: **Abstract** each zone in a **sound** manner

\[
(q_0, Z_0) \quad (q_0, a(Z_0))
\]

\[
(q_1, Z_1) \quad (q_1, a(Z'))
\]

\[
(q_2, Z_2) \quad (q_2, a(Z''))
\]

- **Number of abstracted zones is finite**
- **Coarser** abstraction $\rightarrow$ smaller **abstract zone graph**
Abstractions in literature [Bou04, BBLP06]

Sound and complete

**All the above abstractions preserve state reachability**
Abstractions in literature [Bou04, BBLP06]

Sound and complete

All the above abstractions preserve state reachability

But for implementation abstracted zone should be a zone
Abstractions in literature [Bou04, BBLP06]

Only convex abstractions in implementations!
Efficient use of the non-convex $\alpha_{LU}$ abstraction!
Using $\preceq_{LU}$ for reachability

$q_3 = q_1 \land a(Z_3) \subseteq a(Z_1)$?

Standard algorithm: covering tree
Using $a \preceq_{LU}$ for reachability

$\alpha_{LU}(Z)$ cannot be efficiently stored
Using $\alpha_{LU}$ for reachability

Do not store abstracted zones!

Better abstractions for timed automata - 11/34
Using $\alpha \preceq_{LU}$ for reachability

$q_3 = q_1 \land Z_3 \subseteq \alpha \preceq_{LU}(Z_1)$?

Use $\alpha \preceq_{LU}$ for termination!
Efficient $Z \subseteq a_{LU}(Z')$
What is $\alpha \lessapprox_{LU}$? [BBLP06]

$L_x := \max c \text{ over guards } x > c \text{ and } x \geq c$

$U_x := \max c \text{ over guards } x < c \text{ and } x \leq c$
What is $a \preceq_{LU}$? [BBLP06]

\[ L_x := \max_c \text{ over guards } \quad \begin{cases} \quad x > c \quad \text{and} \quad x \geq c \end{cases} \]

\[ U_x := \max_c \text{ over guards } \quad \begin{cases} \quad x < c \quad \text{and} \quad x \leq c \end{cases} \]

\[ \triangleright v \preceq_{LU} v' : \text{v is simulated by v'} \]

\[ \triangleright a \preceq_{LU}(Z) := \{ v | \exists v' \in Z \text{ s.t. } v \preceq_{LU} v' \} \]
What is $a_{\leq LU}$? [BBLP06]

$L_x := \max c \text{ over guards } x > c \text{ and } x \geq c$

$U_x := \max c \text{ over guards } x < c \text{ and } x \leq c$

$\triangleright v \preceq_{LU} v': v \text{ is simulated by } v' \text{ when for all } x \in X$

$\triangleright a_{\preceq_{LU}}(Z) := \{v \mid \exists v' \in Z \text{ s.t. } v \preceq_{LU} v'\}$
What is $\alpha \preceq_{LU}$? [BBLP06]

$L_x := \max c \text{ over guards } x > c \text{ and } x \geq c$

$U_x := \max c \text{ over guards } x < c \text{ and } x \leq c$

$\triangleright \ v \preceq_{LU} v': v$ is simulated by $v'$ when for all $x \in X$

$\triangleright \ v(x) > v'(x)$

$\triangleright \ \alpha_{LU}(Z) := \{v \mid \exists v' \in Z \text{ s.t. } v \preceq_{LU} v'\}$
What is $a \preceq_{LU}$? [BBLP06]

$L_x := \max c$ over guards $x > c$ and $x \geq c$

$U_x := \max c$ over guards $x < c$ and $x \leq c$

$\triangleright v \preceq_{LU} v': v$ is simulated by $v'$ when for all $x \in X$

$\triangleright v(x) > v'(x)$

$\triangleright a \preceq_{LU}(Z) := \{v \mid \exists v' \in Z \text{ s.t. } v \preceq_{LU} v'\}$
What is $\alpha \lesssim_{LU}$? [BBLP06]

$L_x := \max c \text{ over guards } x > c \text{ and } x \geq c$

$U_x := \max c \text{ over guards } x < c \text{ and } x \leq c$

$\gg \nu \lesssim_{LU} \nu': \nu \text{ is simulated by } \nu' \text{ when for all } x \in X$

