Model Checking for Real-time systems

using

Generalized Timed Automata

R. Govind

IMSc Chennai

S. Akshay

Paul Gastin

B. Srivathsan

QuantFormal Workshop

FSTTCS 2025

Formal Verification of Real-Time Systems

Overview

Model Checking for Real-time systems

Motivations for our approach

Generalised Timed Automata (GTA)

Metric Interval Temporal Logic (MITL)

Logic to Automata translation

MITL to GTA

Uses powerful features of GTA

Exponentially more succinct than the state-of-the-art

Improvements to Translation

Towards more Determinism

No time-abstract bisimulation for GTAs

A new zone-based algorithm for checking Büchi non-emptiness in GTAs

Model Checking

Does **System S** satisfy **Property P** ?

modelled using

Automata

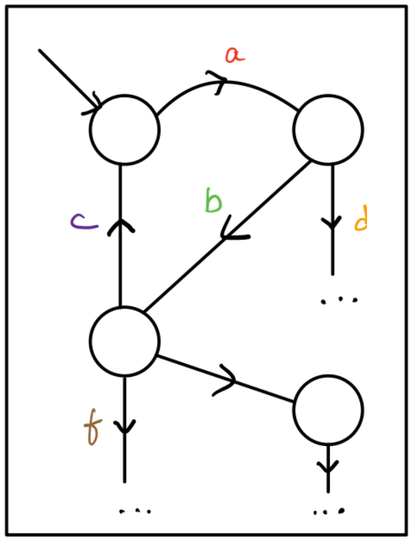
specified using

Logical formula

Linear Temporal Logic

Model

$\mathscr{A}$

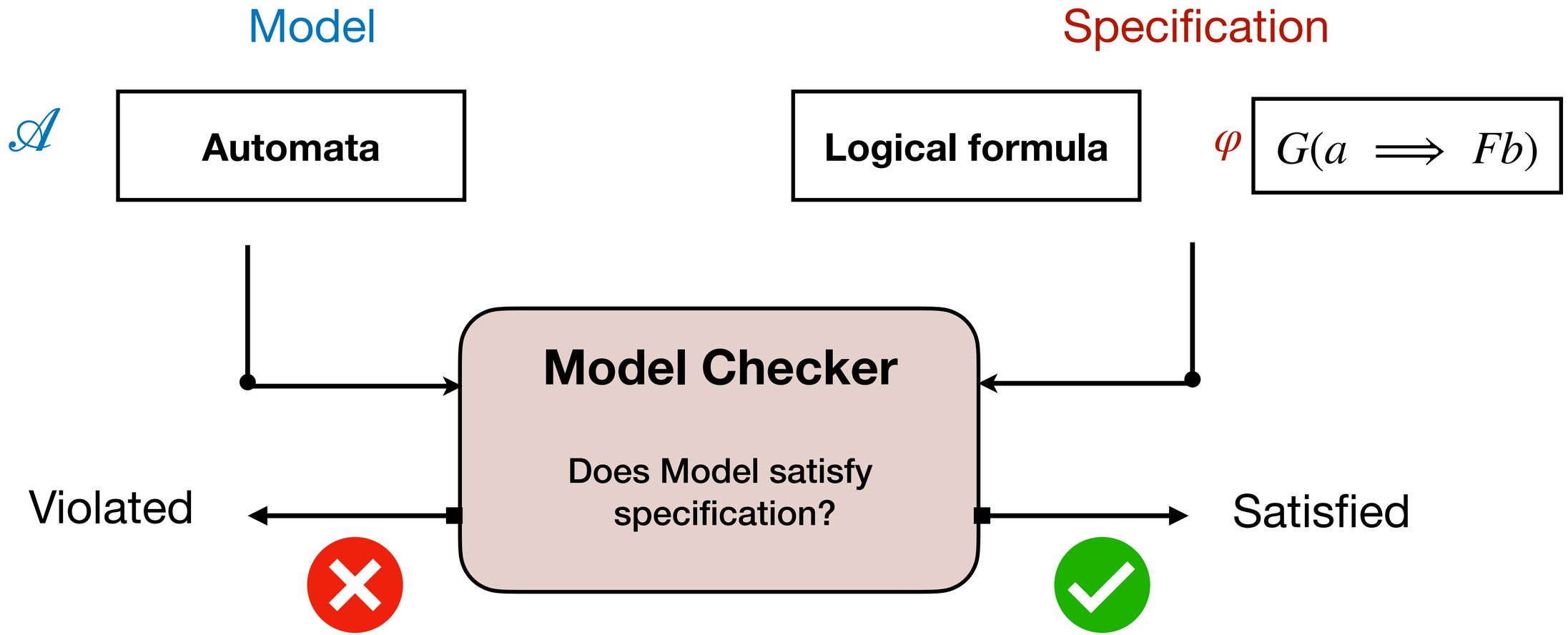


Specification

φ $G(a \implies Fb)$

Every request is followed by a response

Model Checking



Model Checking

Model

$\mathcal{A}$ **Automata**

Specification

Logical formula φ $G(a \implies Fb)$

Model Checker

Logic to Automata Translation

$\varphi \longmapsto \neg\varphi \longmapsto \mathcal{A}_{\neg\varphi}$

Reachability/Liveness in Network of Automata

$\mathcal{A} \times \mathcal{A}_{\neg\varphi}$

$L(\mathcal{A}) \subseteq L(\varphi)$?

↓

Is $L(\mathcal{A}) \cap \overline{L(\varphi)}$ empty?

Our Focus

Model Checking
for Real-time systems

Model Checking for Real-time systems

Does **System S** satisfy **Property P** ?

modelled using

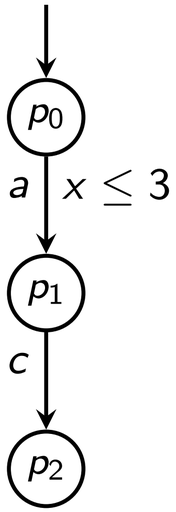
Automata

Timed Automata

Automata with Timers

Event-clock Automata

Several choices!



A_1

specified using

Logical formula

Metric Interval Temporal Logic

Specification

φ $G(a \implies F_{<3} b)$

Every request is followed by a response ***within 3 time units***

MITL Model Checking

For Continuous semantics

[Ferrere Maler Nickovic Pnueli ’19]

MITL formulae → Formulae with only one-sided intervals

Does not work for pointwise semantics!

✅ Easier to construct timed automata for these simpler formulae

For Pointwise semantics

MightyL
[Brihaye Geeraerts Ho Monmege ’17]

MITL formulae → One-clock alternating timed automata → Network of timed automata

Proceeds through

Our goal: A new procedure for efficient MITL model-checking?

Model Checking for Real-time systems

Model

$\mathcal{A}$ **Network of Timed Automata**

Specification

MITL formula φ $G(a \implies F_{<3} b)$

Logic to Automata Translation

$\varphi \longmapsto \neg\varphi \longmapsto \mathcal{A}_{\neg\phi}$

Reachability/Liveness in Network of Automata

$\mathcal{A} \times \mathcal{A}_{\neg\phi}$

Efficient Translation from MITL to automata

Efficient Algorithms for Reachability/Liveness

To which automata?

