ICFEM 2024

日本，広島市

Wednesday 4th December 2024

Tuning Trains Speed in Railway Scheduling

(short paper)

Étienne André

Université Sorbonne Paris Nord
CNRS, LIPN, F-93430 Villetaneuse, France

CC BY SA

Context: Verifying critical timed systems

- Critical systems: Failures may result in dramatic consequences
- Need for early bug detection
  - Bugs discovered when final testing: expensive

  ⇝ Need for a thorough specification and verification phase

Zoufftgen, 2006
6 fatalities

温州, 2011
40 fatalities

花蓮縣, 2021
50 fatalities

- Formal verification is needed to ensure the absence of bugs

Context: trains scheduling

Problems

- Complex networks with railroad switches
- Different train speeds
- Timed uncertainty
  - Maximal speeds may not be known in advance (depend on segments and on trains)
  - Actual speeds can vary at runtime

Need for formal guarantees to assess the correctness of complex systems

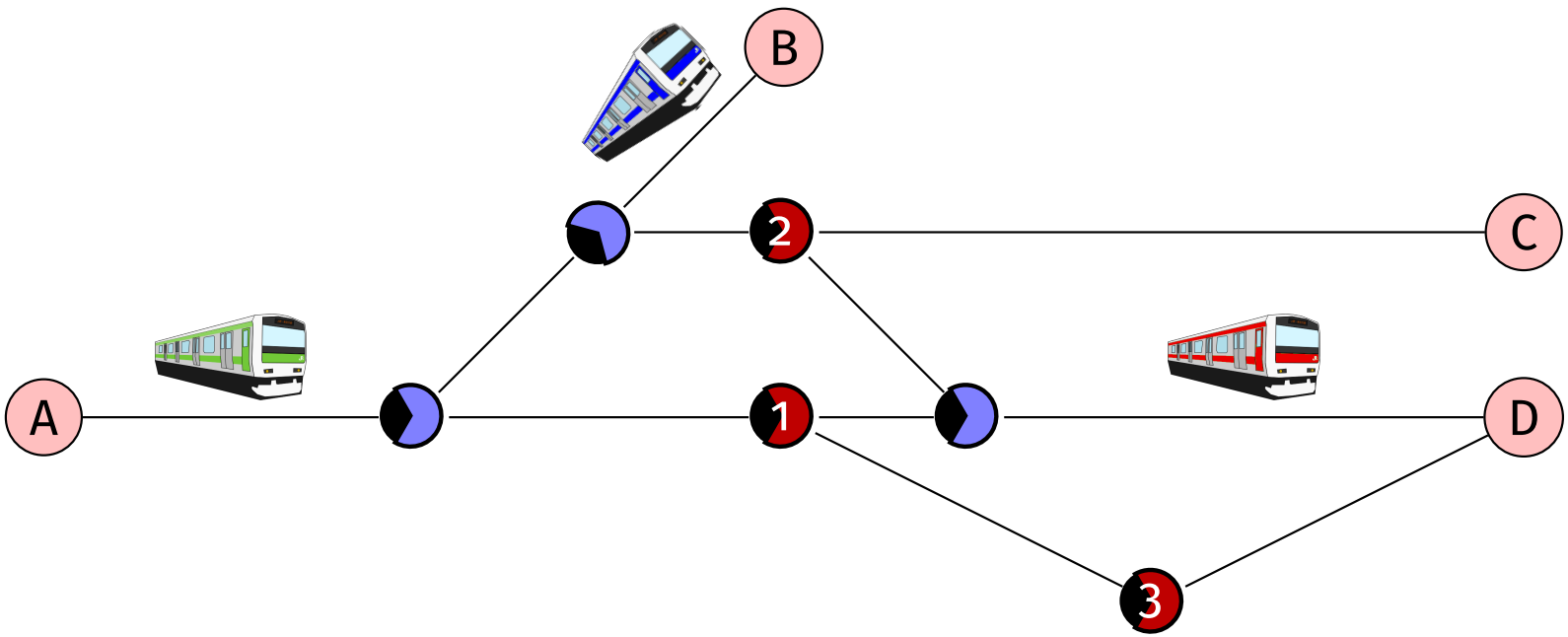
Conflicting constraints

- Safety: absence of collisions
- Optimality
  - Fastest connections
  - Lowest energy consumption (not considered here)