$\gg \nu(x) > \nu'(x) \Rightarrow \nu'(x) > L_x, \text{ and}$

$\gg \alpha \lesssim_{LU}(Z) := \{\nu | \exists \nu' \in Z \text{ s.t. } \nu \lesssim_{LU} \nu'\}$
What is $a_{\preceq_{LU}}$? [BBLP06]

$L_x := \max c \text{ over guards } x > c \text{ and } x \geq c$

$U_x := \max c \text{ over guards } x < c \text{ and } x \leq c$

$\triangleright v \preceq_{LU} v'$: $v$ is simulated by $v'$ when for all $x \in X$

$\triangleright v(x) > v'(x) \Rightarrow v'(x) > L_x$, and

$\triangleright v(x) < v'(x)$

$\triangleright a_{\preceq_{LU}}(Z) := \{v \mid \exists v' \in Z \text{ s.t. } v \preceq_{LU} v'\}$
What is $a \preceq_{LU}$? [BBLP06]

$L_x := \max c$ over guards $x > c$ and $x \geq c$

$U_x := \max c$ over guards $x < c$ and $x \leq c$

$\triangleright \; v \preceq_{LU} v'$: $v$ is simulated by $v'$ when for all $x \in X$

$\triangleright \; v(x) > v'(x) \Rightarrow v'(x) > L_x$, and

$\triangleright \; v(x) < v'(x)$

$\triangleright \; a \preceq_{LU} (Z) := \{v \mid \exists v' \in Z \text{ s.t. } v \preceq_{LU} v'\}$
What is $a \preceq_{LU}$? [BBLP06]

$L_x := \max c \text{ over guards } x > c \text{ and } x \geq c$

$U_x := \max c \text{ over guards } x < c \text{ and } x \leq c$

$\uparrow v \preceq_{LU} v': v$ is simulated by $v'$ when for all $x \in X$

$\uparrow v(x) > v'(x) \Rightarrow v'(x) > L_x$, and

$\uparrow v(x) < v'(x) \Rightarrow v(x) > U_x$

$\Rightarrow a \preceq_{LU} (Z) := \{v \mid \exists v' \in Z \text{ s.t. } v \preceq_{LU} v'\}$
Example

\[
\begin{align*}
\text{Better abstractions for timed automata} & \quad 14/34
\end{align*}
\]

\[
\begin{align*}
\triangleright & \quad v \preceq_{LU} v': \text{ when for all } x \in X \\
\triangleright & \quad v(x) > v'(x) \Rightarrow v'(x) > L_x \quad \text{and} \\
\triangleright & \quad v(x) < v'(x) \Rightarrow v(x) > U_x
\end{align*}
\]
Example

\[ v \preceq_{LU} v' : \text{ when for all } x \in X \]

\[ v(x) > v'(x) \Rightarrow v'(x) > L_x \quad \text{and} \]

\[ v(x) < v'(x) \Rightarrow v(x) > U_x \]
Better abstractions for timed automata

Example

- $v \preceq_{LU} v'$: when for all $x \in X$
  - $v(x) > v'(x) \implies v'(x) > L_x$ and
  - $v(x) < v'(x) \implies v(x) > U_x$
Example

- $\mathbf{v} \preceq_{LU} \mathbf{v'}$: when for all $x \in X$
  
  - $\mathbf{v}(x) > \mathbf{v'}(x) \Rightarrow \mathbf{v'}(x) > L_x$ and
  
  - $\mathbf{v}(x) < \mathbf{v'}(x) \Rightarrow \mathbf{v}(x) > U_x$
Example

\[
\begin{align*}
\bullet \quad v \preceq_{LU} v': \text{ when for all } x \in X \\
\quad & v(x) > v'(x) \Rightarrow v'(x) > L_x \checkmark \text{ and } \\
\quad & v(x) < v'(x) \Rightarrow v(x) > U_x
\end{align*}
\]
$v \preceq_{LU} v'$: when for all $x \in X$

- $v(x) > v'(x) \Rightarrow v'(x) > L_x$ and
- $v(x) < v'(x) \Rightarrow v(x) > U_x$
Example

- $v \preceq_{LU} v'$: when for all $x \in X$
  - $v(x) > v'(x) \Rightarrow v'(x) > L_x$ and
  - $v(x) < v'(x) \Rightarrow v(x) > U_x$
Example

\( v \preceq_{LU} v' \): when for all \( x \in X \)

\( v(x) > v'(x) \Rightarrow v'(x) > L_x \) and

\( v(x) < v'(x) \Rightarrow v(x) > U_x \)

\( a \preceq_{LU} (Z) : \{ v \mid \exists v' \in Z \text{ s.t. } v \preceq_{LU} v' \} \)
Coming next…

\[ Z \subseteq \mathcal{a}_{LU}(Z') \]
Step 1: Focus on regions

Union of regions

Every region $R$ that intersects $a_{\leq LU}(Z')$ is included in $a_{\leq LU}(Z')$. 