Generalised Timed Automata (GTA)

Models

Allows to use the features of these models simultaneously

- Timed Automata
- Automata with Timers
- Event-clock Automata

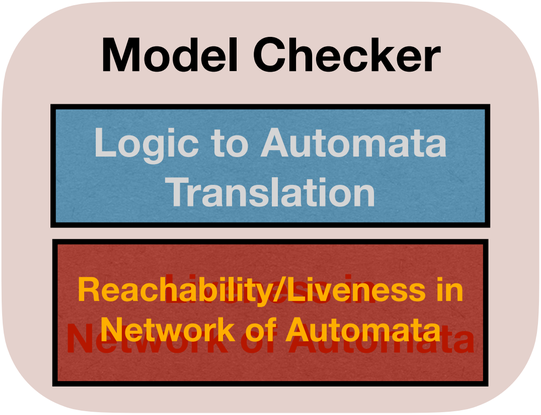
Specifications

Direct translation from MITL to GTA

Efficient techniques comparable to state-of-the-art techniques for Timed Automata

- Zones
- Zone Graphs
- Finite abstractions for Zone graphs

Our Pipeline



A translation from MITL formulae to GTA

A model that succinctly captures several widely used timed formalisms

Exponentially more succinct than the current logic-automata translation

Procedure to check Reachability/Liveness for GTA

Zone-based with similar complexity as Timed Automata procedures

A new procedure for MITL model-checking

Formal Verification of Real-Time Systems

Overview

Model Checking for Real-time systems

Motivations for our work

Generalised Timed Automata (GTA)

Metric Interval Temporal Logic (MITL)

Logic to Automata translation

MITL to GTA

Uses powerful features of GTA

Exponentially more succinct than the state-of-the-art

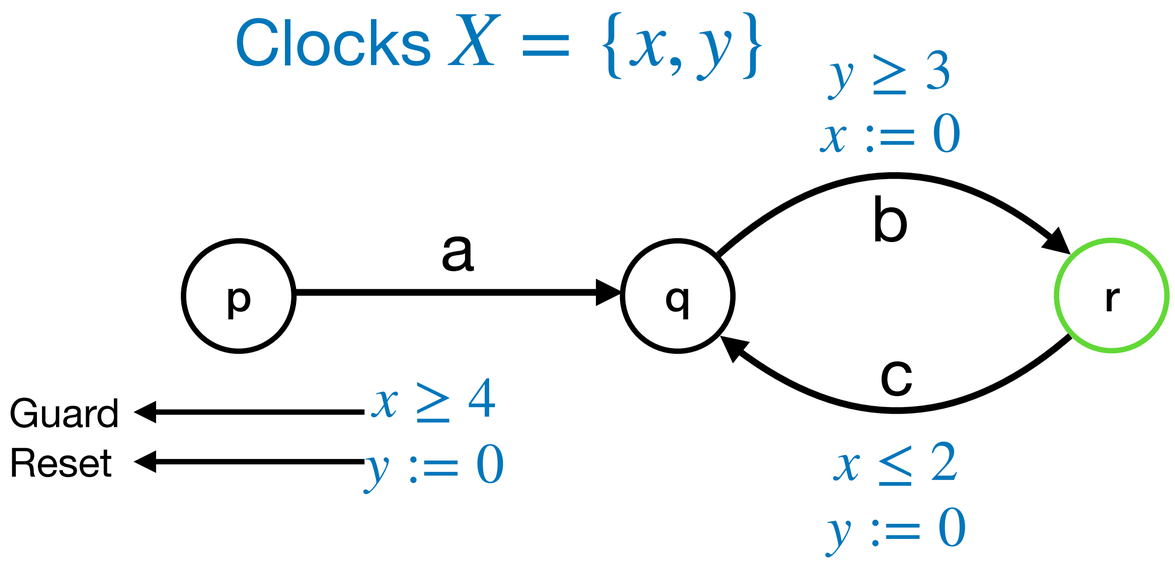
Improvements to Translation

Towards more Determinism

No time-abstract bisimulation for GTAs

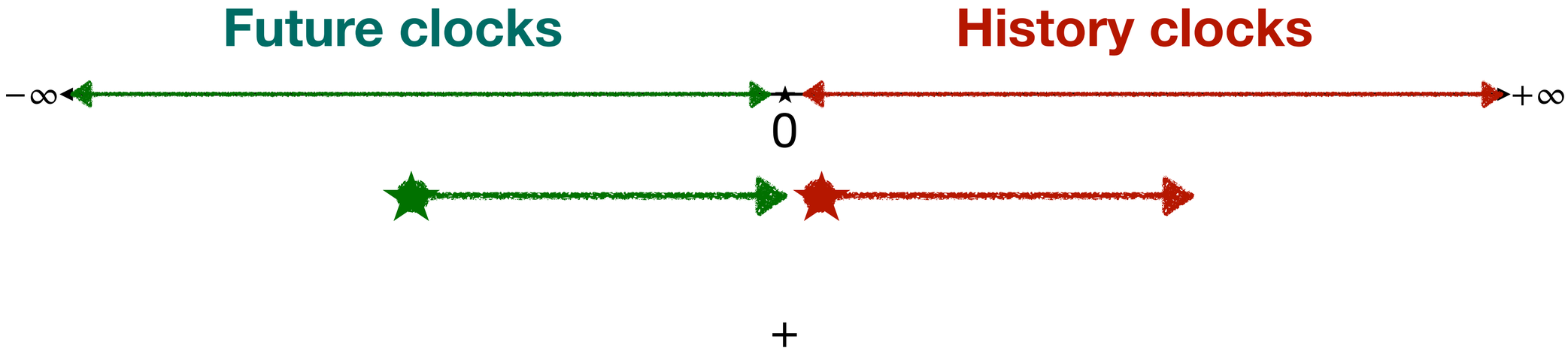
A new zone-based algorithm for checking Büchi non-emptiness in GTAs

Timed Automata [Alur Dill ’89]



Generalized Timed Automata (GTA) [AGGJS ’23]

We enrich the timing constructs



Richer syntax on transitions

Generalising Clocks, Event clocks, Timers [AGGJS ’23]

Future clocks

Guards: $-5 \leq y \leq -2$

Release: sets y to an arbitrary value in $[-\infty, 0]$

$$< [y]; -5 \leq y \leq -2 >$$

Generalizes timers and
predicting clocks (for specifications)

History clocks

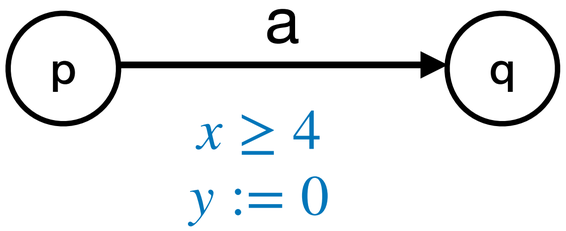
Guards: $2 \leq x \leq 5$

Reset: sets x to 0.