Overall goal

Inputs:

1. a rail network graph
2. a list of trains, each having an origin, a target and intermediate constraints
3. a list of schedule constraints [KR23]



Goal: exhibit a low-level schedule within the railway network such that all trains to meet their criteria

[KR23] Tomáš Kolárik and Stefan Ratschan. "Railway Scheduling Using Boolean Satisfiability Modulo Simulations". In: FM. vol. 14000. Lecture Notes in Computer Science. Springer, 2023, pp. 56–73

Our contributions in a nutshell

1. a formalization of the system model and of the problem, including a support for timed uncertainty
2. a translation into (parametric) timed automata [AHV93]
3. a set of experiments using IMITATOR

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: STOC. ACM, 1993, pp. 592–601

Outline

1 System model

2 Problem

3 Translation

4 Experiments

5 Perspectives

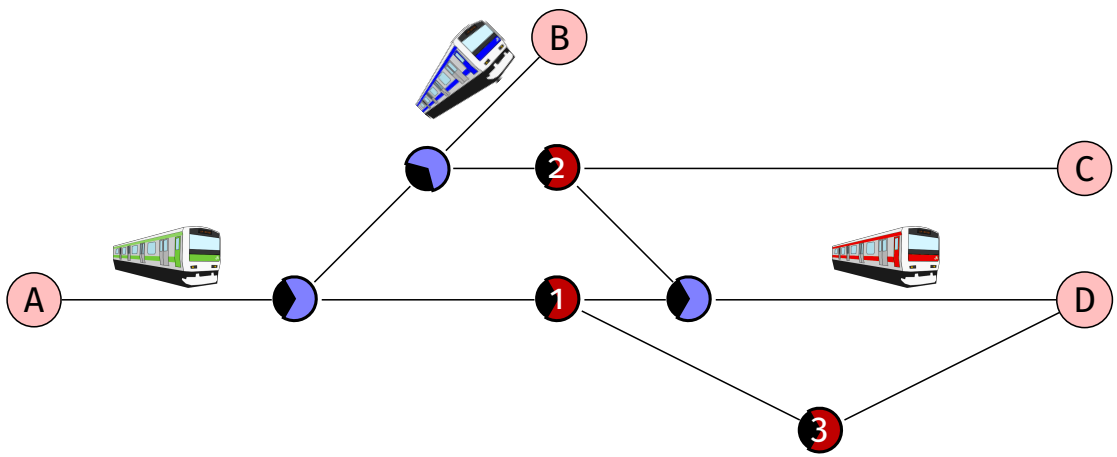
Outline

1 System model
- Rail network graph
- Trains
- Schedule constraints

Rail network graph [KR23]

- double-vertex graph, with nodes and segments
- nodes can be normal (not allowing stopping), stations (where trains may or not stop) or boundary (start/end nodes for trains)
- segments are bidirectional, have a length and a maximal speed (encoded using traversal times)