![Diagram](image-url)
Step 1: Focus on regions

Union of regions

Every region $R$ that intersects $a_{LU}(Z')$ is included in $a_{LU}(Z')$.

$Z \not\subseteq a_{LU}(Z') \iff \exists R. \ R$ intersects $Z$, $R$ not included in $a_{LU}(Z')$
Step 2: When is $R \not\subseteq a_{LU}(Z')$?

$\forall v \leq_{LU} v'$: when for all $x \in X$

$\forall v(x) > v'(x) \Rightarrow v'(x) > L_x$ and

$\forall v(x) < v'(x) \Rightarrow v(x) > U_x$

Collect all $v'$ that simulate $v$
Step 2: When is $R \not\subseteq a_{\preceq LU}(Z')$?

Collect all $v'$ that simulate $v$

$\vDash v \preceq_{LU} v'$: when for all $x \in X$

$\vDash v(x) > v'(x) \Rightarrow v'(x) > L_x$ and

$\vDash v(x) < v'(x) \Rightarrow v(x) > U_x$
Step 2: When is $R \not\subseteq \alpha_{LU}(Z')$?

- $v \preceq_{LU} v'$: when for all $x \in X$
  - $v(x) > v'(x) \Rightarrow v'(x) > L_x$ and
  - $v(x) < v'(x) \Rightarrow v(x) > U_x$

Collect all $v'$ that simulate $v$
Step 2: When is $R \not\subseteq a^{\leq LU}(Z')$?

$\text{Collect all } v' \text{ that simulate } v$

$\blacktriangleright v \preceq_{LU} v'$: when for all $x \in X$

$\blacktriangleright v(x) > v'(x) \Rightarrow v'(x) > L_x$ and

$\blacktriangleright v(x) < v'(x) \Rightarrow v(x) > U_x$
Step 2: When is $R \not\subseteq a_{\preceq LU}(Z')$?

- $v_{\preceq LU} v'$: when for all $x \in X$
  - $v(x) > v'(x) \Rightarrow v'(x) > L_x$ and
  - $v(x) < v'(x) \Rightarrow v(x) > U_x$

- $a_{\preceq LU}^{-1}(R)$: $\{v' \mid \exists v \in R \text{ s.t. } v_{\preceq LU} v'\}$

Inclusion to intersection

$R \not\subseteq a_{\preceq LU}(Z') \iff a_{\preceq LU}^{-1}(R) \cap Z'$ is empty
Step 2: When is $R \not\subseteq a_{\preceq_{LU}}(Z')$?

- $\nabla v_{\preceq_{LU}} v'$: when for all $x \in X$
  - $v(x) > v'(x) \Rightarrow v'(x) > L_x$ and
  - $v(x) < v'(x) \Rightarrow v(x) > U_x$

- $a_{\preceq_{LU}}^{-1}(R)$: \{ $v' : \exists v \in R \text{ s.t. } v_{\preceq_{LU}} v'$ \}

**Inclusion to intersection**

$R \not\subseteq a_{\preceq_{LU}}(Z') \iff a_{\preceq_{LU}}^{-1}(R) \cap Z'$ is empty

For every region $R$, the set $a_{\preceq_{LU}}^{-1}(R)$ is a zone!
Step 2: When is $R \not\subseteq a_{LU}(Z')$?

