# Timed systems through the lens of logic

**S. Akshay**[1], P. Gastin[2], V. Jugé[3], S. Krishna[1]

[1] Dept of CSE, Indian Institute of Technology Bombay
[2] LSV, ENS-Paris Saclay & CNRS
[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 AD94,AFH99
- Timed pushdown automata BER94,AAS12
- Timed message-passing automata AGKS10,AAK18

- Popular approach: region construction. For each timing feature and each data structure, redo the proof.
- Do we need to do this? Is there something unifying them?

# A unifying graphical view

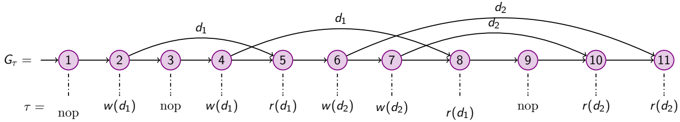
- A run of system $S$ is a sequence of instructions
  - e.g., with a queue $d_1$ and stack $d_2$ consider

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

# A unifying graphical view

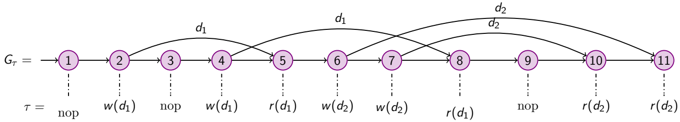
- A run of system $S$ is a sequence of instructions
  - e.g., with a queue $d_1$ and stack $d_2$ consider

    $\tau = \mathrm{nop}\ w(d_1)\ \mathrm{nop}\ w(d_1)\ r(d_1)\ w(d_2)\ w(d_2)\ r(d_1)\ \mathrm{nop}\ r(d_2)\ r(d_2)$
- Gives rise to a node and edge-labeled graph $G_\tau$



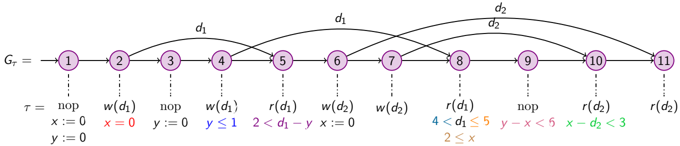
# A unifying graphical view

- A run of system $S$ is a sequence of instructions
- $G_\tau$ is valid: push matches pop, FIFO on stack, LIFO on queue



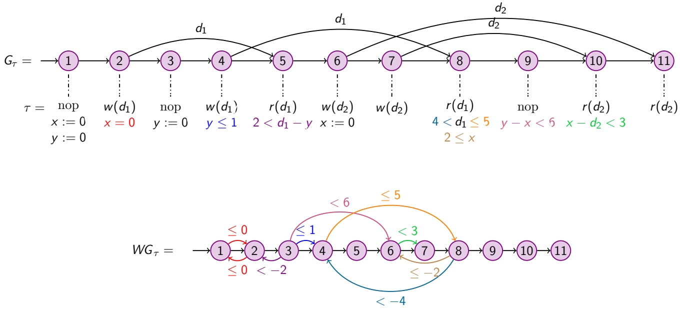
# A unifying graphical view

- 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



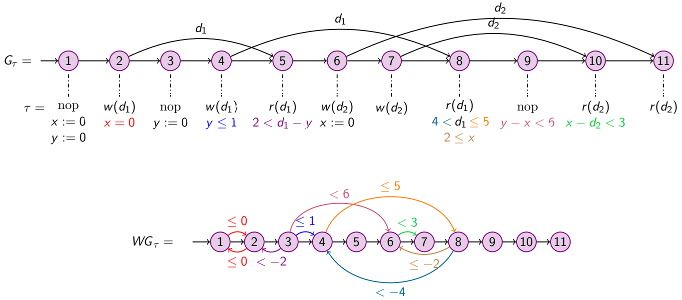
# A unifying graphical view

- 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!




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

MP11,AKG12,AG14

- 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 $WG_\tau$ 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.