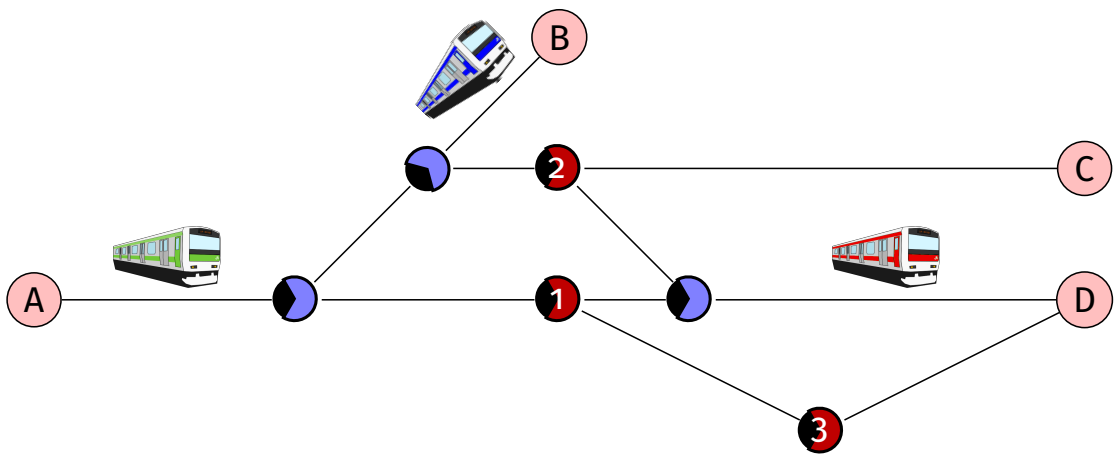
- switches are directional (prevents impossible turns)
- cycles are allowed



[KR23] Tomáš Kolárik and Stefan Ratschan. "Railway Scheduling Using Boolean Satisfiability Modulo Simulations". In: FM. vol. 14000. Lecture Notes in Computer Science. Springer, 2023, pp. 56–73

Rail network graph [KR23]

- double-vertex graph, with nodes and segments
- nodes can be normal (not allowing stopping), stations (where trains may or not stop) or boundary (start/end nodes for trains)
- segments are bidirectional, have a length and a maximal speed (encoded using traversal times)
  - possibly uncertain or unknown timing constants (set $\mathbb{P}$ of parameters)
- switches are directional (prevents impossible turns)
- cycles are allowed



[KR23] Tomáš Kolárik and Stefan Ratschan. "Railway Scheduling Using Boolean Satisfiability Modulo Simulations". In: FM. vol. 14000. Lecture Notes in Computer Science. Springer, 2023, pp. 56–73

Outline

1 System model
- Rail network graph
- Trains
- Schedule constraints

Train

- A train is characterized by
  - its velocity limit (per segment)
  - its connection
- trains always drive at their maximum possible speed; they can however stop arbitrarily long in stations
- not encoded: train length, weight, acceleration and deceleration

Connection constraints

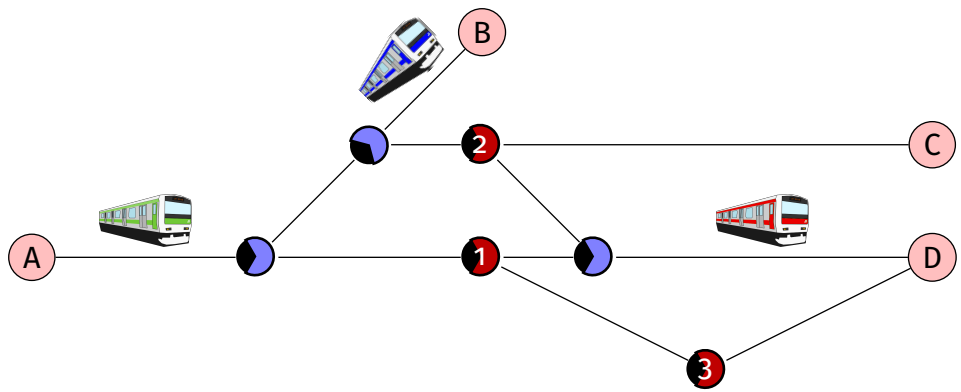
Each train has exactly one connection:

1. 1 starting node
2. a list of nodes that must be visited in the given order
3. 0 or 1 target node

Connection constraints

Each train has exactly one connection:

1. 1 starting node
2. a list of nodes that must be visited in the given order
3. 0 or 1 target node



