ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

TIMED AUTOMATON

TIMED WORD

TIMED WORD LANGUAGE

NON-EMPTINESS / REACHABILITY PROBLEM
EMPTINESS FOR (PUSHDOWN) TIMED AUTOMATON

- Well-studied problem with standard approach
  - Timed automata (TA): Region construction [Alur-Dill’90]
    Many optimizations
  - Pushdown timed automata (PDTA): Lifting region construction
    [Bouajjani et al. ’94] [Abdulla et al. ’12]
- Common feature:
  - represent behaviors as timed words
  - use abstractions to reduce to finite automata
EMPTINESS FOR (PUSHDOWN) TIMED AUTOMATON

- Well-studied problem with standard approach
  - Timed automata (TA): Region construction [Alur-Dill’90]
    Many optimizations
  - Pushdown timed automata (PDTA): Lifting region construction
    [Bouajjani et al. ’94] [Abdulla et al. ’12]
- Common feature:
  - represent behaviors as timed words
  - use abstractions to reduce to finite automata

- Our new approach
  - represent behaviors as graphs: words with timing constraints
  - Interpret graphs in trees to reduce to tree automata
    - Higher level and more powerful formalism
    - Simpler and uniform proofs for more complicated systems
    - New technique not relying on regions/zones
OUTLINE

▸ BEHAVIOURS AS GRAPHS
▸ DECIDING GRAPH PROPERTIES
▸ DEFINABILITY OF PROPERTIES FOR TIMED SYSTEMS
▸ TREE-WIDTH FOR TIMED SYSTEMS
▸ INTERPRETING GRAPHS IN TREES
▸ CONCLUSION
BEHAVIORS AS GRAPHS: TIMED SYSTEMS

TC-WORDS: WORDS WITH TIMING CONSTRAINTS
BEHAVIORS AS GRAPHS: TIMED SYSTEMS

TC-WORDS: WORDS WITH TIMING CONSTRAINTS

TC-WORD LANGUAGE: $\mathcal{L}_{TCW}(\mathcal{A})$

- Every accepting path $\rho$ in the timed system generates one TC-word $\text{tcw}(\rho) \in \mathcal{L}_{TCW}(\mathcal{A})$
BEHAVIORS AS GRAPHS: TIMED SYSTEMS

TC-WORDS: WORDS WITH TIMING CONSTRAINTS

TC-WORD LANGUAGE: \( \mathcal{L}_{TCW}(A) \)

- Every accepting path \( \rho \) in the timed system generates one TC-word \( tcw(\rho) \in \mathcal{L}_{TCW}(A) \)
BEHAVIORS AS GRAPHS: TIMED SYSTEMS

TC-WORDS: WORDS WITH TIMING CONSTRAINTS

TC-WORD LANGUAGE: $\mathcal{L}_{\text{TCW}}(A)$

REALIZABLE TC-WORDS: $\text{Real}_{\text{TCW}}$
BEHAVIORS AS GRAPHS: TIMED SYSTEMS

TC-WORDS: WORDS WITH TIMING CONSTRAINTS

TC-WORD LANGUAGE:

REALIZABLE TC-WORDS:
REALIZATIONS OF TC-WORDS

TC-WORD

REALIZATION

TIMED-WORDS
Analyzing Timed Systems Using Tree Automata

**Timed Words vs TC-Words**

**Timed-words**
- Uncountably many realizations
- No realizations
- Words over an infinite alphabet

**TC-Words**
- One TC-word
- One TC-word
- Graphs over a finite signature

\[ \mathcal{L}_T(\mathcal{A}) = \text{Realizations}(\mathcal{L}_{TCW}(\mathcal{A})) \]

\[ \mathcal{L}_T(\mathcal{A}) \neq \emptyset \quad \iff \quad \mathcal{L}_{TCW}(\mathcal{A}) \cap \text{Real}_{TCW} \neq \emptyset \]

This is a graph property!
OUTLINE

- BEHAVIOURS AS GRAPHS
- DECIDING GRAPH PROPERTIES
- DEFINABILITY OF PROPERTIES FOR TIMED SYSTEMS
- TREE-WIDTH FOR TIMED SYSTEMS
- INTERPRETING GRAPHS IN TREES
- CONCLUSION
COURCELLE’S THEOREM

- Let $TW_k$ be the set of graphs of tree-width at most $k$
- Let $P$ be a property of graphs
- If $P$ is MSO-definable then $P \cap TW_k \neq \emptyset$ is decidable

- Graphs in $TW_k$ can be interpreted in trees (k-terms)
- Let $P$ be an MSO-definable property of graphs
- $\Phi_P$ MSO over graphs $\iff \Phi^k_P$ MSO over trees (k-terms)
- Then $P \cap TW_k \neq \emptyset$ iff $\Phi^k_P$ satisfiable over trees (k-terms)

THATCHER&WRIGHT’68: REDUCTION TO EMPTINESS OF TREE AUTOMATA
COURCELLE’S THEOREM

- Let $TW_k$ be the set of graphs of tree-width at most $k$
- Let $P$ be a property of graphs
- If $P$ is MSO-definable then $P \cap TW_k \neq \emptyset$ is decidable