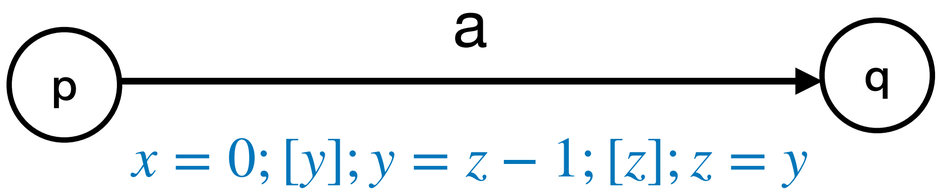
$$< 2 \leq x \leq 5; [x] >$$

Generalizes standard clocks and
recording clocks (for specifications)

Generalising Transitions



Guard followed by Reset

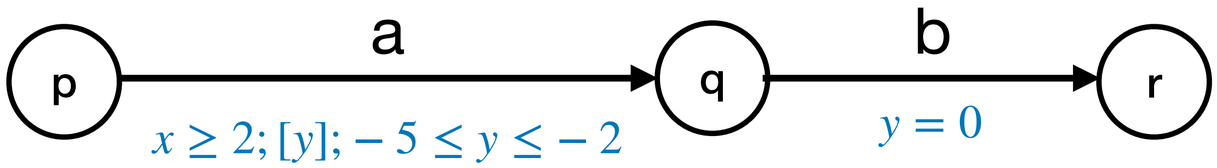


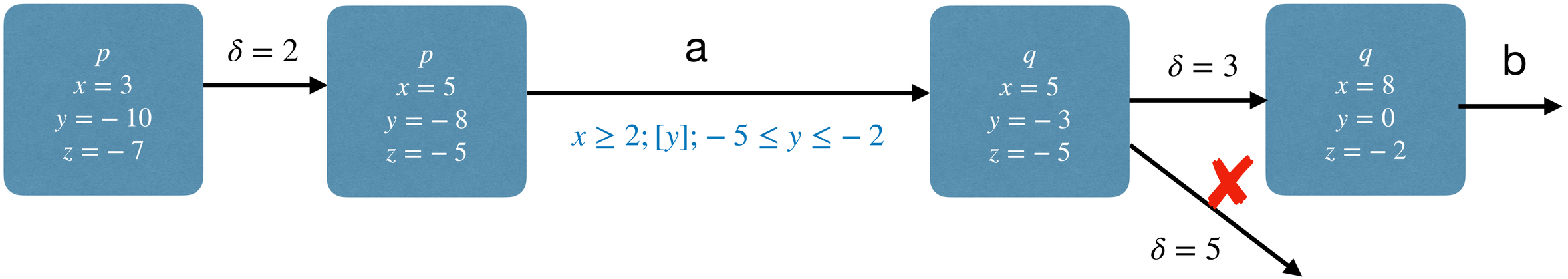
Instantaneous Timed Programs

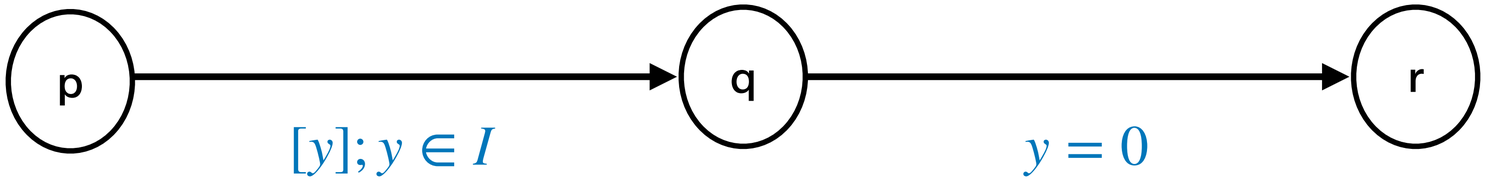
Arbitrary interleaving of guards and clock-transformations

Allows succinct modelling!

A run of GTA







Set y to a value in I

Validate the guess

Formal Verification of Real-Time Systems

Overview

Model Checking for Real-time systems

Motivations for our work

Generalised Timed Automata (GTA)

Metric Interval Temporal Logic (MITL)

Logic to Automata translation

MITL to GTA

Uses powerful features of GTA

Exponentially more succinct than the state-of-the-art

Improvements to Translation

Towards more Determinism

No time-abstract bisimulation for GTAs

A new zone-based algorithm for checking Büchi non-emptiness in GTAs

Linear Temporal Logic [Pnueli '77]

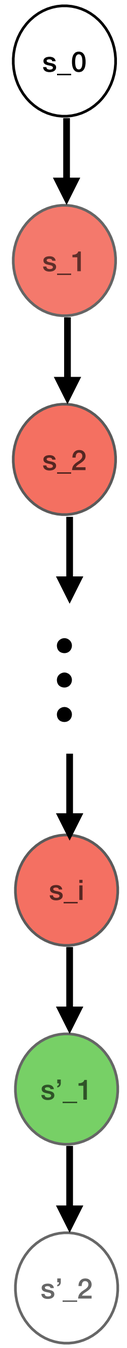
φ := $p \in \text{Prop}$

$\varphi_1 \wedge \varphi_2$ $\varphi_1 \vee \varphi_2$ $\neg\varphi$

$X\ \varphi$ φ holds in the next position

$F\ \varphi$ φ holds eventually

$\varphi_1\ U\ \varphi_2$ φ_1 holds until φ_2 holds



Metric Interval Temporal Logic

[Alur Feder Henzinger '96]

φ := $p \in \text{Prop}$

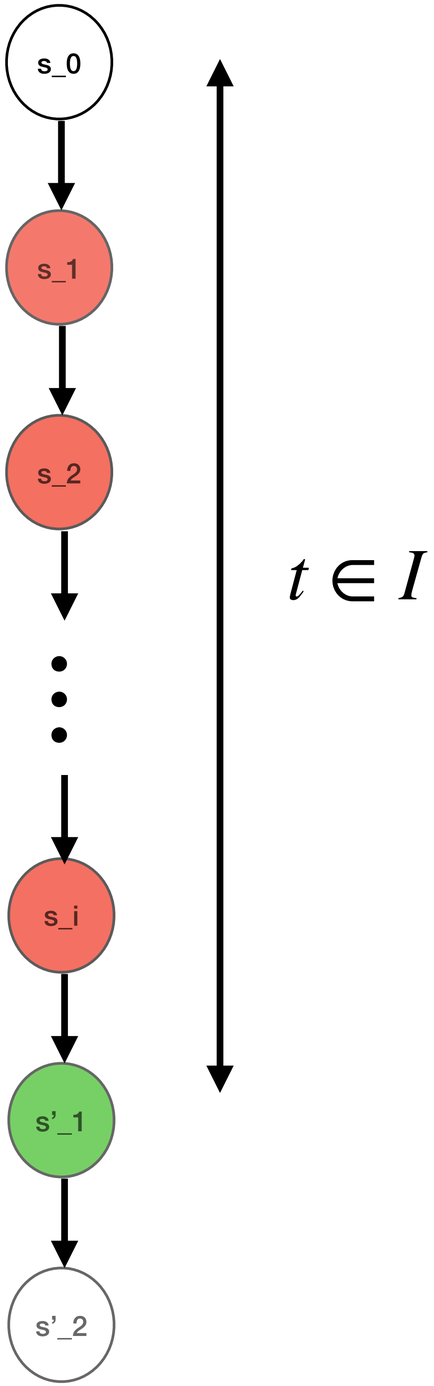
$\varphi_1 \wedge \varphi_2$ $\quad$ $\varphi_1 \vee \varphi_2$ $\quad$ $\neg\varphi$