Example: green train has connection $[A, 3]$

- cannot stop at station 1, may stop at station 3
- can end in any boundary node (here only D due to the topology)

Outline

1 System model
- Rail network graph
- Trains
- Schedule constraints

Schedule constraints

Notations:

- $arrival(t, n)$: denotes the time when train t arrives at node n
- $departure(t, n)$: denotes the time when train t leaves node n
- generic notation $visit(t, n)$: denotes arrival or departure

Three types of schedule constraints

1. Ordering constraints
2. Absolute timing constraints
3. Relative timing constraints

Ordering constraints

Ordering constraints constrain the order in which visits should be made

$$visit_1(t_1, n_1) \bowtie visit_2(t_2, n_2)$$

Example: $departure(t_{blue}, A) \leq departure(t_{green}, A)$

- "the blue train must start from node A before the green train"

Absolute timing constraints

Absolute timing constraints constrain the visit of a node at an absolute time

- can use timing parameters

$$visit(t, n) \bowtie d, \text{ with } d \in \mathbb{Q}_{\geq 0} \cup \mathbb{P}$$

Example: $visit(t_{blue}, 2) \geq J$

- "the blue train must visite station 2 after date J"

Relative timing constraints

Let $transfer\big(visit_1(t_1, n_1), visit_2(t_2, n_2)\big) :- visit_2(t_2, n_2) - visit_1(t_1, n_1)$.

Relative timing constraints constrain the time between two visits

- can use timing parameters

$$transfer\big(visit_1(t_1, n_1), visit_2(t_2, n_2)\big) \bowtie d, \text{ with } d \in \mathbb{Q}_{\geq 0} \cup \mathbb{P}$$

Example: $transfer\big(departure(t_{red}, D), arrival(t_{red}, A)\big) \leq 10$

- "The red train must reach A within 10 time units after leaving D"

Outline

1 System model

2 Problem

3 Translation

4 Experiments

5 Perspectives

Problem

Train trajectory problem under uncertain speeds

Input: a constrained railway system over a set of timing parameters

Goal: Synthesize segment durations and schedule constraints parameters such that a low-level schedule exists, i. e., all train connections and schedule constraints can be met.

Outline

1 System model

2 Problem

3 Translation

4 Experiments

5 Perspectives

Overview of the proposed method

1. modular translation to parametric timed automata [AHV93]
2. verify (and exhibit) schedules
   - using timed model checking
3. or synthesize parameter valuation for which schedules can be exhibited
   - using parametric timed model checking

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: STOC. ACM, 1993, pp. 592–601

Outline

3 Translation
- (Parametric) timed model checking
- Modular translation

Model checking in a nutshell

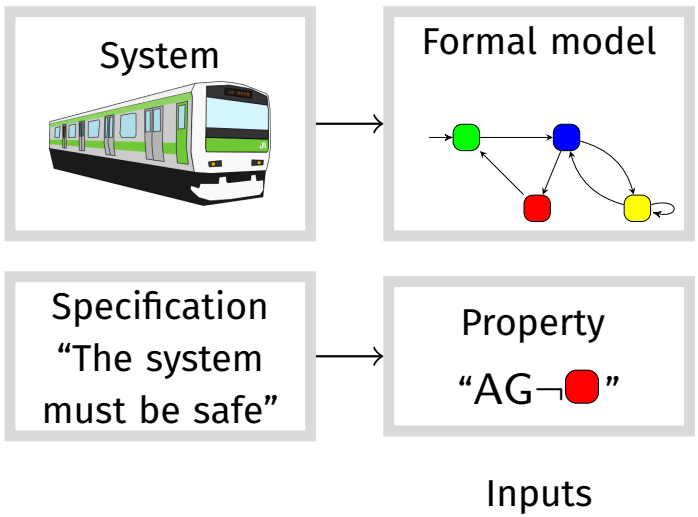
- Principle of model checking

System

Specification
“The system must be safe”