Graphs in $TW_k$ can be interpreted in trees ($k$-terms)
- Let $P$ be a property of graphs
- Build directly a tree automaton $A^k_P$ accepting $k$-terms denoting graphs satisfying $P$
- Then $P \cap TW_k \neq \emptyset$ iff $L(A^k_P) \neq \emptyset$
Courcelle’s Theorem

- Let $TW_k$ be the set of graphs of tree-width at most $k$
- Let $P$ be a property of graphs
- If $P$ is MSO-definable then $P \cap TW_k \neq \emptyset$ is decidable

- Graphs in $TW_k$ can be interpreted in trees (k-terms)
- Show that $P$ is definable by some $\Phi_P$ in some efficient logic
- Translate efficiently formula $\Phi_P$ to some tree automaton $A^k_P$ accepting k-terms denoting graphs satisfying $P$
- Then $P \cap TW_k \neq \emptyset$ iff $L(A^k_P) \neq \emptyset$
LCPDL: PROPOSITIONAL DYNAMIC LOGIC WITH LOOP & CONVERSE

\[ \phi ::= A \phi \mid \phi \lor \phi \mid \neg \phi \]

\[ \varphi ::= a \mid \varphi \lor \varphi \mid \neg \varphi \mid \langle \pi \rangle \varphi \mid \text{Loop}(\pi) \]

\[ \pi ::= \{ \varphi \}? \mid \rightarrow \mid \leftarrow \mid \bowtie \mid \bowtie \mid \pi \cdot \pi \mid \pi + \pi \mid \pi^* \]

**THEOREM**

Based on Göller, Lohrey & Lutz '09

For each LCPDL formula \( \phi \) we can construct a 2-way alternating tree automaton of size Poly\( (k, \phi) \) accepting all k-terms denoting graphs satisfying \( \phi \)

We obtain a non-deterministic tree automaton of size \( 2^{\text{Poly}(k, \phi)} \)
COURCELLE’S THEOREM

- Let $\mathcal{TW}_k$ be the set of graphs of tree-width at most $k$
- Let $P$ be a property of graphs
- If $P$ is MSO-definable then $P \cap \mathcal{TW}_k \neq \emptyset$ is decidable

WE WANT TO SOLVE $\mathcal{L}_{TCW}(A) \cap \text{Real}^{TCW} \neq \emptyset$

- Show that TC-words have bounded tree-width
- Show that our properties are MSO-definable
- Show that our properties are LCPDL-definable
- Build directly tree automata for our properties
P. Madhusudan & G. Parlato, POPL’11
The tree-width of auxiliary storage

C. Aiswarya, PG & K. Narayan Kumar, CONCUR’12
MSO decidability of multi-pushdown systems via split-width

C. Aiswarya PhD’14
Verification of communicating recursive programs via split-width

C. Aiswarya & PG, FSTTCS’14
Reasoning about distributed systems: WYSIWYG
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

OUTLINE

▸ BEHAVIOURS AS GRAPHS
▸ DECIDING GRAPH PROPERTIES
▸ DEFINABILITY OF PROPERTIES FOR TIMED SYSTEMS
▸ TREE-WIDTH FOR TIMED SYSTEMS
▸ INTERPRETING GRAPHS IN TREES
▸ CONCLUSION
**MSO-DEFINABLE GRAPH PROPERTIES**

\[ G = (V, \rightarrow) \text{ is a word: linear order} \]

Word(\[\rightarrow\]) ::= \forall x, y, z \left( \neg (x \rightarrow^+ x) \land (x = y \lor x \rightarrow^+ y \lor y \rightarrow^+ x) \land \neg (x \rightarrow z \lor x \rightarrow y \rightarrow^+ z) \right)
**MSO-DEFINABLE GRAPH PROPERTIES**

\( G = (V, \rightarrow, \xleftarrow{\sim}) \) IS A 1-CLOCK TC-WORD

**Forward** \( (\xleftarrow{\sim}) \) ::= \( \forall x, y \ (x \xleftarrow{\sim} y \implies x < y) \)

**Clock** \( (\xleftarrow{\sim}) \) ::= \( \neg \exists x, y, x', y' \ (x \xleftarrow{\sim} y \land x' \xleftarrow{\sim} y' \land x < x' < y) \)

Forward ::= A \neg \text{Loop}(\xleftarrow{\sim} \cdot \rightarrow^*)

Clock ::= A \neg \text{Loop}(\xleftarrow{\sim} \cdot \leftarrow^+ \cdot \{\langle \xleftarrow{\sim} \rangle\} \cdot ? \cdot \leftarrow^+ \cdot \rightarrow^*)
$G = (V, \rightarrow, \circ)$ IS A 1-CLOCK TC-WORD

