Timed systems through the lens of logic

S. Akshay\textsuperscript{1}, P. Gastin\textsuperscript{2}, V. Jugé\textsuperscript{3}, S. Krishna\textsuperscript{1}

\textsuperscript{1} Dept of CSE, Indian Institute of Technology Bombay
\textsuperscript{2} LSV, ENS-Paris Saclay & CNRS
\textsuperscript{3} LIGM, Université Paris-Est Marne la Vallée, CNRS

26 June 2019
LICS’2019, Vancouver, Canada
A global view of timed systems

A timed system has several parts:

1. A regular way to generate behaviors: Automata, Expressions
2. Timing features: Clock resets and guards, Event-clocks, Clock updates etc.
A global view of timed systems

A timed system has several parts:

1. A regular way to generate behaviors: Automata, Expressions
2. Timing features: Clock resets and guards, Event-clocks, Clock updates etc.
3. Data structures: stacks, queues, bags, etc.
A global view of timed systems

A timed system has several parts:

1. A regular way to generate behaviors: Automata, Expressions
2. Timing features: Clock resets and guards, Event-clocks, Clock updates etc.
3. Data structures: stacks, queues, bags, etc.

Examples

- Timed automata, event clock automata AD94, AFH99
- Timed pushdown automata BER94, AAS12
- Timed message-passing automata AGKS10, AAK18
A global view of timed systems

A timed system has several parts:

1. A regular way to generate behaviors: Automata, Expressions
2. Timing features: Clock resets and guards, Event-clocks, Clock updates etc.
3. Data structures: stacks, queues, bags, etc.

Examples

- Timed automata, event clock automata \textit{AD94,AFH99}
- Timed pushdown automata \textit{BER94,AAS12}
- Timed message-passing automata \textit{AGKS10,AAK18}

Popular approach: region construction. For each \textit{timing feature} and each \textit{data structure}, redo the proof.

Do we need to do this? Is there something unifying them?
A run of system $S$ is a sequence of instructions

- e.g., with a queue $d_1$ and stack $d_2$ consider

$$\tau = \text{nop } w(d_1) \text{ nop } w(d_1) r(d_1) w(d_2) w(d_2) r(d_1) \text{ nop } r(d_2) r(d_2)$$
A run of system $S$ is a sequence of instructions

- e.g., with a queue $d_1$ and stack $d_2$ consider

$$\tau = \text{nop } w(d_1) \text{ nop } w(d_1) \text{ r}(d_1) \text{ w}(d_2) \text{ w}(d_2) \text{ r}(d_1) \text{ nop } r(d_2) \text{ r}(d_2)$$

- Gives rise to a node and edge-labeled graph $G_\tau$
A run of system $S$ is a sequence of instructions

$G_T$ is valid: push matches pop, FIFO on stack, LIFO on queue
A unifying graphical view

- A run of timed system $S$ is a sequence of timed instructions
- $G_T$ is valid: push matches pop, FIFO on stack, LIFO on queue
A unifying graphical view

- A run of timed system $S$ is a sequence of timed instructions
- $G_\tau$ is valid: push matches pop, FIFO on stack, LIFO on queue
- A run must be realizable, i.e., weighted graph $WG_\tau$ should not have negative cycle!
A unifying graphical view

- A run of timed system \( S \) is a sequence of timed instructions
- \( G_\tau \) is valid: push matches pop, FIFO on stack, LIFO on queue
- A run must be realizable, i.e., weighted graph \( WG_\tau \) should not have negative cycle!

\[
\begin{align*}
G_\tau &= 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \\
\tau &= \text{nop} \\
s_1 &= x := 0 \\
s_2 &= w(d_1) \\
s_3 &= y := 0 \\
s_4 &= w(d_1) \\
s_5 &= y \leq 1 \\
s_6 &= r(d_1) \\
s_7 &= w(d_2) \\
s_8 &= w(d_2) \\
s_9 &= r(d_2) \\
s_{10} &= \text{nop} \\
s_{11} &= r(d_2)
\end{align*}
\]

\[
\begin{align*}
WG_\tau &= 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 11 \\
\text{Conditions:} \\
&\leq 0 \\
&\leq 0 < -2 \\
&\leq 1 < 6 \\
&\leq 1 < 3 \\
&\leq 3 < 5 \\
&\leq -2 < -4
\end{align*}
\]