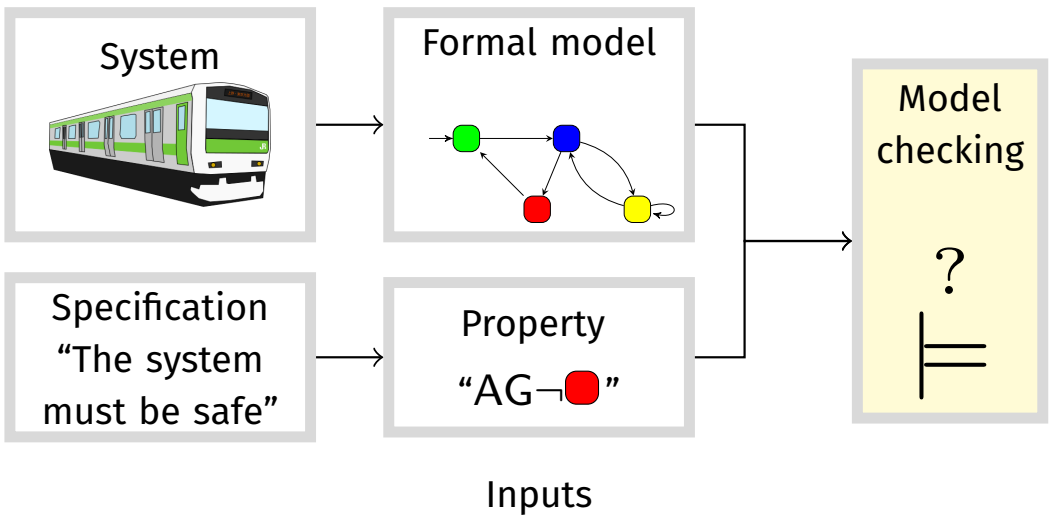
Model checking in a nutshell

- Principle of model checking



Model checking in a nutshell

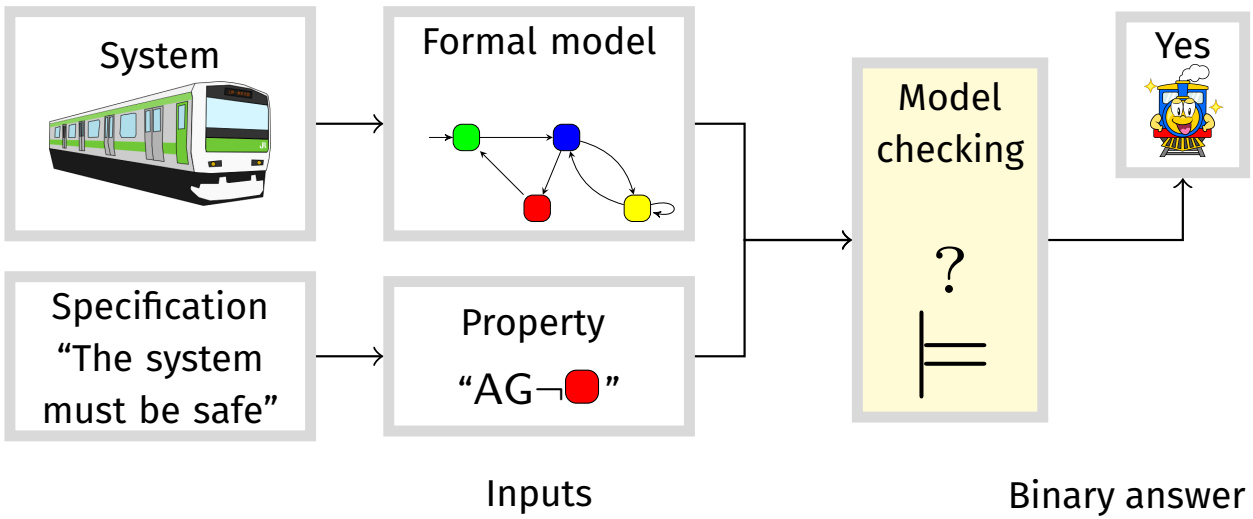
- Principle of model checking



- Question: does the model of the system satisfy the property?