$X_I\ \varphi$ $\quad$ φ holds in the next position

$F_I\ \varphi$ $\quad$ φ holds eventually

$\varphi_1\ U_I\ \varphi_2$ $\quad$ φ_1 holds until φ_2 holds

No punctual constraints: I is not singleton



Formal Verification of Real-Time Systems

Overview

Model Checking for Real-time systems

Motivations for our work

Generalised Timed Automata (GTA)

Metric Interval Temporal Logic (MITL)

Logic to Automata translation

MITL to GTA

Uses powerful features of GTA

Exponentially more succinct than the state-of-the-art

Improvements to Translation

Towards more Determinism

No time-abstract bisimulation for GTAs

A new zone-based algorithm for checking Büchi non-emptiness in GTAs

"Prediction is difficult, especially when dealing with the future"

Next

MITL to GTA translation

Extending

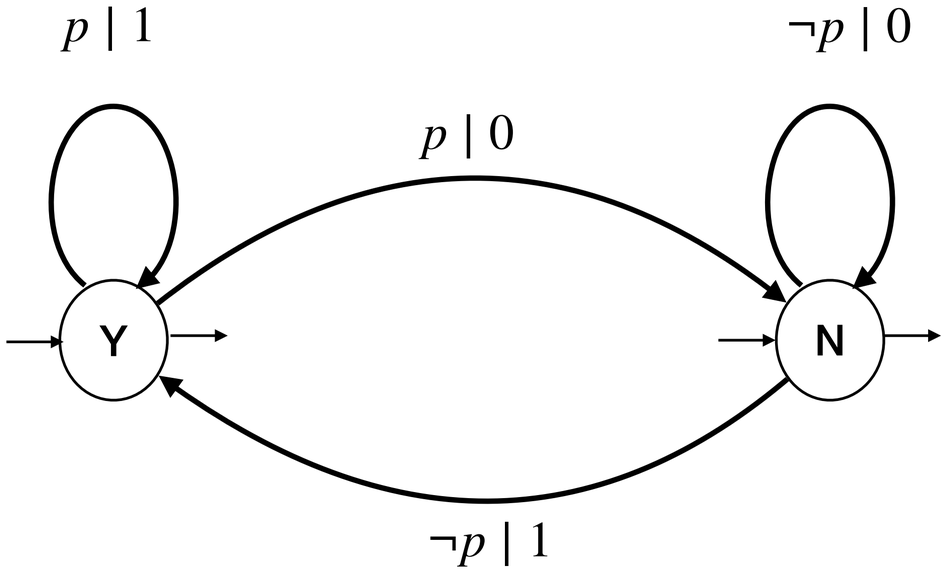
LTL to Automata translation

using

the powerful features of GTA

Warmup: LTL to NFA

Xp - Next position is labelled by p



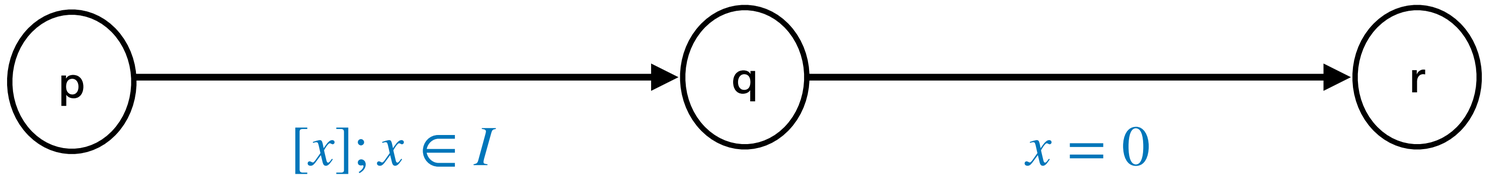
$$p \ \neg p \ p \ p \ p \ \neg p \ p \ \neg p \ \cdots$$
$$0 \quad 1 \ 1 \ 1 \ 0 \quad 1 \ 0 \quad \cdots$$

Y - next event is a p-event

N - next event is not a p-event

MITL to GTA translation [Akshay Gastin G. Srivathsan '24]

Central idea: Predictions using Future Clocks



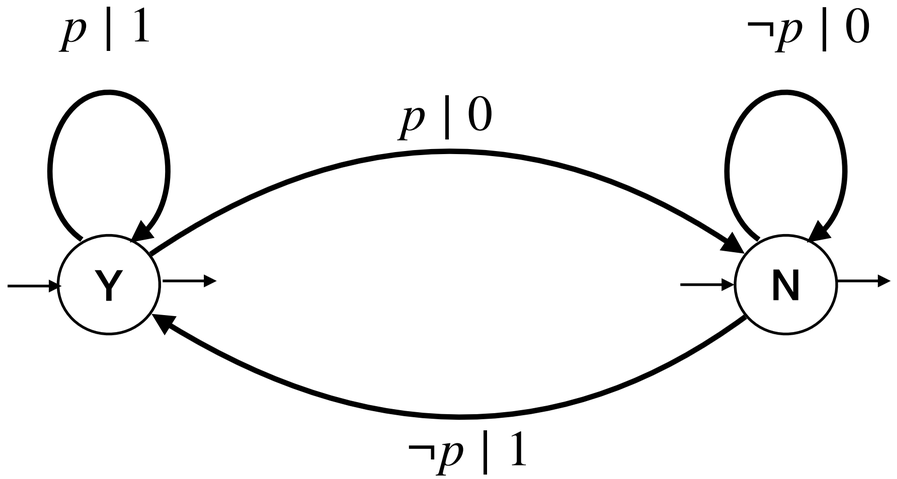
Set x to a value in I

Validate the guess

LTL to NFA

Xp - Next position is labelled by p

$$p \ \neg p \ p \ p \ p \ \neg p \ p \ \neg p \ \cdots$$
$$0 \quad 1 \ 1 \ 1 \ 0 \quad 1 \ 0 \quad \cdots$$