- $v \lesssim_{LU} v'$: when for all $x \in X$
  - $v(x) > v'(x) \Rightarrow v'(x) > L_x$ and
  - $v(x) < v'(x) \Rightarrow v(x) > U_x$

- $a_{LU}^{-1}(R)$: $\{v' \mid \exists v \in R \text{ s.t. } v \lesssim_{LU} v'\}$

Inclusion to intersection

$$R \not\subseteq a_{LU}(Z') \iff a_{LU}^{-1}(R) \cap Z' \text{ is empty}$$

$$Z \not\subseteq a_{LU}(Z') \iff \exists R. \text{ } R \text{ intersects } Z, \ a_{LU}^{-1}(R) \text{ does not intersect } Z'$$
Step 3: When is $\alpha_{LU}^{-1}(R) \cap Z'$ empty?

**Reduction to two clocks**

$\alpha_{LU}^{-1}(R) \cap Z'$ is empty iff there exist 2 clocks $x, y$ s.t.

$$\alpha_{LU}^{-1}(\text{Proj}_{xy}(R)) \cap \text{Proj}_{xy}(Z')$$

is empty.
Step 3: When is $\alpha_{LU}^{-1}(R) \cap Z'$ empty?

**Reduction to two clocks**

$\alpha_{LU}^{-1}(R) \cap Z'$ is empty iff there exist 2 clocks $x, y$ s.t.

$\alpha_{LU}^{-1}(\text{Proj}_{xy}(R)) \cap \text{Proj}_{xy}(Z')$ is empty

$\text{Proj}_{xy}(R)$ is a region, $\text{Proj}_{xy}(Z')$ is a zone
Step 3: When is $\alpha_{LU}^{-1}(R) \cap Z'$ empty?

<table>
<thead>
<tr>
<th>Reduction to two clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_{LU}^{-1}(R) \cap Z'$ is empty iff there exist 2 clocks $x, y$ s.t. $\alpha_{LU}^{-1}(\text{Proj}<em>{xy}(R)) \cap \text{Proj}</em>{xy}(Z')$ is empty</td>
</tr>
</tbody>
</table>

$\text{Proj}_{xy}(R)$ is a region, $\text{Proj}_{xy}(Z')$ is a zone

$\alpha_{LU}^{-1}(\text{Proj}_{xy}(R)) \cap \text{Proj}_{xy}(Z')$ is empty $\Leftrightarrow$ $\text{Proj}_{xy}(R) \not\subseteq \alpha_{LU}(\text{Proj}_{xy}(Z'))$
Step 3: When is $a_{LU}^{-1}(R) \cap Z'$ empty?

Reduction to two clocks

$a_{LU}^{-1}(R) \cap Z'$ is empty iff there exist 2 clocks $x, y$ s.t.

$$a_{LU}^{-1}(\text{Proj}_{xy}(R)) \cap \text{Proj}_{xy}(Z')$$

is empty

\[\text{Proj}_{xy}(R)\text{ is a region, } \text{Proj}_{xy}(Z')\text{ is a zone}\]

$$a_{LU}^{-1}(\text{Proj}_{xy}(R)) \cap \text{Proj}_{xy}(Z')$$

is empty $\Leftrightarrow \text{Proj}_{xy}(R) \not\subseteq a_{LU}(\text{Proj}_{xy}(Z'))$

$$Z \not\subset a_{LU}(Z') \Leftrightarrow \exists R, x, y. \text{ R intersects } Z, \text{Proj}_{xy}(R) \not\subseteq a_{LU}(\text{Proj}_{xy}(Z'))$$
Efficient inclusion testing

**Theorem**

\[ Z \not\subseteq a_{LU}(Z') \text{ if and only if there exist 2 clocks } x, y \text{ s.t.} \]

\[ \text{Proj}_{xy}(Z) \not\subseteq a_{LU}(\text{Proj}_{xy}(Z')) \]
Efficient inclusion testing

Theorem

\[ Z \not\subseteq a_{LU} (Z') \text{ if and only if there exist 2 clocks } x, y \text{ s.t.} \]

\[ \text{Proj}_{xy}(Z) \not\subseteq a_{LU} (\text{Proj}_{xy}(Z')) \]

Complexity: \( \mathcal{O}(|X|^2) \), where \( X \) is the set of clocks
Efficient inclusion testing

**Theorem**

\[ Z \not\subseteq a_{\leq LU}(Z') \text{ if and only if there exist 2 clocks } x, y \text{ s.t.} \]

\[ \text{Proj}_{xy}(Z) \not\subseteq a_{\leq LU}(\text{Proj}_{xy}(Z')) \]