Model checking in a nutshell

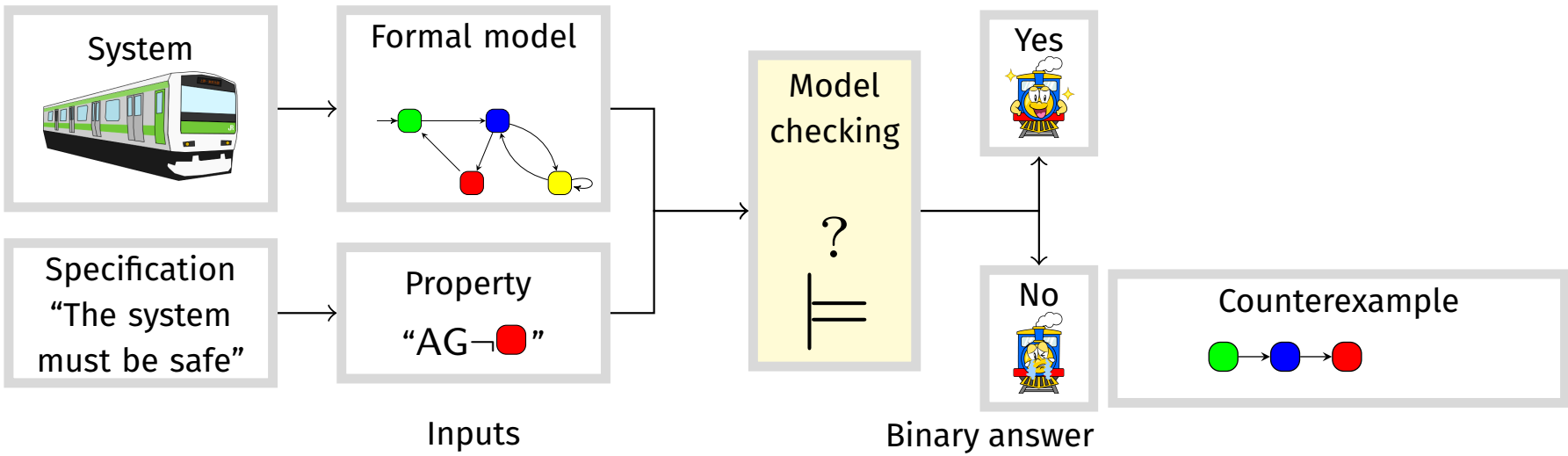
- Principle of model checking



- Question: does the model of the system satisfy the property?

Model checking in a nutshell

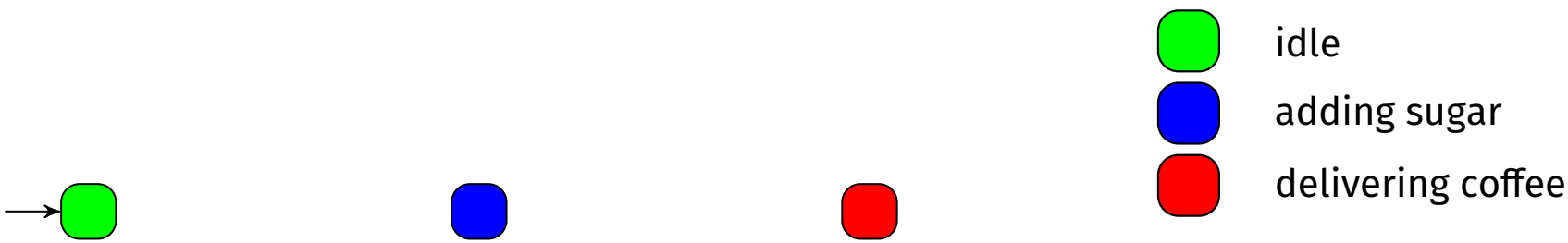
- Principle of model checking



- Question: does the model of the system satisfy the property?