**Forward** ($\circ$) ::= $\forall x, y \ (x \circ y \implies x < y)$

**Clock** ($\circ$) ::= $\neg \exists x, y, x', y' \ (x \circ y \land x' \circ y' \land x < x' < y)$

Forward ::= $A \neg \text{Loop}(\circ \cdot \rightarrow^*)$

Clock ::= $A \neg \text{Loop}(\circ \cdot \leftarrow^+ \cdot \{\langle \circ \rangle\}? \cdot \leftarrow^+)$
\[ G = (V, \rightarrow, \curlywedge) \] is a 1-clock TC-word

- **Forward** \( (\curvearrowright) : \) : \( \forall x, y \ (x \curvearrowright y \implies x < y) \)
- **Clock** \( (\curvearrowright) : \) : \( \neg \exists x, y, x', y' \ (x \curvearrowright y \land x' \curvearrowright y' \land x < x' < y) \)

Forward \( \::=:\ \text{A } \neg\text{Loop}(\curvearrowright \cdot \rightarrow^*) \)

Clock \( \::=:\ \text{A } \neg\text{Loop}(\curvearrowright \cdot \leftrightarrow^+ \cdot \{\langle \curvearrowright \rangle\}\? \cdot \leftrightarrow^+) \)
MSO-DEFINABLE GRAPH PROPERTIES

$G = (V, \rightarrow, \looparrowright)$ IS A 1-STACK TC-WORD

Forward$(\looparrowright) := \forall x, y \ (x \looparrowright y \implies x < y)$

Stack$(\looparrowright) := \neg \exists x, y, x', y' \ (x \looparrowright y \land x' \looparrowright y' \land (y = x' \lor x \leq x' < y < y' \lor x < x' < y \leq y'))$

Forward ::= A \neg Loop(\looparrowright \cdot \rightarrow^*)

Stack ::= A \neg \langle \looparrowright \cdot \looparrowright \rangle \land A \neg Loop(\looparrowright \cdot \rightarrow^+ \cdot \looparrowright + \looparrowright \cdot \rightarrow^+ \cdot \looparrowright) \land A \{\langle \looparrowright \rangle\} \implies Loop(\looparrowright \cdot \leftarrow \cdot (\looparrowright \cdot \leftarrow + \{\neg \langle \looparrowright + \looparrowright \rangle\} \cdot \leftarrow)^*)
MSO-DEFINABLE GRAPH PROPERTIES

\( G = (V, \rightarrow, \curvearrowright) \) IS A 1-STACK TC-WORD

Forward(\( \curvearrowright \)) ::= \( \forall x, y \ (x \curvearrowright y \implies x < y) \)

Stack(\( \curvearrowright \)) ::= \( \neg \exists x, y, x', y' \ (x \curvearrowright y \wedge x' \curvearrowright y' \wedge (y = x' \vee x \leq x' < y < y' \vee x < x' < y < y')) \)

\[\checkmark\]

Forward ::= \( A \neg \text{Loop}(\curvearrowright \cdot \rightarrow^*) \)

Stack ::= \( A \neg \langle \curvearrowright \cdot \curvearrowright \rangle \wedge A \neg \text{Loop}(\curvearrowright \cdot \rightarrow^+ \cdot \curvearrowright + \langle \cdot \rightarrow^+ \cdot \curvearrowright \rangle) \wedge A \langle \cdot \rangle \? \implies \text{Loop}(\curvearrowright \cdot \leftarrow \cdot (\langle \cdot \leftarrow + \{\neg \langle \cdot + \langle \cdot \rangle \} \? \cdot \leftarrow \} ^*) \)

\[\times\]
MSO-DEFINABLE GRAPH PROPERTIES

\[ G = (V, \rightarrow, \rightleftharpoons) \text{ IS A 1-STACK TC-WORD} \]

Forward(\rightleftharpoons) ::= \forall x, y (x \rightleftharpoons y \implies x < y)

Stack(\rightleftharpoons) ::= \neg \exists x, y, x', y' (x \rightleftharpoons y \land x' \rightleftharpoons y' \land (y = x' \lor x \leq x' < y < y' \lor x < x' < y < y'))

Forward ::= A \neg \text{Loop}(\rightleftharpoons \cdot \rightarrow^*)

Stack ::= A \neg (\rightleftharpoons \cdot \rightleftharpoons) \land A \neg \text{Loop}(\rightleftharpoons \cdot \rightarrow^+ \cdot \rightleftharpoons + \rightleftharpoons \cdot \rightarrow^+ \cdot \rightleftharpoons) 
\land A \{\langle \rightleftharpoons \rangle\}? \implies \text{Loop}(\rightleftharpoons \cdot \leftarrow \cdot (\rightleftharpoons \cdot \leftarrow + \{\neg \langle \rightleftharpoons + \rightleftharpoons \rangle\}? \cdot \leftarrow)^*)
MSO-DEFINABLE GRAPH PROPERTIES

\( G = (V, \to, \gamma) \) IS AN M-STACKS N-CLOCKS TC-WORD

Forward(\( \gamma \)) \& \exists \overline{R} = (R_1, \ldots, R_n) \exists \overline{S} = (S_1, \ldots, S_m)