So emptiness asks if there exists \( \tau \) generated by \( S \) such that
(i) \( G_\tau \) is valid and (ii) \( WG_\tau \) is realizable?
Related work

Untimed setting - Use Courcelle’s theorem

- Show that graphs $G_\tau$ obtained have bounded tree-width
- Write validity in MSO over these graphs $G_\tau$
- Interpret these graphs over trees, and reduce to emptiness of tree automata

Main questions:

- Is realizability MSO definable?
- Do timed graphs $WG_\tau$ have bounded tree-width?
- Can you avoid the complexity blowup?
Related work (contd.)

Timed setting - for TPDA and restricted TMPDA

AGK16, AGKS17, AGK18

- Show that timed graphs from TPDA have bounded tree-width
- Directly and carefully build tree automata to check emptiness

Main question:

- how to handle generic data structures, timing features?
- Is there a higher level treatment, whereby we can avoid showing bound on tree-width for each timed system?
Related work (contd.)

Timed setting - for TPDA and restricted TMPDA

- AGK16, AGKS17, AGK18
- Show that timed graphs from TPDA have bounded tree-width
- Directly and carefully build tree automata to check emptiness

Main question:
- how to handle generic data structures, timing features?
- Is there a higher level treatment, whereby we can avoid showing bound on tree-width for each timed system?

Orthogonal technique for TA, TPDA

- Encoding as registers and going via atoms CL15, CLLM17, CL18
Our results

Theorem
Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

- width is the size of the largest antichain
- width=1 implies linear order, i.e., graphs coming from sequential systems.
Our results

Theorem
Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

A template to analyze timed systems via graphs and logic
1. Timed systems to Graphs
   - allows us to decouple data structure $G_\tau$ and timing $WG_\tau$ issues
2. Graphs to Logic
Our results

**Theorem**
Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

**A template to analyze timed systems via graphs and logic**

1. Timed systems to Graphs
   - allows us to decouple data structure $G_\tau$ and timing $WG_\tau$ issues
2. Graphs to Logic
   - Using above theorem above get $\Psi$ for realizability over $WG$
Our results

Theorem
Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

A template to analyze timed systems via graphs and logic

1. Timed systems to Graphs
   - allows us to decouple data structure $G_{\tau}$ and timing $WG_{\tau}$ issues

2. Graphs to Logic
   - Using above theorem above get $\Psi$ for realizability over $WG$
   - A challenge: how do we relate the decoupled graphs?
Our results

**Theorem**

Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

**A template to analyze timed systems via graphs and logic**

1. Timed systems to Graphs
   - allows us to decouple data structure $G_\tau$ and timing $WG_\tau$ issues

2. Graphs to Logic
   - Using above theorem above get $\Psi$ for realizability over $WG$
   - Lemma: Given valid $\tau$, $WG_\tau$ can be logically interpreted into $G_\tau$. Thus convert $\Psi$ over $WG_\tau$ into $\Psi'$ over $G_\tau$
Our results

Theorem

Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

A template to analyze timed systems via graphs and logic

1. Timed systems to Graphs
   - allows us to decouple data structure $G_\tau$ and timing $WG_\tau$ issues

2. Graphs to Logic
   - Using above theorem above get $\Psi$ for realizability over $WG$
   - Lemma: Given valid $\tau$, $WG_\tau$ can be logically interpreted into $G_\tau$. Thus convert $\Psi$ over $WG_\tau$ into $\Psi'$ over $G_\tau$

Theorem

Emptiness of timed system can be reduced to satisfiability of formula in MSO/EQ-ICPDL
Our results