**Same** complexity as \( Z \subseteq Z' \)!
### Theorem

\[ Z \nsubseteq a_{LU}(Z') \text{ if and only if there exist 2 clocks } x, y \text{ s.t.} \]

\[ \text{Proj}_{xy}(Z) \nsubseteq a_{LU}(\text{Proj}_{xy}(Z')) \]

---

**Slightly** modified comparison works!
So what do we have now...

\[ q_3 = q_1 \land Z_3 \subseteq a_{\leq LU}(Z_1)? \]

Efficient algorithm for \( Z \subseteq a_{\leq LU}(Z') \)
So what do we have now...

\[ \begin{align*}
    q_3 &= q_1 \land \\
    Z_3 &\subseteq a_{\leq LU}(Z_1) \\
\end{align*} \]

Coming next: **prune** the LU bounds!
LU-bounds

\[ x = 1 \]
\[ \{x\} \]

\[ x = 10^6 \]
\[ \{x, y\} \]
\[ y = 10^6 \]

Naive: \( L_x = U_x = 10^6 \), \( L_y = U_y = 10^6 \)

Size of graph \( \sim 10^6 \)
Static analysis: bounds for every $q$ \[BBFL03\]

$$x = 1$$
$$\{x\}$$

Naive: $L_x = U_x = 10^6$, $L_y = U_y = 10^6$

Size of graph $< 10$
Static analysis: bounds for every \( q \) [BBFL03]

\[ x = 1 \]
\[ \{x\} \]

\[ x = 10^6 \]
\[ \{x, y\} \]
\[ y = 10^6 \]

Naive: \( L_x = U_x = 10^6 \), \( L_y = U_y = 10^6 \)

Size of graph \(< 10\)

But this is not enough!
Need to look at semantics...

\[ x = 1 \]
\[ \{x\} \]

\[ x \geq 2 \]
\[ x \leq 1 \]
\[ x = 10^6 \]
\[ y = 10^6 \]

Static analysis: \( 10^6 \)

More than \( 10^6 \) zones at \( q_0 \) **not necessary!**
LU bounds for every \((q, Z)\) in \(ZG(\mathcal{A})\)

constants at

depend on subtree
Constant propagation

\[ L(x) = -\infty \]
\[ U(x) = -\infty \]

\[(q, Z, LU)\]

All tentative nodes consistent → No more exploration → Terminate!

Better abstractions for timed automata
Constant propagation

\[ L(x) = -\infty \]
\[ U(x) = -\infty \]

\((q, Z, LU)\)

\[ x \leq 3 \]
Constant propagation

\[ L(x) = -\infty \]
\[ U(x) = 3 \]

\( (q, Z, LU) \)

\( x \leq 3 \)
Constant propagation

\[ L(x) = -\infty \]
\[ U(x) = 3 \]

\[ x \leq 3 \]
Constant propagation

\[
L(x) = 4 \\
U(x) = 5
\]

\[
(q, Z, LU)
\]

\[
x \leq 3
\]
Constant propagation

\[ L(x) = 4 \]
\[ U(x) = 5 \]

\( x \leq 3 \)

\( Z' \subseteq a_{\leq LU}(Z) \)

(\( q, Z, LU \))

Better abstractions for timed automata - 24/34
Constant propagation

\[ L(x) = 4 \]
\[ U(x) = 5 \]

\[ Z' \subseteq a_{\leq LU}(Z) \]

\[ (q, Z, LU) \]

\[ x \leq 3 \]
\[ x > 6 \]
Constant propagation

\[ L(x) = 6 \]
\[ U(x) = 5 \]

\[ (q, Z, LU) \]

\[ Z' \subseteq a_{LU}(Z) \]

Better abstractions for timed automata - 24/34
Constant propagation

\[ L(x) = 6 \]
\[ U(x) = 5 \]

\[ Z' \subseteq a_{\leq LU}(Z) \]

All tentative nodes consistent → No more exploration → Terminate!