\( \forall i = 1 \) Clock(\( R_i \)) \& \( \forall i = 1 \) Stack(\( S_i \))
\[ G = (V, \rightarrow, \leftarrow) \text{ is an M-stacks n-clocks TC-word} \]

Forward(\sim) \land \exists \overline{R} = (R_1, \ldots, R_n) \exists \overline{S} = (S_1, \ldots, S_m)

Partition(\sim, \overline{R}, \overline{S}) \land \bigwedge_{i=1}^{n} \text{Clock}(R_i) \land \bigwedge_{i=1}^{m} \text{Stack}(S_i)
G = (V, →, \rightsquigarrow) IS AN M-STACKS N-CLOCKS TC-WORD

\begin{align*}
\text{Forward}(\rightsquigarrow) & \land \exists \overline{R} = (R_1, \ldots, R_n) \land \exists \overline{S} = (S_1, \ldots, S_m) \\
\text{Partition}(\rightsquigarrow, \overline{R}, \overline{S}) & \land \bigwedge_{i=1}^{n} \text{Clock}(R_i) \land \bigwedge_{i=1}^{m} \text{Stack}(S_i)
\end{align*}
MSO-DEFINABLE GRAPH PROPERTIES

\[ \forall X = (X_{\delta})_{\delta \in \Delta} \text{ Partition}(X) \land \text{AcceptingPath}(X) \]

\[ \land \exists (c \rightsucccurlyeq)_{c \in \text{Clocks}} \text{ Partition}(\rightsucccurlyeq, (c \rightsucccurlyeq)_{c \in \text{Clocks}}) \]

\[ \land \bigwedge_{c \in \text{Clocks}} \forall x, y \left( x \xleftarrow{c} y \implies \text{Reset}_{c}(x) \land \neg \exists z \left( x < z < y \land \text{Reset}_{c}(z) \right) \right) \]

\[ \land \bigwedge_{\delta \in \Delta} \forall y \left( X_{\delta}(y) \implies \exists x \left( x \xleftarrow{c} y \land x \xrightarrow{I} y \right) \right) \]

\[ \text{Reset}_{c}(x) ::= \bigvee_{\delta \in \Delta} X_{\delta}(x) \quad \rightsucccurlyeq ::= \bigcup_{I} \rightsucccurlyeq^{I} \]
**MSO-DEFINABLE GRAPH PROPERTIES**

\[ G = (V, \to, \rightsquigarrow) \text{ is a TC-word accepted by a TA } A \]

\[ \exists X = (X_\delta)_{\delta \in \Delta} \text{ Partition}(X) \land \text{AcceptingPath}(X) \land \exists (c \rightsquigarrow)_{c \in \text{Clocks}} \text{ Partition}(\rightsquigarrow, (c \rightsquigarrow)_{c \in \text{Clocks}}) \land \bigwedge_{c \in \text{Clocks}} \forall x, y \left( x \rightsquigarrow y \implies \text{Reset}_c(x) \land \neg \exists z \ x < z < y \land \text{Reset}_c(z) \right) \land \bigwedge_{\delta \in \Delta} \left( (c : 0) \in \delta \right) \forall y \left( X_\delta(y) \implies \exists x \ (x \rightsquigarrow y \land x \rightsquigarrow^I y) \right) \]

\[ \text{Reset}_c(x) ::= \bigvee_{\delta \in \Delta} \left( X_\delta(x) \right) \land \left( (c : 0) \in \delta \right) \land \rightsquigarrow ::= \bigcup_I \rightsquigarrow^I \]
**MSO-DEFINABLE GRAPH PROPERTIES**

\[ G = (V, \to, \curvearrowright) \] is a TC-word accepted by a TA \( A \)

\[ \exists X = (X_\delta)_{\delta \in \Delta} \text{ Partition}(X) \land \text{AcceptingPath}(X) \land \exists (c \curvearrowright)_{c \in \text{Clocks}} \text{ Partition}(\curvearrowright, (c \curvearrowright)_{c \in \text{Clocks}}) \land \forall c \in \text{Clocks} \left( x \overset{c}{\rightarrow} y \implies \text{Reset}_c(x) \land \neg \exists (z \mid x < z < y \land \text{Reset}_c(z)) \right) \land \forall y \left( X_\delta(y) \implies \exists x \left( x \overset{c}{\rightarrow} y \land x \overset{I}{\rightarrow} y \right) \right) \land \forall \delta \in \Delta \land (c \in I) \in \delta \left( c \in I \right) \in \delta \]

\[ \text{Reset}_c(x) ::= \bigvee_{\delta \in \Delta} X_\delta(x) \quad \curvearrowright ::= \biguplus_I \curvearrowright^I \]
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

**MSO-DEFINABLE GRAPH PROPERTIES**

\[ G = (V, \rightarrow, \rightarrow) \text{ IS REALIZABLE} \]

\[ \text{NO NEGATIVE CYCLE IN THE WEIGHTED GRAPH} \]