# 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 $\gamma \in \Gamma$:

$$\Phi ::= \mathsf{E}\,\sigma \;:\; \neg\Phi \;:\; \Phi \vee \Phi$$

$$\sigma ::= \top \;:\; p \;:\; \sigma \vee \sigma \;:\; \neg\sigma \;:\; \langle\pi\rangle\sigma \;:\; \mathsf{loop}(\pi)$$

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

# What logic shall we use?

## Propositional dynamic logic with Intersection & Converse(ICPDL)

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

$$\Phi ::= \mathsf{E}\,\sigma \;:\; \neg\Phi \;:\; \Phi \vee \Phi$$

$$\sigma ::= \top \;:\; p \;:\; \sigma \vee \sigma \;:\; \neg\sigma \;:\; \langle\pi\rangle\sigma \;:\; \mathsf{loop}(\pi)$$

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

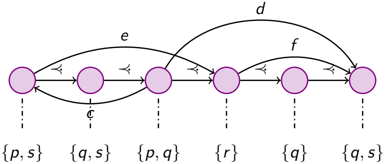
- $\Phi$ are sentences, $E$ existential node quantifier.
- $\sigma$ node or state formulae with one (implicit) free FOvar
- $\pi$ path or program formulae with two (implicit) free FOvar

# What logic shall we use?

## Propositional dynamic logic with Intersection & Converse(ICPDL)

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

$$\Phi ::= \mathsf{E}\,\sigma \ : \ \neg\Phi \ : \ \Phi \vee \Phi$$
$$\sigma ::= \top \ : \ p \ : \ \sigma \vee \sigma \ : \ \neg\sigma \ : \ \langle\pi\rangle\sigma \ : \ \mathrm{loop}(\pi)$$
$$\pi ::= \xrightarrow{\gamma} \ : \ \mathrm{test}\{\sigma\} \ : \ \pi + \pi \ : \ \pi \cdot \pi \ : \ \pi^* \ : \ \pi^{-1} \ : \ \pi \cap \pi$$




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

# What logic shall we use?

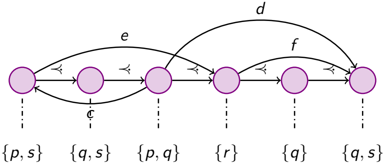
## Propositional dynamic logic with Intersection & Converse(ICPDL)

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

$$\Phi ::= \mathsf{E}\,\sigma \;:\; \neg\Phi \;:\; \Phi \vee \Phi$$

$$\sigma ::= \top \;:\; p \;:\; \sigma \vee \sigma \;:\; \neg\sigma \;:\; \langle\pi\rangle\sigma \;:\; \mathrm{loop}(\pi)$$

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




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

$\mathsf{E}\,\langle(\mathrm{test}\{p \vee q\} \cdot \xrightarrow{\prec})^{*}\rangle r$

$\neg\mathsf{E}\,\langle\xrightarrow{\prec}\rangle(p \wedge s)$

# What logic shall we use?

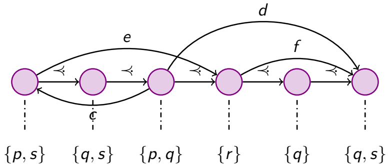
## Propositional dynamic logic with Intersection & Converse(ICPDL)

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

$$\Phi ::= \mathsf{E}\,\sigma \;:\; \neg\Phi \;:\; \Phi \vee \Phi$$

$$\sigma ::= \top \;:\; p \;:\; \sigma \vee \sigma \;:\; \neg\sigma \;:\; \langle\pi\rangle\sigma \;:\; \mathrm{loop}(\pi)$$