Better abstractions for timed automata - 24/34
Constant propagation

\[ L(x) = 6 \]
\[ U(x) = 5 \]

\[ Z' \subseteq a_{LU}(Z) \]

\[(q, Z, LU)\]

\[ x \leq 3 \]
\[ x > 6 \]

Better abstractions for timed automata - 24/34
**Constant propagation**

\[ L(x) = 6 \]
\[ U(x) = 5 \]

\[ Z' \subseteq a_{\preceq LU}(Z) \]

\[ (q', Z', L'U') \]

\[ x \leq 3 \]
\[ x > 6 \]
\[ x \geq 11 \]
Constant propagation

\[ \begin{align*}
L(x) &= 11 \\
U(x) &= 5
\end{align*} \]

\[ (q, Z, LU) \]

\[ Z' \subseteq a_{LU}(Z) \]

\[ x \leq 3 \quad x > 6 \quad x \geq 11 \]
Constant propagation

\[ L(x) = 11 \]
\[ U(x) = 5 \]

\[ Z' \subseteq a_{\leq LU}(Z) \]

\[ x \leq 3 \]
\[ x > 6 \]
\[ x \geq 11 \]
Constant propagation

\[ L(x) = 11 \]
\[ U(x) = 5 \]

\[ Z' \subseteq a_{\leq LU}(Z) \]

\[ x \leq 3 \]
\[ x > 6 \]
\[ x \geq 11 \]
Constant propagation

\[ L(x) = 11 \]
\[ U(x) = 5 \]

\[ (q, Z, LU) \]

\[ x \leq 3 \]
\[ x > 6 \]
\[ x \geq 11 \]

\[ Z' \subseteq a_{\leq LU}(Z) \]

Better abstractions for timed automata - 24/34
Constant propagation

\[ L(x) = 11 \]
\[ U(x) = 5 \]

\[ (q, Z, LU) \]

\[ x \leq 3 \]
\[ x > 6 \]
\[ x \geq 11 \]

\[ Z' \subseteq a_{\leq LU}(Z) \]

All tentative nodes consistent + No more exploration → Terminate!

Better abstractions for timed automata - 24/34
Invariants on the bounds

- Non tentative nodes: \( LU = \max\{LU_{\text{succ}}\} \) (modulo resets)

- Tentative nodes: \( LU = LU_{\text{covering}} \)
Invariants on the bounds

- Non tentative nodes: \( LU = \max\{LU_{\text{succ}}\} \) (modulo resets)

- Tentative nodes: \( LU = LU_{\text{covering}} \)

**Theorem (Correctness)**

An accepting state is reachable in \( ZG(A) \) iff the algorithm reaches a node with an accepting state and a non-empty zone.
Overall algorithm

- Compute \( ZG(A) : Z \subseteq a_{LU}(Z') \) for termination

- **LU-bounds calculated on-the-fly**
A bonus

- LU-automata: automata with guards determined by $L$ and $U$
- $Z$: an arbitrary reachable zone in some LU-automaton
A bonus

- **LU-automata**: automata with guards determined by $L$ and $U$
- **$Z$**: an arbitrary **reachable zone** in some LU-automaton

Every **sound** and **complete** abstraction $b$ satisfies $b(Z) \subseteq a_{\leq LU}(Z)$

**Theorem**

In the context of **reachability**, the $a_{\leq LU}$ abstraction is the **biggest abstraction** that is **sound** and **complete** for all LU-automata.
<table>
<thead>
<tr>
<th>Model</th>
<th>Our algorithm</th>
<th>UPPAAL’s algorithm</th>
<th>UPPAAL 4.1.3 (-n4 -C -o1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>nodes</td>
<td>s.</td>
<td>nodes</td>
</tr>
<tr>
<td>CSMA/CD7</td>
<td>5046</td>
<td>0.39</td>
<td>5923</td>
</tr>
<tr>
<td>CSMA/CD8</td>
<td>16609</td>
<td>0.75</td>
<td>19017</td>
</tr>
<tr>
<td>CSMA/CD9</td>
<td>54467</td>
<td>9.40</td>
<td>60783</td>
</tr>
<tr>
<td>FDDI10</td>
<td>459</td>
<td>0.04</td>
<td>525</td>
</tr>
<tr>
<td>FDDI20</td>
<td>1719</td>
<td>0.41</td>
<td>2045</td>
</tr>
<tr>
<td>FDDI30</td>
<td>3779</td>
<td>1.70</td>
<td>4565</td>
</tr>
<tr>
<td>Fischer7</td>
<td>7737</td>
<td>0.40</td>
<td>18353</td>
</tr>
<tr>
<td>Fischer8</td>
<td>25080</td>
<td>1.50</td>
<td>85409</td>
</tr>
<tr>
<td>Fischer9</td>
<td>81035</td>
<td>5.70</td>
<td>397989</td>
</tr>
<tr>
<td>Fischer10</td>
<td>—</td>
<td>T.O.</td>
<td>—</td>
</tr>
</tbody>
</table>