![Graph Diagram](Image)
MSO-DEFINABLE GRAPH PROPERTIES

$G = (V, \to, \looparrowright)$ IS REALIZABLE

NO NEGATIVE CYCLE IN THE WEIGHTED GRAPH

THEOREM: REALIZABILITY OF TC-WORDS IS MSO-DEFINABLE

WORK IN PROGRESS
MSO-DEFINABLE GRAPH PROPERTIES

\[
G = (V, \rightarrow, \sim) \text{ IS REALIZABLE}
\]

REALIZABILITY IS NOT MSO-DEFINABLE WITHOUT THE LINEAR ORDER
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

**COURCELLE’S THEOREM**

- Let $TW_k$ be the set of graphs of tree-width at most $k$
- Let $P$ be a property of graphs
- If $P$ is MSO-definable then $P \cap TW_k \neq \emptyset$ is decidable

**WE WANT TO SOLVE** $L_{TCW}(A) \cap \text{Real}_{TCW} \neq \emptyset$

- Show that TC-words have bounded tree-width
- Show that our properties are MSO-definable
- Show that our properties are LCPDL-definable
- Build directly tree automata for our properties

CONCUR’16 SPLIT-WIDTH

CONCUR’16
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

OUTLINE

▸ BEHAVIOURS AS GRAPHS
▸ DECIDING PROPERTIES OF GRAPHS
▸ DEFINABILITY OF PROPERTIES FOR TIMED SYSTEMS
▸ TREE-WIDTH FOR TIMED SYSTEMS
▸ INTERPRETING GRAPHS IN TREES
▸ CONCLUSION
TREE-WIDTH ALGEBRA

\[ \tau ::= i \mid i \rightarrow j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau \]

ATOMIC
TREE-WIDTH ALGEBRA

\[ \tau ::= i \mid i \rightarrow j \mid fg_i(\tau) \mid \tau \oplus \tau \]
TREE-WIDTH ALGEBRA

\[
\tau ::= i \mid i \rightarrow j \mid f_{g_i}(\tau) \mid \tau \oplus \tau
\]
TREE-WIDTH ALGEBRA

\[ \tau ::= i \mid i \xrightarrow{} j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \]

ATOMIC

FORGET

COMBINE
TREE-WIDTH ALGEBRA

\[ \tau ::= i \mid i \to j \mid \text{fg}_{i}(\tau) \mid \tau \oplus \tau \]

ATOMIC

FORGET

COMBINE
TREE-WIDTH ALGEBRA

\[ \tau : = i \mid i \rightarrow j \mid \text{fg}_{\tau} \mid \tau \oplus \tau \]

\( \tau \) : graphs of tree-width at most \( k \)

\( \text{TW}_k \) : graphs of tree-width at most \( k \)

**Atomic**

\( \bullet \)

\( \bullet \rightarrow \bullet \)

\( \bullet \rightarrow \bullet \rightarrow \bullet \)

**Forget**

\( \bullet \)

Graph \( G \) has tree-width at most \( k \) if it can be constructed using \( k+1 \) colors.

**Combine**

\( \oplus \)

Graph \( G \) has tree-width at most \( k \) if it can be constructed using \( k+1 \) colors.
\( \tau ::= i \mid i \xrightarrow{} j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \)
TREE-WIDTH ALGEBRA

\[ \tau ::= i \mid i \rightarrow j \mid fg_i(\tau) \mid \tau \oplus \tau \]
TREE-WIDTH ALGEBRA

\[
T ::= i \mid i \rightarrow j \mid f_{g_i}(T) \mid T \oplus T
\]
Tree-Width Algebra

\[ \tau ::= i \mid i \xrightarrow{\tau} j \mid \text{fg}_{i}(\tau) \mid \tau \oplus \tau \]
TREE-WIDTH ALGEBRA

\[
\tau ::= i \mid i \quad j \mid fg_i(\tau) \mid \tau \oplus \tau
\]
\[ \tau ::= i \mid i \xrightarrow{\quad} j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \]
\[ \tau ::= i \mid i \rightarrow j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau \]

ATOMIC

**FORGET**

**ADD**

**COMBINE**

**DIVIDE**
1-STACK TC-WORDS $\subseteq TW_2$

$$\tau ::= i \mid i \rightarrow j \mid i \leftarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$$
1-STACK TC-WORDS $\subseteq \mathbb{TW}_2$

$$\tau ::= i \mid i \rightarrow j \mid i \bowtie j \mid fg_i(\tau) \mid \tau \oplus \tau$$
1-STACK TC-WORDS $\subseteq TW_2$

$G = (V, \rightarrow, \rightarrow) \text{ TC-WORD}$

$$\tau ::= i \mid i \to j \mid i \bowtie j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau$$
1-STACK TC-WORDS $\subseteq TW_2$

$\mathcal{G} = (V, \rightarrow, \rightsquigarrow) \text{ TC-WORD}$

$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid f_{g_i}(\tau) \mid \tau \oplus \tau$
1-STACK TC-WORDS $\subseteq TW_2$