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




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

$$\mathsf{E}\,\langle(\mathrm{test}\{p \vee q\} \cdot \xrightarrow{\prec})^*\rangle r$$

$$\neg\mathsf{E}\,\langle\xrightarrow{\prec}\rangle(p \wedge s)$$

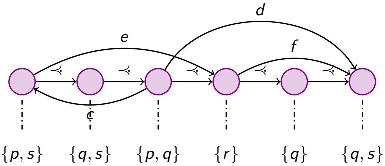
$$\mathsf{E}\,\bigvee_{(d,d') \in (\Gamma \setminus \{\prec\})^2, d \neq d'} \mathrm{loop}(\xrightarrow{d} \cdot \xrightarrow{d'}{}^{-1})$$

# What logic shall we use?

## Propositional dynamic logic with Intersection & Converse(ICPDL)

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

$$\Phi ::= \mathsf{E}\,\sigma \ : \ \neg\Phi \ : \ \Phi \vee \Phi$$
$$\sigma ::= \top \ : \ p \ : \ \sigma \vee \sigma \ : \ \neg\sigma \ : \ \langle\pi\rangle\sigma \ : \ \mathrm{loop}(\pi)$$
$$\pi ::= \xrightarrow{\gamma} \ : \ \mathrm{test}\{\sigma\} \ : \ \pi + \pi \ : \ \pi \cdot \pi \ : \ \pi^* \ : \ \pi^{-1} \ : \ \pi \cap \pi$$




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

$$\mathsf{E}\,\langle(\mathrm{test}\{p \vee q\} \cdot \xrightarrow{\prec})^*\rangle r$$

$$\neg\mathsf{E}\,\langle\xrightarrow{\prec}\rangle(p \wedge s)$$

$$\mathsf{E}\,\bigvee_{(d,d') \in (\Gamma\setminus\{\prec\})^2, d \neq d'} \mathrm{loop}(\xrightarrow{d} \cdot \xrightarrow{d'}{}^{-1})$$

$$\mathsf{E}\,\langle\mathrm{test}\{s\} \cdot \xrightarrow{e} \cdot \mathrm{test}\{r\} \cdot \xrightarrow{f} \cdot \mathrm{test}\{s\} \cdot \xrightarrow{d}{}^{-1} \cdot \xrightarrow{c}\rangle p$$

# What logic shall we use?

## Propositional dynamic logic with Intersection & Converse(ICPDL)

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

$$\Phi ::= \mathsf{E}\,\sigma \;:\; \neg\Phi \;:\; \Phi \vee \Phi$$

$$\sigma ::= \top \;:\; p \;:\; \sigma \vee \sigma \;:\; \neg\sigma \;:\; \langle\pi\rangle\sigma \;:\; \mathrm{loop}(\pi)$$

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

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

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

# 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 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

- $\forall u \curvearrowright^{a} v, ts(v) - ts(u) \leq a$
- $\forall u \to v, 0 \leq ts(v) - ts(u)$

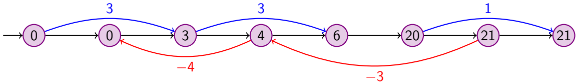
# Solving realizability by modulo counting

**Assume only closed guards: $G$ is realizable**

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

- $\forall u \curvearrowright^{a} v, ts(v) - ts(u) \leq a$
- $\forall u \to v, 0 \leq ts(v) - ts(u)$

$M = 5$

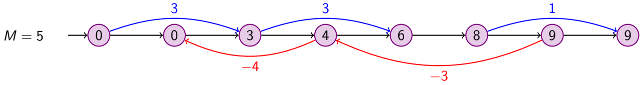


# Solving realizability by modulo counting

**Assume only closed guards: $G$ is realizable**

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

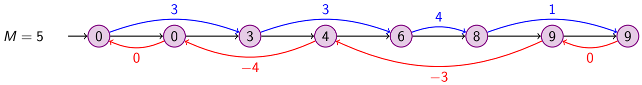
- $\forall u \curvearrowright^{a} v, ts(v) - ts(u) \leq a$
- $\forall u \to v, 0 \leq ts(v) - ts(u) \leq M - 1$, where $M$ is max const



# Solving realizability by modulo counting

**Assume only closed guards: $G$ is realizable**

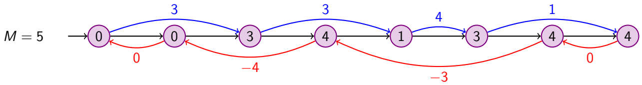
iff $\exists ts : V \to \mathbb{N}$ s.t $\forall u \curvearrowright^a v, ts(v) - ts(u) \leq a$.