Theorem

Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

A template to analyze timed systems via graphs and logic

1. Timed systems to Graphs
   - allows us to decouple data structure $G_\tau$ and timing $W G_\tau$ issues

2. Graphs to Logic

3. (if you really want emptiness,) Logic back to Automata
   - under-approximate approach to decidability of emptiness.
   - e.g., if tree-width is bounded, then interpret these graphs in trees and obtain tree automata.
Our results

Theorem
Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

A template to analyze timed systems via graphs and logic

1. Timed systems to Graphs
   - allows us to decouple data structure $G_\tau$ and timing $WG_\tau$ issues

2. Graphs to Logic

3. (if you really want emptiness,) Logic back to Automata
   - under-approximate approach to decidability of emptiness.
   - e.g., if tree-width is bounded, then interpret these graphs in trees and obtain tree automata.
   - Note tree-width is now in untimed graphs $G_\tau$!
Our results

Theorem

Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

A template to analyze timed systems via graphs and logic

1 Timed systems to Graphs
   - allows us to decouple data structure $G_{\tau}$ and timing $WG_{\tau}$ issues

2 Graphs to Logic

3 (if you really want emptiness,) Logic back to Automata
   - under-approximate approach to decidability of emptiness.
   - e.g., if tree-width is bounded, then interpret these graphs in trees and obtain tree automata.
   - Note tree-width is now in untimed graphs $G_{\tau}$!
   - Using EQ-ICPDL instead of MSO gives good complexity
Our results

**Theorem**
Realizability of weighted graphs is MSO (and EQ-ICPDL) definable iff the set of graphs has width 1.

**A template to analyze timed systems via graphs and logic**

1. Timed systems to Graphs
   - allows us to decouple data structure $G_\tau$ and timing $WG_\tau$ issues

2. Graphs to Logic

3. (if you really want emptiness,) Logic back to Automata
   - under-approximate approach to decidability of emptiness.
   - e.g., if tree-width is bounded, then interpret these graphs in trees and obtain tree automata.
   - Note tree-width is now in untimed graphs $G_\tau$!
   - Using EQ-ICPDL instead of MSO gives good complexity

**Extensions**: Capture rich timing interplay & model checking
What logic shall we use?

Propositional dynamic logic with Intersection & Converse (ICPDL)

We have the following, with \( p \in \Sigma \) and \( \gamma \in \Gamma \):

\[
\begin{align*}
\Phi & ::= E \sigma : \neg \Phi : \Phi \lor \Phi \\
\sigma & ::= \top : p : \sigma \lor \sigma : \neg \sigma : \langle \pi \rangle \sigma : \text{loop}(\pi) \\
\pi & ::= \gamma : \text{test}\{\sigma\} : \pi + \pi : \pi \cdot \pi : \pi^* : \pi^{-1} : \pi \cap \pi
\end{align*}
\]
What logic shall we use?

Propositional dynamic logic with Intersection & Converse (ICPDL)

We have the following, with $p \in \Sigma$ and $\gamma \in \Gamma$:

$$\Phi ::= E\sigma : \neg\Phi : \Phi \lor \Phi$$

$$\sigma ::= \top : p : \sigma \lor \sigma : \neg\sigma : \langle \pi \rangle\sigma : \text{loop}(\pi)$$

$$\pi ::= \gamma \rightarrow : \text{test}\{\sigma\} : \pi + \pi : \pi \cdot \pi : \pi^* : \pi^{-1} : \pi \cap \pi$$

- $\Phi$ are sentences, $E$ existential node quantifier.
- $\sigma$ node or state formulae with one (implicit) free FOvar.
- $\pi$ path or program formulae with two (implicit) free FOvar.
What logic shall we use?

Propositional dynamic logic with Intersection & Converse (ICPDL)

We have the following, with $p \in \Sigma$ and $\gamma \in \Gamma$:

$$
\Phi ::= E\sigma : \neg\Phi : \Phi \lor \Phi \\
\sigma ::= \top : p : \sigma \lor \sigma : \neg\sigma : \langle \pi \rangle\sigma : \text{loop(}\pi\text{)} \\
\pi ::= \gamma \rightarrow : \text{test}\{\sigma\} : \pi + \pi : \pi \cdot \pi : \pi^* : \pi^{-1} : \pi \cap \pi
$$

(\Sigma, \Gamma)-labeled graph, $\Sigma = \{p, q, r, s\}$, $\Gamma = \{c, e, d, f, \prec\}$
What logic shall we use?