$G = (V, \rightarrow, \rightsquigarrow)$ TC-WORD

$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$
1-STACK TC-WORDS \( \subseteq \text{TW}_2 \)

\[ \tau ::= i \mid i \rightarrow j \mid i \bowtie j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau \]

\[ G = (V, \rightarrow, \bowtie) \text{ TC-WORD} \]
1-STACK TC-WORDS $\subseteq TW_2$

$G = (V, \rightarrow, \curvearrowright) TC\text{-WORD}$

$$\tau ::= i \mid i \rightarrow j \mid i \curvearrowright j \mid fg_i(\tau) \mid \tau \oplus \tau$$
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

$1$-STACK TC-WORDS $\subseteq \text{TW}_2$

$\tau ::= i \mid i \rightarrow j \mid i \rightleftharpoons j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau$

$G = (V, \rightarrow, \rightsquigarrow) \text{ TC-WORD}$
1-STACK TC-Words $\subseteq \mathbf{TW}_2$

$$G = (V, \rightarrow, \rightsquigarrow) \text{ TC-WORD}$$

$$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$$
1-STACK TC-WORDS ⊆ TW₂

G = (V, →, ↷) TC-WORD

\[ \tau ::= i \mid i \to j \mid i ↷ j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau \]
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

1-STACK TC-WORDS ⊆ TW₂

G = (V, →, ⊸) TC-WORD

\[ \tau ::= i \mid i \rightarrow j \mid i \overset{\tau}{\rightarrow} j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \]
1-STACK TC-WORDS $\subseteq TW_2$

$G = (V, \rightarrow, \rightsquigarrow)$ TC-WORD

$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

$k$-CLOCKS TC-WORDS $\subseteq \mathbf{TW}_{k+1}$

$$\tau ::= i \mid i \to j \mid i \leftarrow j \mid \mathbf{fg}_i(\tau) \mid \tau \oplus \tau$$

$\mathbf{G} = (V, \to, \curvearrowright)$ TC-WORD
\( k\text{-CLOCKS TC-WORDS} \subseteq \mathcal{TW}_{k+1} \)

\[ G = (V, \rightarrow, \tau) \text{ TC-WORD} \]

\[ \tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid f_{i\rightarrow j}(\tau) \mid \tau \oplus \tau \]
Analyzing Timed Systems Using Tree Automata

$k$-Clocks TC-Words $\subseteq TW_{k+1}$

\[
\tau ::= i \mid i \to j \mid i \curvearrowright j \mid fg_i(\tau) \mid \tau \oplus \tau
\]
\[
\begin{align*}
\mathcal{G} &= (V, \rightarrow, \rightsquigarrow) \\
\text{k-CLOCKS TC-WORDS} &\subseteq \mathcal{T}W_{k+1} \\
\tau &::= i | i \rightarrow j | i \rightsquigarrow j | f_{g_i}(\tau) | \tau \oplus \tau
\end{align*}
\]
$k$-CLOCKS TC-WORDS $\subseteq TW_{k+1}$

$$\tau ::= i \mid i \rightarrow j \mid i \leftarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$$
The text contains a formal definition of a timed system using tree automata. The definition is given by the grammar:

\[ \tau ::= i \mid i \rightarrow j \mid i \rightleftharpoons j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \]

Here, \( \tau \) represents a term in the language of the grammar, and the symbols represent different operations and actions in the system. The diagram visualizes the structure and transitions of the system, with nodes and edges indicating the flow of the system.
\( \mathbf{G} = (V, \rightarrow, \rightleftharpoons) \) TC-WORD

\[ \tau ::= i \mid i \rightarrow j \mid i \rightleftharpoons j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \]

\( k \)-CLOCKS TC-WORDS \( \subseteq \mathbf{TW}_{k+1} \)
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

**k-CLOCKS TC-WORDS** \( \subseteq TW_{k+1} \)

\[ \tau ::= i \mid i \rightarrow j \mid i \circlearrowright j \mid fg_i(\tau) \mid \tau \oplus \tau \]
k-CLOCKS TC-WORDS $\subseteq TW_{k+1}$

$G = (V, \rightarrow, \vdash) \text{ TC-WORD}$

$\tau ::= i \mid i \rightarrow j \mid i \vdash j \mid f_{g_i}(\tau) \mid \tau \oplus \tau$
$\kappa$-CLOCKS TC-WORDS $\subseteq TW_{k+1}$

$G = (V, \rightarrow, \rtimes)$ TC-WORD

$\tau ::= i \mid i \rightarrow j \mid i \rtimes j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau$
\( k\)-CLOCKS TC-WORDS \( \subseteq TW_{k+1} \)

\[ \tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid fg_i(\tau) \mid \tau \oplus \tau \]
1-STACK k-CLOCKS TC-WORDS ⊆ TW_{3k+2}  \quad G = (V, \rightarrow, \rightsquigarrow) TC-WORD

\[ \tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \]
1-STACK k-CLOCKS TC-WORDS ⊆ TW_{3k+2} \quad G = (V, \to, \rightleftharpoons) \text{ TC-WORD}

\[ \tau ::= i \mid i \to j \mid i \rightleftharpoons j \mid fg_i(\tau) \mid \tau \oplus \tau \]
1-STACK $k$-CLOCKS TC-WORDS $\subseteq TW_{3k+2}$