# Solving realizability by modulo counting

**Assume only closed guards: $G$ is realizable**

iff $\exists ts : V \to \mathbb{N}$ s.t $\forall u \curvearrowright^a v, ts(v) - ts(u) \le a$.



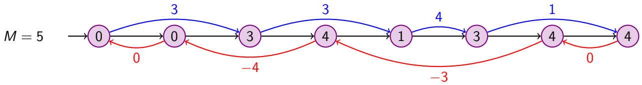

$\exists tsm : V \to \{0, \ldots M-1\}$ s.t $\forall u \curvearrowright^a v$,

- if $u \preceq v$, then $(tsm(v) - tsm(u))[M] \le a$
- if $v \prec u$, then $(tsm(v) - tsm(u))[M] \ge -a$

# Solving realizability by modulo counting

**Assume only closed guards: $G$ is realizable**

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



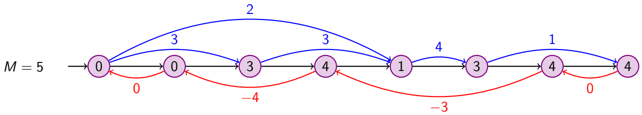

$\exists tsm : V \to \{0, \dots M-1\}$ s.t $\forall u \curvearrowright^a v$,

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

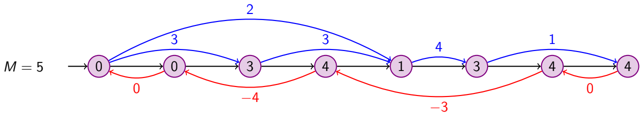



$\exists tsm : V \to \{0, \dots M-1\}$ s.t $\forall u \curvearrowright^a v$,

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



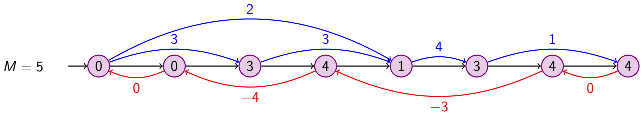

$\exists tsm : V \to \{0, \ldots M-1\}$ s.t $\forall u \curvearrowright^a v$,

- 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**

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



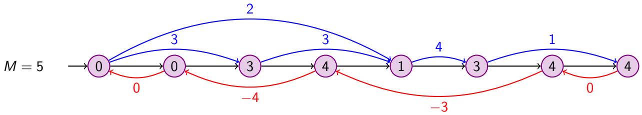

iff $\exists tsm : V \to \{0, \ldots M-1\}$ s.t $\forall u \curvearrowright^{a} v$,

- 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**

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




iff $\exists tsm : V \to \{0, \dots M-1\}$ s.t $\forall u \curvearrowright^{a} v$,

- 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

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

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

# Writing it in EQ-ICPDL

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

- 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

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

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

# Writing it in EQ-ICPDL

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

- 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

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

$$\mathrm{BigPath} = \sum_{\substack{0 \leq i,j,k < M \\ (j-i)[M]+(k-j)[M] \geq M}} \mathrm{test}\{p_i\} \cdot \to^{+} \cdot \mathrm{test}\{p_j\} \cdot \to^{+} \cdot \mathrm{test}\{p_k\} \cdot \to^{*}$$

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

# Writing it in EQ-ICPDL

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

- 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

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

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

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

$$\text{Forward} = \neg \mathsf{E} \bigvee_{-M < \alpha < M} \text{loop}(\text{BigPath} \cdot \xrightarrow{\leq \alpha}{}^{-1}) \wedge \neg \mathsf{E} \bigvee_{\substack{0 \leq i, j < M \\ (j-i)[M] > \alpha}} \text{loop}(\text{test}\{p_i\} \cdot \xrightarrow{\leq \alpha} \cdot \text{test}\{p_j\} \cdot (\to^{-1})^{+})$$