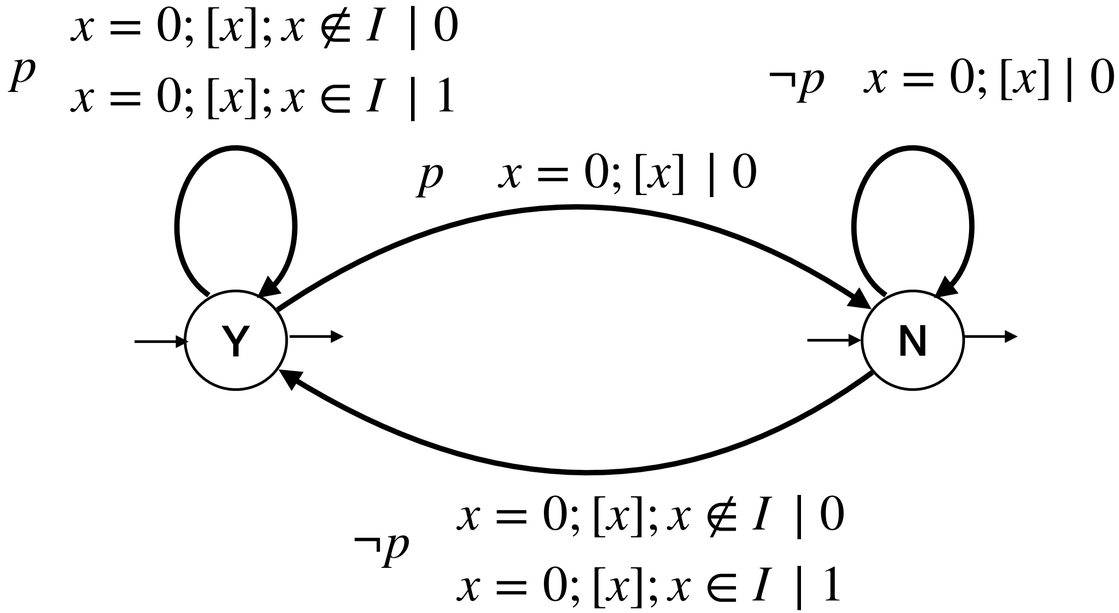


Y - next position is labelled by p

N - next event is not labelled by p

MITL to GTA

$X_{[3,5]}p$ - Next position is labelled by p and is within interval [3,5]



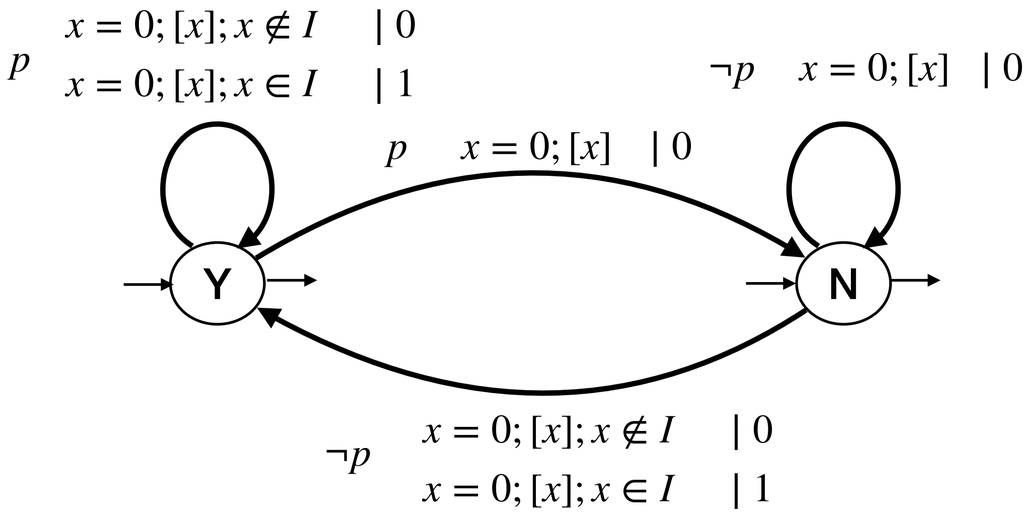
Future clock x - predicts the time to next event

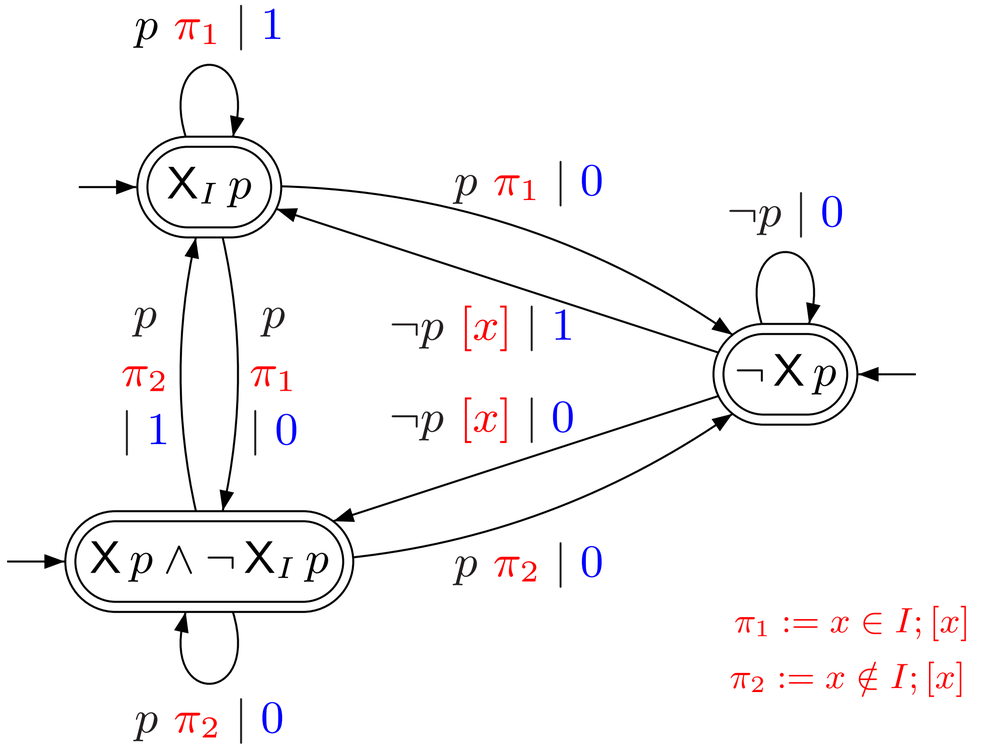
When next event is p-event, check if $x \in I$

MITL to GTA

Translation to TA?