$G = (V, \rightarrow, \rightarrowrightarrow)$ TC-WORD

$\tau ::= i \mid i \rightarrow j \mid i \rightarrowrightarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$
1-STACK k-CLOCKS TC-WORDS $\subseteq \mathbf{TW}_{3k+2}$

$G = (V, \rightarrow, \rightsquigarrow) \text{ TC-WORD}$

$$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau$$
1-STACK k-CLOCKS TC-WORDS $\subseteq \mathbb{TW}_{3k+2}$

$G = (V, \rightarrow, \circlearrowright)$ TC-WORD

$\tau ::= i \mid i \rightarrow j \mid i \circlearrowright j \mid fg_i(\tau) \mid \tau \oplus \tau$
1-STACK $k$-CLOCKS TC-WORDS $\subseteq \text{TW}_{3k+2}$

$G = (V, \rightarrow, \rightsquigarrow)$ TC-WORD

$$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau$$
1-STACK $k$-CLOCKS TC-WORDS $\subseteq \text{TW}_{3k+2}$

$G = (V, \rightarrow, \cdot \cdot \cdot) \text{ TC-WORD}$

$$\tau ::= i \mid i \rightarrow j \mid i \cdot \cdot \cdot j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau$$
1-STACK k-CLOCKS TC-WORDS \( \subseteq TW_{3k+2} \)

\[
\begin{align*}
\tau &::= i | i \to j | i \rightleftharpoons j | fg_i(\tau) | \tau \oplus \tau
\end{align*}
\]
1-STACK $k$-CLOCKS TC-WORDS $\subseteq TW_{3k+2}$

$G = (V, \rightarrow, \rightsquigarrow) \text{ TC-WORD}$

$$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid f_{g_i}(\tau) \mid \tau \oplus \tau$$
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

1-STACK k-CLOCKS TC-WORDS $\subseteq TW_{3k+2}$

$G = (V, \rightarrow, \rightarroweq)$ TC-WORD

$$\tau ::= i \mid i \rightarrow j \mid i \rightarroweq j \mid fg_i(\tau) \mid \tau \oplus \tau$$
ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

$1$-STACK $k$-CLOCKS TC-WORDS $\subseteq \text{TW}_{3k+2}$

$G = (V, \rightarrow, \rightsquigarrow)$ TC-WORD

$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$
**ANALYZING TIMED SYSTEMS USING TREE AUTOMATA**

**1-STACK $k$-CLOCKS TC-WORDS $\subseteq TW_{3k+2}$**

$G = (V, \rightarrow, \rightsquigarrow)$ TC-WORD

$$\tau ::= i \mid i \rightarrow j \mid i \rightsquigarrow j \mid fg_i(\tau) \mid \tau \oplus \tau$$
$\tau ::= i \mid i \rightarrow j \mid i \rightarrow j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau$
1-STACK k-CLOCKS TC-WORDS $\subseteq TW_{3k+2}$

$G = (V, \rightarrow, \xrightarrow{	au})$ TC-WORD

$\tau ::= i \mid i \rightarrow j \mid i \xrightarrow{\tau} j \mid \mathbb{f}_{g_i}(\tau) \mid \tau \oplus \tau$

- At most one colored reset node on the left for each clock
- Last reset node colored for each clock
- First and last points are colored
- $k+1$ extra colors to maintain this invariant
Courcelle’s Theorem

- Let $TW_k$ be the set of graphs of tree-width at most $k$
- Let $P$ be a property of graphs
- If $P$ is MSO-definable then $P \cap TW_k \neq \emptyset$ is decidable

We want to solve $L_{TCW}(\mathcal{A}) \cap \text{Real}_{TCW} \neq \emptyset$

- Show that TC-words have bounded tree-width
- Show that our properties are MSO-definable
- Show that our properties are LCPDL-definable
- Build directly tree automata for our properties
OUTLINE

▸ BEHAVIOURS AS GRAPHS
▸ DECIDING PROPERTIES OF GRAPHS
▸ DEFINABILITY OF PROPERTIES FOR TIMED SYSTEMS
▸ TREE-WIDTH FOR TIMED SYSTEMS
▸ INTERPRETING GRAPHS IN TREES
▸ CONCLUSION
TREE INTERPRETATION

\[ \tau ::= i \mid i \xrightarrow{j} j \mid fg_i(\tau) \mid \tau \oplus \tau \]
TREE INTERPRETATION

\[ \tau ::= i | j | \text{fg}_i(\tau) | \tau \oplus \tau \]

- Edge = leaf
TREE INTERPRETATION

\[ \tau ::= i | i \xrightarrow{\cdot} j | f g_i(\tau) | \tau \oplus \tau \]

- Edge = leaf
- Vertex = leaf + \text{color}
TREE INTERPRETATION

\[ \tau ::= i \mid i \xrightarrow{} j \mid f_{g_i}(\tau) \mid \tau \oplus \tau \]