Similarly for Backward, but need to define $\neg$BigPath (see paper)

# What about strict/open guards

**What happens when there are both closed and open guards?**

Realizable $\implies \exists tsm : V \to \{0, \dots M-1\}$ s.t $\forall u \curvearrowright^{a} v$,

- 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

# What about strict/open guards

**What happens when there are both closed and open guards?**

Realizable $\implies \exists tsm : V \to \{0, \dots M-1\}$ s.t $\forall u \curvearrowright^{a} v$,

- 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 assigments.

# What about strict/open guards

## What happens when there are both closed and open guards?

Realizable $\implies \exists tsm : V \to \{0, \ldots M-1\}$ s.t $\forall u \curvearrowright^{a} v$,

- 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 assigments.

## 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).

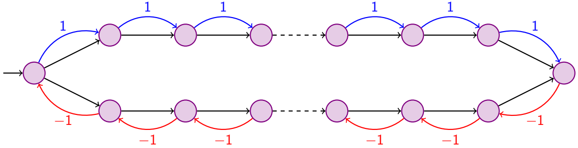
Realizable $= \exists p_1, \ldots, p_{M-1}$ Partition $\wedge$ Forward $\wedge$ Backward $\wedge$ noFracCycle

# Realizability is not MSO definable without the linear order

- Width of a partial order = maximal size of anti-chain.
- Linear order has width 1.

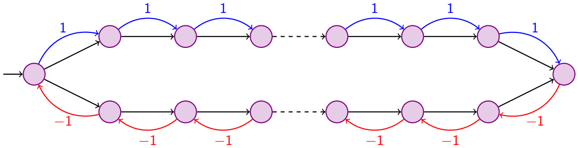
# Realizability is not MSO definable without the linear order

- Width of a partial order = maximal size of anti-chain.
- Linear order has width 1.
- Consider the following example with width 2.



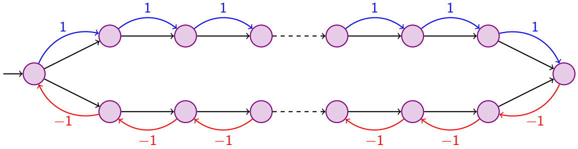
# Realizability is not MSO definable without the linear order

- 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).



# Application and extensions

**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 $next_a$

# 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 $next_a$

- edge between current event and next occurrence of a.

$$\sum_{a \in AP} \mathrm{test}\{(\mathrm{next}_a \lhd \alpha)\} \cdot \to \cdot (\mathrm{test}\{\neg a\} \cdot \to)^* \cdot \mathrm{test}\{a\}$$

# 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

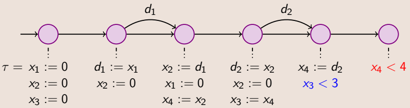



Figure: Intricate flow of information in complex updates.

# 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

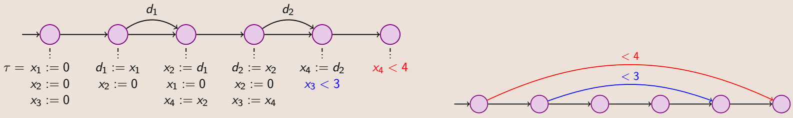



Figure: Intricate flow of information in complex updates.

# 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

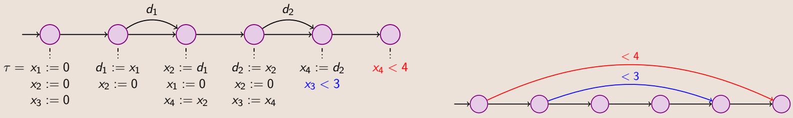



Figure: Intricate flow of information in complex updates.

## 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

# 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?!