Propositional dynamic logic with Intersection & Converse (ICPDL)

We have the following, with $p \in \Sigma$ and $\gamma \in \Gamma$:

$$\Phi ::= E \sigma : \neg \Phi : \Phi \lor \Phi$$

$$\sigma ::= \top : p : \sigma \lor \sigma : \neg \sigma : \langle \pi \rangle \sigma : \text{loop}(\pi)$$

$$\pi ::= \gamma : \text{test}\{\sigma\} : \pi + \pi : \pi \cdot \pi : \pi^* : \pi^{-1} : \pi \cap \pi$$

(\Sigma, \Gamma)-labeled graph, $\Sigma = \{p, q, r, s\}$, $\Gamma = \{c, e, d, f, \prec\}$
What logic shall we use?

Propositional dynamic logic with Intersection & Converse (ICPDL)

We have the following, with $p \in \Sigma$ and $\gamma \in \Gamma$:

$$\Phi ::= E\sigma : \neg \Phi : \Phi \lor \Phi$$

$$\sigma ::= \top : p : \sigma \lor \sigma : \neg \sigma : \langle \pi \rangle \sigma : \text{loop}(\pi)$$

$$\pi ::= \gamma : \text{test}\{\sigma\} : \pi + \pi : \pi \cdot \pi : \pi^* : \pi^{-1} : \pi \cap \pi$$

$(\Sigma, \Gamma)$-labeled graph, $\Sigma = \{p, q, r, s\}$, $\Gamma = \{c, e, d, f, \prec\}$

$$E \langle (\text{test}\{p \lor q\} \cdot \prec)^* \rangle r$$

$$\neg E \langle \prec \rangle (p \land s)$$

$$E \lor_{(d, d') \in (\Gamma \backslash \{\prec\})^2, d \neq d'} \text{loop}(d \cdot d'^{-1})$$
What logic shall we use?

Propositional dynamic logic with Intersection & Converse (ICPDL)

We have the following, with $p \in \Sigma$ and $\gamma \in \Gamma$:

$$
\Phi ::= E \sigma : \neg \Phi : \Phi \lor \Phi \\
\sigma ::= \top : p : \sigma \lor \sigma : \neg \sigma : \langle \pi \rangle \sigma : \text{loop}(\pi) \\
\pi ::= \gamma : \text{test}\{\sigma\} : \pi + \pi : \pi \cdot \pi : \pi^* : \pi^{-1} : \pi \cap \pi
$$

(\Sigma, \Gamma)-labeled graph, $\Sigma = \{p, q, r, s\}$, $\Gamma = \{c, e, d, f, \prec\}$
What logic shall we use?

Propositional dynamic logic with Intersection & Converse (ICPDL)

We have the following, with $p \in \Sigma$ and $\gamma \in \Gamma$:

$$
\begin{align*}
\Phi &::= E\sigma : \neg\Phi : \Phi \lor \Phi \\
\sigma &::= \top : p : \sigma \lor \sigma : \neg\sigma : \langle\pi\rangle\sigma : \text{loop}(\pi) \\
\pi &::= \gamma : \text{test}\{\sigma\} : \pi + \pi : \pi \cdot \pi : \pi^* : \pi^{-1} : \pi \cap \pi
\end{align*}
$$

EQ-ICPDL$(\Sigma, \Gamma)$ allows $\exists$-quant over new propositional variables

$$
\Psi = \exists p_1, \ldots, p_n \Phi \text{ where } AP = \{p_1, \ldots, p_n\} \text{ is disjoint from } \Sigma \text{ and } \Phi \in \text{ICPDL}(\Sigma \uplus AP, \Gamma).
$$
Why this dynamic logic