$X_{[3,5]}p$ - Next position is labelled by p
and is within interval [3,5]





Predictions are handled using future clocks

Predictions are stored in states

MITL to GTA translation

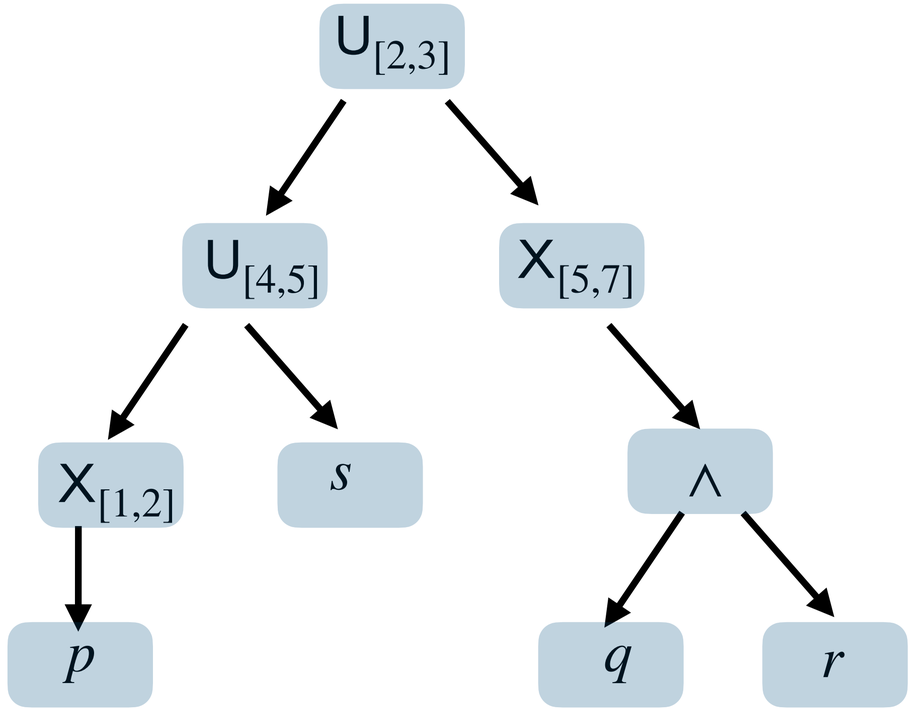
Produces a network of GTAs

Each sub-formula → GTA (with outputs)

Each component

Reads the outputs produced by children

Feeds its output to its parents



$(X_{[1,2]}\ p\ U_{[4,5]}\ s)\ U_{[2,3]}\ (X_{[5,7]}\ (q \wedge r))$

Formal Verification of Real-Time Systems

Overview

Model Checking for Real-time systems

Motivations for our work

Generalised Timed Automata (GTA)

Metric Interval Temporal Logic (MITL)

Logic to Automata translation

MITL to GTA

Uses powerful features of GTA

Exponentially more succinct than the state-of-the-art

Improvements to Translation

Towards more Determinism

No time-abstract bisimulation for GTAs

A new zone-based algorithm for checking Büchi non-emptiness in GTAs

What next?

Extensions to allow more modalities

Improvements to our translation

Determinism where possible

Controlled non-determinism otherwise

Sharing of information between components

TEMPORA **tool implementing the pipeline**

Extending our Translation

MITL^{+p}

Past operators Not considered in earlier translations

$Y_I\ \varphi$ φ holds in the previous position

$\varphi_1\ S_I \varphi_2$ φ_1 holds since φ_2 holds

Punctual intervals at outermost level

$(\mathsf{X}_{[1,2]}\ p\ \mathsf{U}_{[4,5]}\ s)\ \mathsf{U}_{[2,2]}\ (\mathsf{X}_{[5,7]}\ (q \wedge r))$

Improvements to Translation

A Deterministic Fragment detMITL^{+p}

Outer operator: Any — Future/past (even punctual)

Inner operators: past only

Obtained network is deterministic

Linear-time translation

Beyond Determinism — Full MITL^{+p}

Limit non-determinism

Improvements to Translation

MITL^{+p}

Optimise size of resultant GTA network

Sharing of information

- **Clocks**
- **Components/Automata**
- **Predictions**

→

Reduced non-deterministic branching

More compact networks

MITL to GTA translation

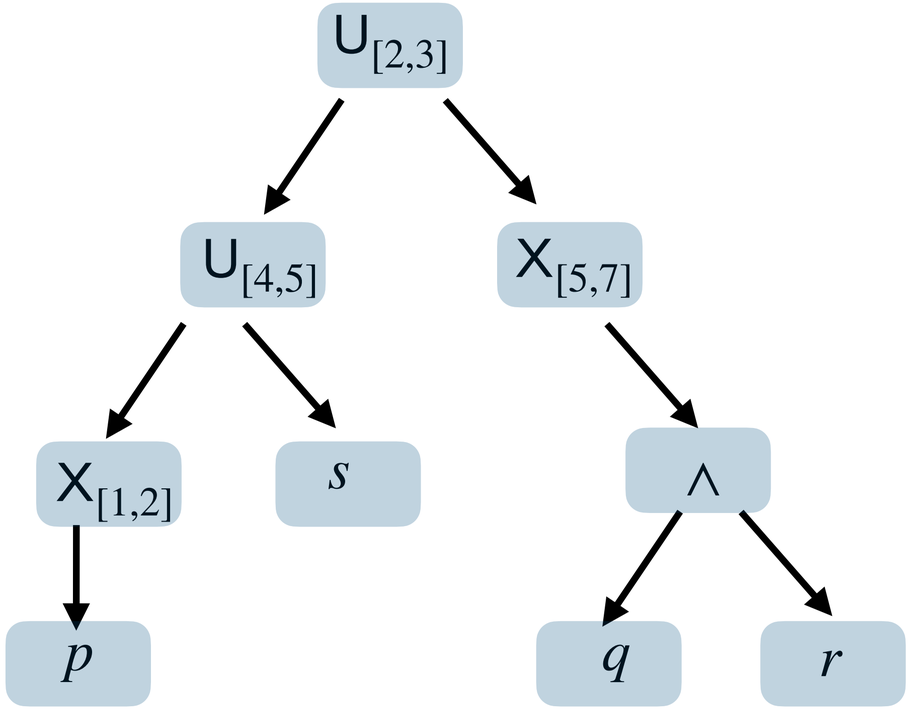
Produces a network of GTAs

Each sub-formula → GTA (with outputs)