- Edge = leaf
- Vertex = leaf + color
- One vertex = several leaves
TREE INTERPRETATION

- Edge = leaf
- Vertex = leaf + color
- One vertex = several leaves
**Tree Interpretation**

- Edge = leaf
- Vertex = leaf + color
- One vertex = several leaves
- SameVertex$_i(x,y)$

\[
\text{SameVertex}_i(x,y) ::= \exists z \left( z < x \land z < y \land \forall z' \left( (z < z' < x \lor z < z' < y) \implies \neg f_{g_i}(z') \right) \right)
\]
Tree Interpretation

- Edge = leaf
- Vertex = leaf + color
- One vertex = several leaves
- SameVertex\(_i(x,y)\)

\[
\text{SameVertex}_i(x, y) ::= \exists z \left( z < x \land z < y \land \forall z' \left( (z < z' < x \lor z < z' < y) \implies \neg f_{g_i}(z') \right) \right)
\]
TREE INTERPRETATION

\[ \tau ::= i \mid i \rightarrow j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau \]

- Edge = leaf
- Vertex = leaf + color
- One vertex = several leaves
- \( \text{SameVertex}_i(x, y) \)

\[
\text{SameVertex}_i(x, y) ::= \exists z \left( z < x \land z < y \land \forall z' \left( (z < z' < x \lor z < z' < y) \implies \neg \text{fg}_i(z') \right) \right)
\]
TREE INTERPRETATION

- Edge = leaf
- Vertex = leaf + color
- One vertex = several leaves
- \( \text{SameVertex}_i(x,y) \)

\[
\tau ::= i \mid i \xrightarrow{\quad} j \mid \text{fg}_i(\tau) \mid \tau \oplus \tau
\]

\[
\text{SameVertex}_i(x,y) ::= \exists z \left( z < x \land z < y \land \forall z' \left( (z < z' < x \lor z < z' < y) \implies \neg \text{fg}_i(z') \right) \right)
\]
DIRECTLY BUILDING TREE AUTOMATA

We can build a tree automaton $A_{\text{valid}}^k$ of size $2^{O(k^2)}$ which accepts all $k$-terms denoting valid TCWs.

We can build a tree automaton $A_{\text{real}}^{k,M}$ of size $M^{\text{poly}(k)}$ which accepts all $k$-terms denoting realizable TCWs using constants at most $M$.

Let $S$ be a pushdown timed automaton with set of clocks $X$. Let $|S|$ be its size (constants encoded in unary) and $k = 3|X| + 2$.

We can build a tree automaton $A_S^k$ of size $|S|^{\text{poly}(k)}$ which accepts all $k$-terms denoting TCWs in $L_{\text{TCW}}(S)$.

NON EMPTINESS / REACHABILITY \[ \mathcal{L}(S) \neq \emptyset \iff \mathcal{L}(A_{\text{valid}}^k \cap A_{\text{real}}^{k,M} \cap A_S^k) \neq \emptyset \]
COURCELLE’S THEOREM

- Let $\mathcal{TW}_k$ be the set of graphs of tree-width at most $k$
- Let $P$ be a property of graphs
- If $P$ is MSO-definable then $P \cap \mathcal{TW}_k \neq \emptyset$ is decidable

WE WANT TO SOLVE $\mathcal{L}_{TCW}(A) \cap \text{Real}_{TCW} \neq \emptyset$

- Show that TC-words have bounded tree-width ✓
- Show that our properties are MSO-definable ✓
- Show that our properties are LCPDL-definable ✓
- Build directly tree automata for our properties ✓
OUTLINE

▸ BEHAVIOURS AS GRAPHS
▸ DECIDING PROPERTIES OF GRAPHS
▸ DEFINABILITY OF PROPERTIES FOR TIMED SYSTEMS
▸ TREE-WIDTH FOR TIMED SYSTEMS
▸ INTERPRETING GRAPHS IN TREES
▸ CONCLUSION
CONCLUSION

ANALYZING TIMED SYSTEMS USING TREE AUTOMATA

NEW TECHNIQUE FOR ANALYZING TIMED SYSTEMS

1. Write behaviors as graphs with timing constraints
2. Show a bound on tree-width for these graphs
3. Show MSO-definability of the relevant properties, or
4. Build Tree automata directly
CONCLUSION

NEW TECHNIQUE FOR ANALYZING TIMED SYSTEMS

1. Write behaviors as graphs with timing constraints
2. Show a bound on tree-width for these graphs
3. Show MSO-definability of the relevant properties, or
4. Build Tree automata directly

RESULTS

- PSPACE decision procedure for timed automata
- EXPTIME decision procedure for pushdown timed automata
- EXPTIME decision procedure for multi-pushdown timed automata with bounded rounds
CONCLUSION

1. Write behaviors as graphs with timing constraints
2. Show a bound on tree-width for these graphs
3. Show MSO-definability of the relevant properties, or
4. Build Tree automata directly

FUTURE WORK

- Concurrent recursive timed programs
- MSO-definability of realizability
- Model-Checking wrt. timed specifications
THANK YOU