Reasons for using EQ-ICPDL

- The talk got scheduled in session titled “dynamic logics”!
Why this dynamic logic

Reasons for using EQ-ICPDL

- The talk got scheduled in session titled “dynamic logics”!
- Many properties are easier to write!
Why this dynamic logic

Reasons for using EQ-ICPDL

- The talk got scheduled in session titled “dynamic logics”!
- Many properties are easier to write!
- EQ-ICPDL is strictly contained in MSO
Why this dynamic logic

Reasons for using EQ-ICPDL

- The talk got scheduled in session titled “dynamic logics”!
- Many properties are easier to write!
- EQ-ICPDL is strictly contained in MSO
- The following theorem by Göller, Lohrey, Lutz

Theorem [GLL09]

Given $k \geq 1$ in unary and a formula $\Psi$ in $\text{EQ-ICPDL}(\Sigma, \Gamma)$ of intersection width bounded by a constant, checking whether $G \models \Psi$ for some $(\Sigma, \Gamma)$-labeled graph $G$ whose tree-width is at most $k$ can be solved in EXPTIME.
Solving realizability by modulo counting

\( G \) is realizable

iff \( \exists ts : V \to \mathbb{R} \) s.t

- \( \forall u \rightsquigarrow^a v, ts(v) - ts(u) \leq a \)
- \( \forall u \rightarrow v, 0 \leq ts(v) - ts(u) \)
Assume only closed guards: $G$ is realizable

iff $\exists ts : V \rightarrow \mathbb{N}$ s.t

- $\forall u \xrightarrow{a} v, ts(v) - ts(u) \leq a$
- $\forall u \rightarrow v, 0 \leq ts(v) - ts(u)$
Assume only closed guards: \( G \) is realizable

iff \( \exists ts : V \rightarrow \mathbb{N} \) s.t

- \( \forall u \xrightarrow{a} v, ts(v) - ts(u) \leq a \)
- \( \forall u \rightarrow v, 0 \leq ts(v) - ts(u) \leq M - 1 \), where \( M \) is max const

\[ M = 5 \]
Assume only closed guards: $G$ is realizable

iff $\exists ts : V \rightarrow \mathbb{N}$ s.t $\forall u \sim^a v$, $ts(v) - ts(u) \leq a$. 

$M = 5$

$0 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 9 \rightarrow 0$

$0 \rightarrow 0 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 9 \rightarrow 0$

$0 \rightarrow 0 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 9 \rightarrow 0$

$0 \rightarrow 0 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 9 \rightarrow 0$

$0 \rightarrow 0 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 9 \rightarrow 0$
Solving realizability by modulo counting

Assume only closed guards: $G$ is realizable

 iff $\exists ts : V \rightarrow \mathbb{N}$ s.t $\forall u \sim^a v, ts(v) - ts(u) \leq a.$

$M = 5$

$\exists tsm : V \rightarrow \{0, \ldots, M - 1\}$ s.t $\forall u \sim^a v,$

- if $u \preceq v,$ then $(tsm(v) - tsm(u))[M] \leq a$
- if $v \prec u,$ then $(tsm(v) - tsm(u))[M] \geq -a$
Assume only closed guards: $G$ is realizable

iff $\exists ts : V \rightarrow \mathbb{N}$ s.t $\forall u \leftrightarrow^a v, ts(v) - ts(u) \leq a$.

$\exists tsm : V \rightarrow \{0, \ldots, M - 1\}$ s.t $\forall u \leftrightarrow^a v$,

- if $u \preceq v$, then $(tsm(v) - tsm(u))[M] \leq a$
- if $v \prec u$, then $(tsm(v) - tsm(u))[M] \geq -a$ or modulo counting grew big in between
Solving realizability by modulo counting

Assume only closed guards: \( G \) is realizable iff

\[
\exists ts : V \rightarrow \mathbb{N} \text{ s.t. } \forall u \sim^a v, \; ts(v) - ts(u) \leq a.
\]

\( M = 5 \)

\[
\exists tsm : V \rightarrow \{0, \ldots M - 1\} \text{ s.t. } \forall u \sim^a v,
\]