Each component

Reads the outputs produced by children

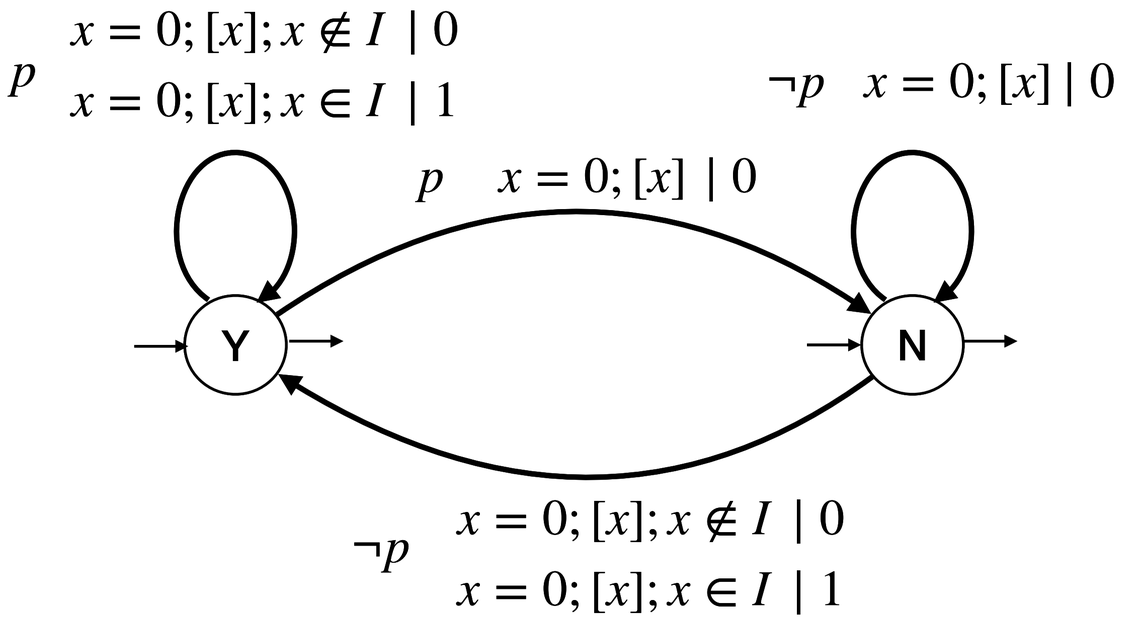
Feeds its output to its parents



$(X_{[1,2]}\ p\ U_{[4,5]}\ s)\ U_{[2,3]}\ (X_{[5,7]}\ (q \wedge r))$

MITL to GTA translation

$X_{[3,5]}p$ - Next position is labelled by p and is within interval [3,5]



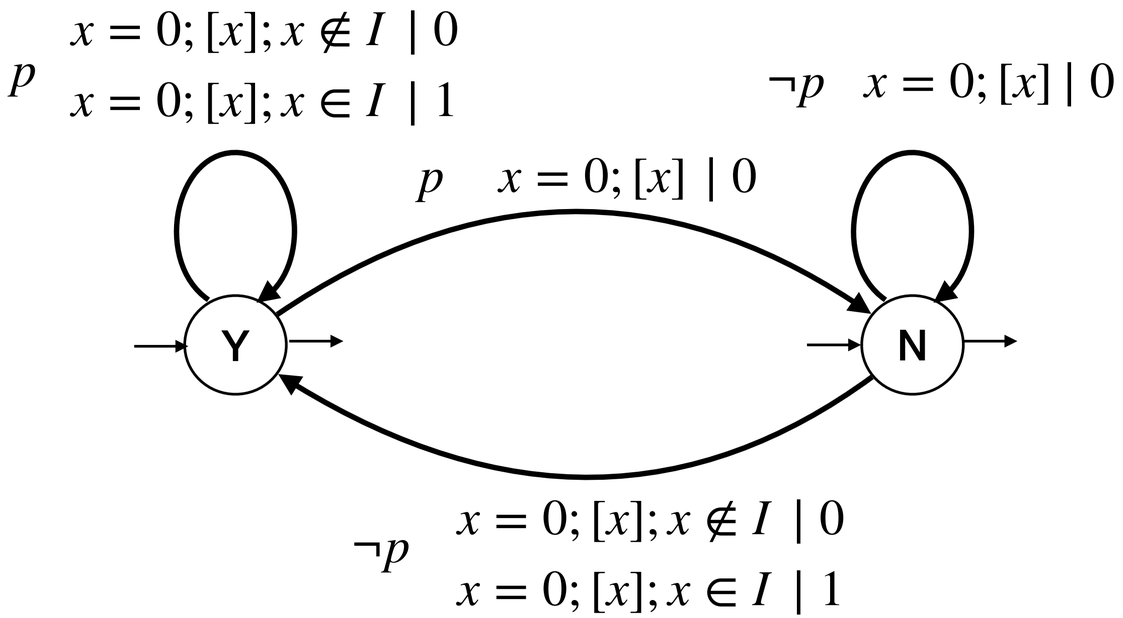
Future clock x - predicts the time to next event

When next event is p-event, check if $x \in I$

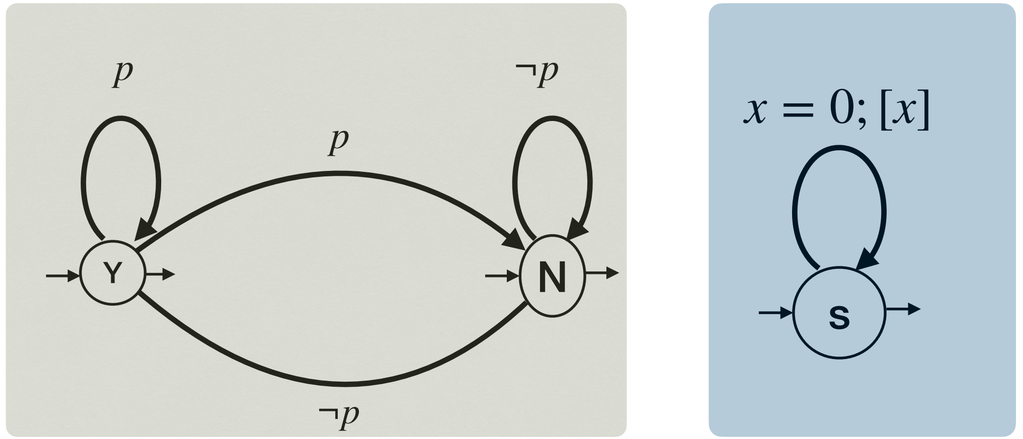
MITL to GTA translation

Original

$X_{[3,5]}p$ - Next position is labelled by p and is within interval [3,5]