- **Extra**\(_{LU}^{+}\) and **static** analysis bounds in UPPAAL
- **α\(_{LU}^{<}\) and **otf** bounds in our algorithm
Experiments I

\[ \mathcal{A}_1 \]

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>y \geq 20 &amp;&amp; x == 2</td>
<td></td>
</tr>
<tr>
<td>q1</td>
<td>x == 1</td>
<td></td>
</tr>
<tr>
<td>q2</td>
<td>y == 10000</td>
<td></td>
</tr>
<tr>
<td>q3</td>
<td>x == 5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x == 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Nodes</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>7</td>
<td>0.0</td>
</tr>
<tr>
<td>UPPAAL's algorithm</td>
<td>2003</td>
<td>0.60</td>
</tr>
<tr>
<td>UPPAAL 4.1.3</td>
<td>2003</td>
<td>0.01</td>
</tr>
</tbody>
</table>
Experiments II

\[ A_2 \]

\[
\begin{align*}
q_0 & : x = 0 \\
q_1 & : y = 10000 \\
q_2 & : y = 10
\end{align*}
\]

<table>
<thead>
<tr>
<th>( A_2 )</th>
<th>nodes</th>
<th>s.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>UPPAAL’s algorithm</td>
<td>10003</td>
<td>0.07</td>
</tr>
<tr>
<td>UPPAAL 4.1.3</td>
<td>10003</td>
<td>0.07</td>
</tr>
</tbody>
</table>
Experiments II

\[ A_2 \]

<table>
<thead>
<tr>
<th></th>
<th>nodes</th>
<th>s.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>UPPAAL’s algorithm</td>
<td>10003</td>
<td>0.07</td>
</tr>
<tr>
<td>UPPAAL 4.1.3</td>
<td>10003</td>
<td>0.07</td>
</tr>
</tbody>
</table>

Occurs in **CSMA/CD**!
Experiments III

\[ A_3 \]

<table>
<thead>
<tr>
<th>( A_3 )</th>
<th>nodes</th>
<th>s.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>UPPAAL’s algorithm</td>
<td>10004</td>
<td>0.37</td>
</tr>
<tr>
<td>UPPAAL 4.1.3</td>
<td>10004</td>
<td>0.32</td>
</tr>
</tbody>
</table>

\[ x == 1 \]

\[ x := 0 \]
Experiments III

\[ A_3 \]

\[
\begin{align*}
x &= 1 \\
x &:= 0 \\
n &= 10 && y \geq 10000 \\
x &\leq 1 \\
x &= 1 \\
y &\leq 10 \\
q_0 &q_1 q_2
\end{align*}
\]

<table>
<thead>
<tr>
<th>( A_3 )</th>
<th>nodes</th>
<th>s.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>UPPAAL’s algorithm</td>
<td>10004</td>
<td>0.37</td>
</tr>
<tr>
<td>UPPAAL 4.1.3</td>
<td>10004</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Occurs in Fischer!
Experiments IV

\[ x - y \geq 1 \]
\[ x > L_x \]

\[ Z' \]
\[ Z \]

\[ L_y = U_y \]
\[ L_x = U_x \]
Experiments IV

\[ L_y = U_y \]

\[ L_x = U_x \]

\[ Z' : x - y \geq 1 \]

\[ Z : x > L_x \]

Occurs in FDDI!
Conclusions & Perspectives

- **Efficient implementation** of a non-convex approximation that is **optimal**
- **On-the-fly learning** of bounds that is **better** than the current static analysis

- Propagating **more** than constants
- Automata with **diagonal** constraints
References

R. Alur and D.L. Dill.
A theory of timed automata.

Static guard analysis in timed automata verification.

Lower and upper bounds in zone-based abstractions of timed automata.

P. Bouyer.
Forward analysis of updatable timed automata.

C. Courcoubetis and M. Yannakakis.
Minimum and maximum delay problems in real-time systems.

C. Daws and S. Tripakis.
Model checking of real-time reachability properties using abstractions.