- if \( u \preceq v \), then \( (tsm(v) - tsm(u))[M] \leq a \)
- if \( v \prec u \), then \( (tsm(v) - tsm(u))[M] \geq -a \) or modulo counting grew big in between
Assume only closed guards: \( G \) is realizable

iff \( \exists ts : V \to \mathbb{N} \text{ s.t } \forall u \overset{a}{\leadsto} v, ts(v) - ts(u) \leq a. \)

\[
M = 5
0 \rightarrow 0 \rightarrow 3 \rightarrow 3 \rightarrow 4 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 4
\]

\( \exists tsm : V \to \{0, \ldots, M - 1\} \text{ s.t } \forall u \overset{a}{\leadsto} v, \)

- if \( u \preceq v \), then \((tsm(v) - tsm(u))[M] \leq a\) and modulo counting didn’t grow big in between
- if \( v \preceq u \), then \((tsm(v) - tsm(u))[M] \geq -a\) or modulo counting grew big in between
Assume only closed guards: \( G \) is realizable
\[ \exists ts : V \rightarrow \mathbb{N} \text{ s.t } \forall u \sim^a v, ts(v) - ts(u) \leq a. \]

\[ M = 5 \]

\[ \iff \exists tsm : V \rightarrow \{0, \ldots, M - 1\} \text{ s.t } \forall u \sim^a v, \]

- if \( u \preceq v \), then \( (tsm(v) - tsm(u))[M] \leq a \) and modulo counting didn’t grow big in between
- if \( v \prec u \), then \( (tsm(v) - tsm(u))[M] \geq -a \) or modulo counting grew big in between
Solving realizability by modulo counting

Assume only closed guards: \( G \) is realizable

\[
\text{iff } \exists ts : V \to \mathbb{N} \text{ s.t } \forall u \overset{a}{\rightsquigarrow} v, ts(v) - ts(u) \leq a.
\]

\[
M = 5
\]

\[
\begin{array}{c}
0 \\
3 \\
4 \\
1 \\
3 \\
4 \\
0
\end{array}
\]

\[
\begin{array}{ccc}
0 & 3 & 2 \\
3 & 4 & 1 \\
1 & 3 & 4 \\
4 & 0 & 0
\end{array}
\]

\[
\text{iff } \exists tsm : V \to \{0, \ldots, M - 1\} \text{ s.t } \forall u \overset{a}{\rightsquigarrow} v,
\]

- if \( u \preceq v \), then \((tsm(v) - tsm(u))[M] \leq a\) and modulo counting didn’t grow big in between
- if \( v \prec u \), then \((tsm(v) - tsm(u))[M] \geq -a\) or modulo counting grew big in between

\( (u, v) \) is big if \( \exists u \prec w \prec x \preceq v \) s.t

\[
(tsm(w) - tsm(u))[M] + (tsm(x) - tsm(w))[M] \geq M
\]
Realizable iff \( \exists tsm : V \to \{0, \ldots, M - 1\} \) s.t \( \forall u \preceq^a v, \)

- if \( u \preceq v \), then \( (tsm(v) - tsm(u))[M] \leq a \) and \( (u, v) \) is not big
- if \( v \prec u \), then \( (tsm(v) - tsm(u))[M] \geq -a \) or \( (u, v) \) is big

where \( (u, v) \) is big if \( \exists u \prec w \prec x \preceq v \) s.t

\[
(tsm(w) - tsm(u))[M] + (tsm(x) - tsm(w))[M] \geq M
\]

\[
\text{BigPath} = \sum_{0 \leq i, j, k < M} \text{test}\{p_i\} \cdot \to^+ \text{test}\{p_j\} \cdot \to^+ \text{test}\{p_k\} \cdot \to^*
\]

\[
(j - i)[M] + (k - j)[M] \geq M
\]
Realizable iff $\exists tsm : V \rightarrow \{0, \ldots, M - 1\}$ s.t $\forall u \rightsquigarrow^a v$, 