Optimised



Untimed part $\mathcal{A}_X$ $\mathcal{A}_{Next}$ Timed part

$\text{out}_X := \mathcal{A}_X.state = Y \wedge -\mathcal{A}_{Next}.x \in I$

Future clock x - predicts the time to next event

When next event is p-event, check if $x \in I$

Allows to use only one clock for all Next operators

Experimental Evaluation

TEMPORA **a tool implementing the pipeline**

Built on top of TChecker

Optimised MITL-to-GTA **translation**

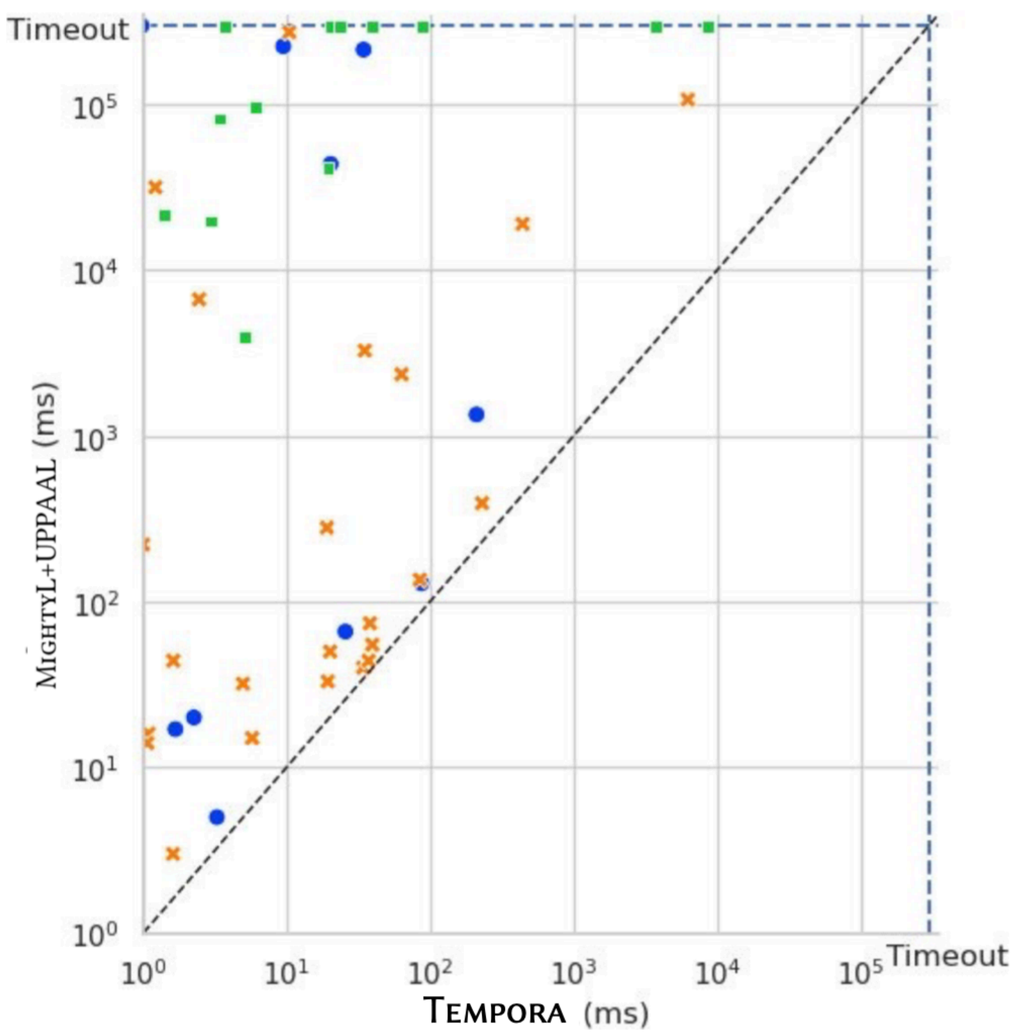
+

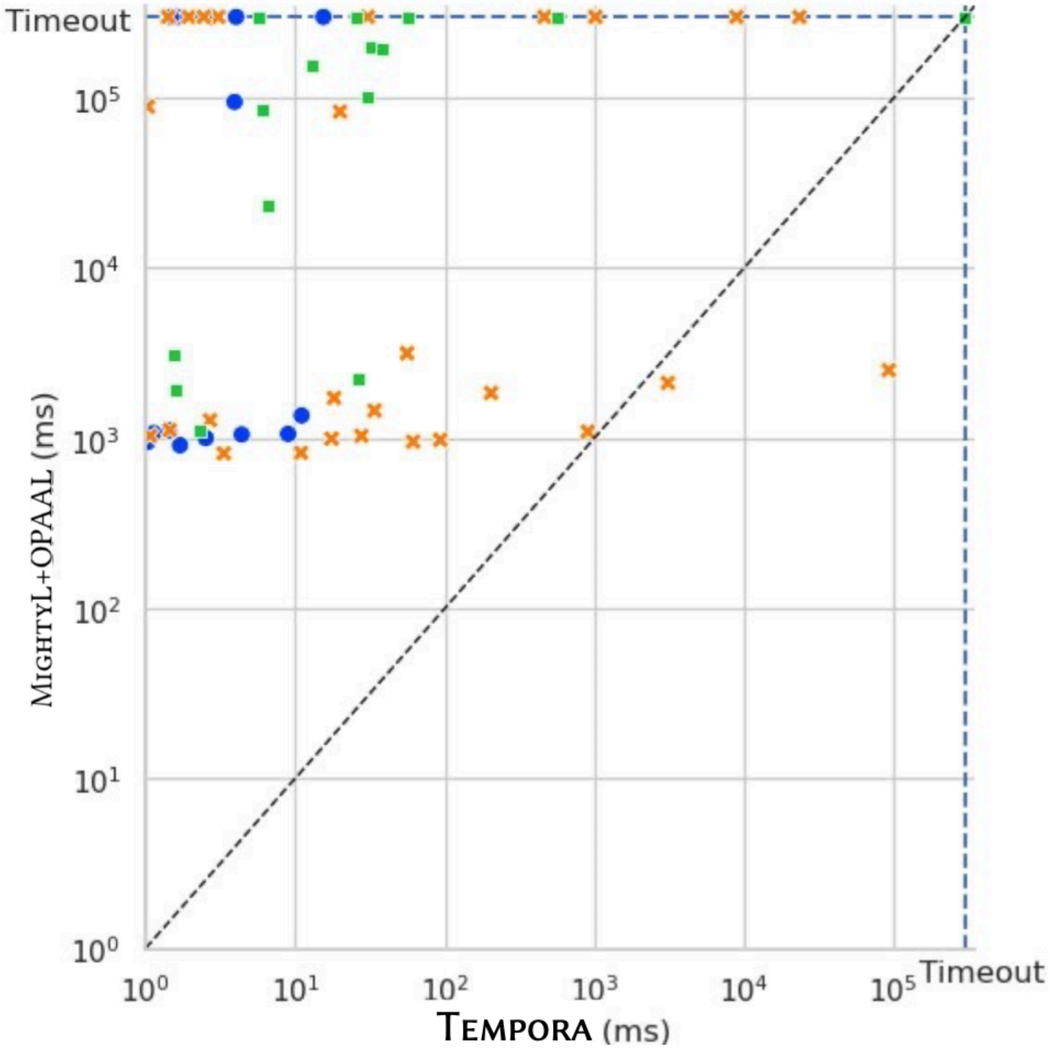
GTA reachability/liveness

Comparison with MightyL + UPPAAL for finite words

MightyL + OPAAL for infinite words

Experimental Evaluation





Formal Verification of Real-Time Systems

Overview

Model Checking for Real-time systems

Motivations for our work

Generalised Timed Automata (GTA)

Metric Interval Temporal Logic (MITL)

Logic to Automata translation

MITL to GTA

Uses powerful features of GTA

Exponentially more succinct than the state-of-the-art

Improvements to Translation

Towards more Determinism

Extending our translation - MITL^{+p}

Maximising determinism

Faster translation for subclass detMITL^{+p}

MITL Model-checking using GTA

A new model that unifies the features of several timed formalisms
Timed Automata, Event Clock Automata, Timers

+

Efficient Procedure to check reachability/liveness for GTAs
Decides reachability/liveness for various models

+

Translation from MITL formulae to GTA
A direct and succinct translation

A new procedure for MITL model-checking

TEMPORA
GTA-based tool