- if $u \preceq v$, then $(tsm(v) - tsm(u))[M] \leq a$ and $(u, v)$ is not big
- if $v < u$, then $(tsm(v) - tsm(u))[M] \geq -a$ or $(u, v)$ is big

where $(u, v)$ is big if $\exists u < w < x \preceq v$ s.t

$$(tsm(w) - tsm(u))[M] + (tsm(x) - tsm(w))[M] \geq M$$

$$\text{BigPath} = \sum_{0 \leq i, j, k < M} \text{test}\{p_i\} \cdot \rightarrow^+ \cdot \text{test}\{p_j\} \cdot \rightarrow^+ \cdot \text{test}\{p_k\} \cdot \rightarrow^*$$

$$(j - i)[M] + (k - j)[M] \geq M$$

$(u, v)$ is not big $= \neg E \bigvee_{-M < \alpha < M} \text{loop}(\text{BigPath} \cdot \xrightarrow{\leq \alpha}^{-1})$
Writing it in EQ-ICPDL

Realizable iff $\exists tsm : V \rightarrow \{0, \ldots, M - 1\}$ s.t $\forall u \ni^a v$,

- if $u \preceq v$, then $(tsm(v) - tsm(u))[M] \leq a$ and $(u, v)$ is not big
- if $v \prec u$, then $(tsm(v) - tsm(u))[M] \geq -a$ or $(u, v)$ is big

where $(u, v)$ is big if $\exists u \prec w \prec x \preceq v$ s.t

$$(tsm(w) - tsm(u))[M] + (tsm(x) - tsm(w))[M] \geq M$$

Realizable $= \exists p_1, \ldots, p_{M-1}$ Partition $\land$ Forward $\land$ Backward

Partition $= A \bigvee_{0 \leq i < M} [p_i \land \bigwedge_{j \neq i} \neg p_j]$

Forward $= \neg E \bigvee_{-M < \alpha < M} \text{loop}(\text{BigPath} \cdot \frac{\leq \alpha}{\rightarrow^{-1}}) \land \neg E \bigvee_{0 \leq i, j < M} \text{loop}(\text{test}\{p_i\} \cdot \frac{\leq \alpha}{\rightarrow} \cdot \text{test}\{p_j\} \cdot (\rightarrow^{-1})^+)$

Similarly for Backward, but need to define $\neg \text{BigPath}$ (see paper)
What about strict/open guards

What happens when there are both closed and open guards?

Realizable $\iff \exists tsm : V \rightarrow \{0, \ldots M - 1\} \text{ s.t } \forall u \leftarrow^a v$,

- if $u \leq v$, then $(tsm(v) - tsm(u))[M] \leq a$ and $(u, v)$ is not big
- if $v \prec u$, then $(tsm(v) - tsm(u))[M] \geq -a$ or $(u, v)$ is big

But the reverse direction is not true, since strictness could invalidate assignments.

Capturing strict guards

Consider the orderings of fractional parts. These should not form a cycle which a strict constraint within!

Uses intersection (but with intersection-width 2).
What about strict/open guards

What happens when there are both closed and open guards?

Realizable $\iff \exists tsm : V \to \{0, \ldots, M - 1\}$ s.t $\forall u \sim^a v$,
- if $u \preceq v$, then $(tsm(v) - tsm(u))[M] \leq a$ and $(u, v)$ is not big
- if $v \prec u$, then $(tsm(v) - tsm(u))[M] \geq -a$ or $(u, v)$ is big

But the reverse direction is not true, since strictness could invalidate assignments.
What about strict/open guards

What happens when there are both closed and open guards?

Realizable \( \iff \exists tsm : V \rightarrow \{0, \ldots, M - 1\} \text{ s.t } \forall u \preccurlyeq^a v,
\begin{align*}
&\text{if } u \leq v, \text{ then } (tsm(v) - tsm(u))[M] \leq a \text{ and } (u, v) \text{ is not big} \\
&\text{if } v \prec u, \text{ then } (tsm(v) - tsm(u))[M] \geq -a \text{ or } (u, v) \text{ is big}
\end{align*}

But the reverse direction is not true, since strictness could invalidate assignments.

Capturing strict guards

- Consider the orderings of fractional parts.
- These should not form a cycle which a strict constraint within!
- Uses intersection (but with intersection-width 2).

Realizable = \( \exists p_1, \ldots, p_{M-1} \text{ Partition } \wedge \text{Forward } \wedge \text{Backward } \wedge \text{noFracCycle} \)
Realizability is not MSO definable without the linear order

- Width of a partial order $=$ maximal size of anti-chain.
- Linear order has width 1.
Realizability is not MSO definable without the linear order

- Width of a partial order $=$ maximal size of anti-chain.
- Linear order has width 1.
- Consider the following example with width 2.
Realizability is not MSO definable without the linear order

- Width of a partial order $=$ maximal size of anti-chain.
- Linear order has width 1.
- Consider the following example with width 2.
- The graph is realizable iff $\#$ blue edges is $\geq$ $\#$ red edges.
Realizability is not MSO definable without the linear order

- Width of a partial order $=$ maximal size of anti-chain.
- Linear order has width 1.
- Consider the following example with width 2.
- The graph is realizable iff $\#$blue edges is $\geq \#$red edges.
- Cannot be expressed in MSO (formal proof by backward translation).
A two step template to capture rich timing features

- Capture timing as edges on graph
- Relate the events in logic
Application and extensions

A two step template to capture rich timing features

- Capture timing as edges on graph
- Relate the events in logic

Example: event-clock $next_a$
A two step template to capture rich timing features

- Capture timing as edges on graph
- Relate the events in logic

Example: event-clock $next_a$
- edge between current event and next occurrence of $a$

$$\sum_{a \in AP} \text{test}\{ (next_a \triangleleft \alpha) \} \cdot \rightarrow \cdot (\text{test}\{ \neg a \} \cdot \rightarrow)^* \cdot \text{test}\{ a \}$$
Application and extensions

A two step template to capture rich timing features

- Capture timing as edges on graph
- Relate the events in logic

Clock tracking/renaming

\[
\tau = x_1 := 0 \\
x_2 := 0 \\
x_3 := 0 \\
\]
\[
d_1 := x_1 \\
x_2 := 0 \\
x_1 := 0 \\
x_4 := x_2 \\
\]
\[
d_2 := x_2 \\
x_2 := 0 \\
x_3 := x_4 \\
x_3 < 3 \\
\]
\[
x_4 := d_2 \\
x_4 < 4 \\
\]

Figure: Intricate flow of information in complex updates.
A two step template to capture rich timing features

- Capture timing as edges on graph
- Relate the events in logic

Clock tracking/renaming

\[
\begin{align*}
\tau &= x_1 := 0 \\
x_2 &= 0 \\
x_3 &= 0 \\
d_1 &= x_1 \\
x_2 &= d_1 \\
x_1 &= 0 \\
x_4 &= x_2 \\
x_3 &= x_4 \\
d_2 &= x_2 \\
x_2 &= 0 \\
x_4 &= d_2 \\
x_3 &= 3 \\
x_4 &= 4
\end{align*}
\]

Figure: Intricate flow of information in complex updates.
Application and extensions

A two step template to capture rich timing features

- Capture timing as edges on graph
- Relate the events in logic

Clock tracking/renaming

Figure: Intricate flow of information in complex updates.

More in the paper: Model checking
Untimed and some limited timed specifications.
Conclusion

Highlights

- Realizability is MSO definable over sequential systems
- Template for analyzing rich time features in systems with data structures using graphs and logic
- Use EQ-ICPDL instead of MSO to obtain good complexity

Future work

- More consequences and applications.
- Distributed systems.
- A converse characterization?!
Conclusion

Highlights

- Realizability is MSO definable over sequential systems
- Template for analyzing rich time features in systems with data structures using graphs and logic
- Use EQ-ICPDL instead of MSO to obtain good complexity

Future work

- More consequences and applications.
- Distributed systems.
- A converse characterization?!