Timed automaton (TA)

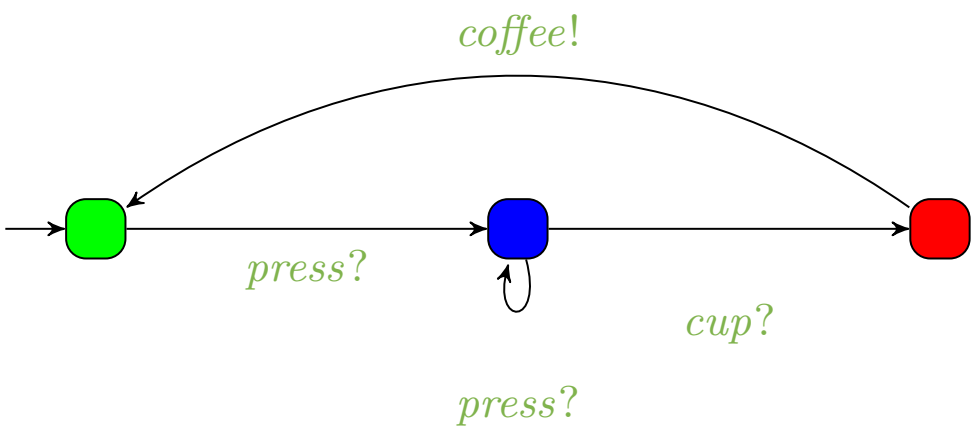
- Finite-state automaton (sets of locations)

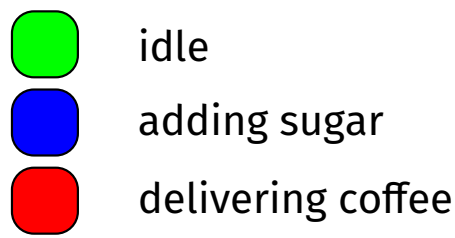


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: Theoretical Computer Science 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions)

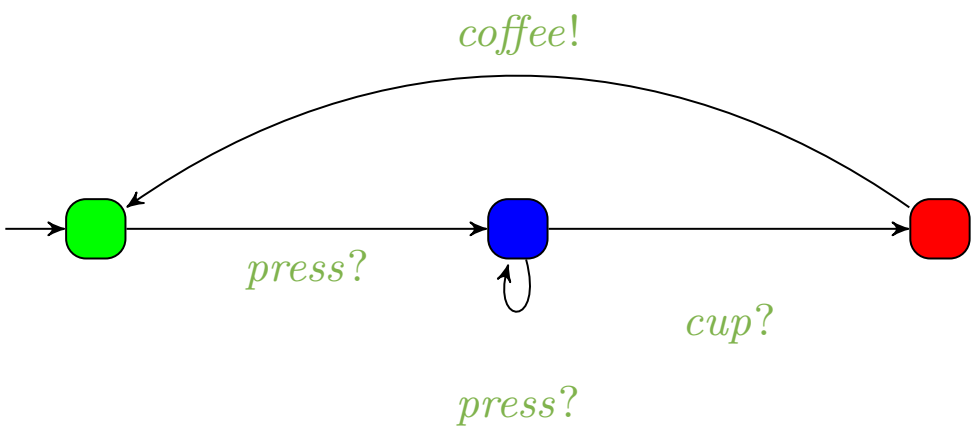


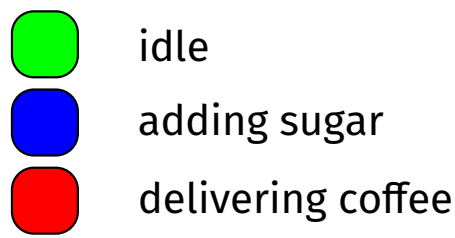


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: Theoretical Computer Science 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate



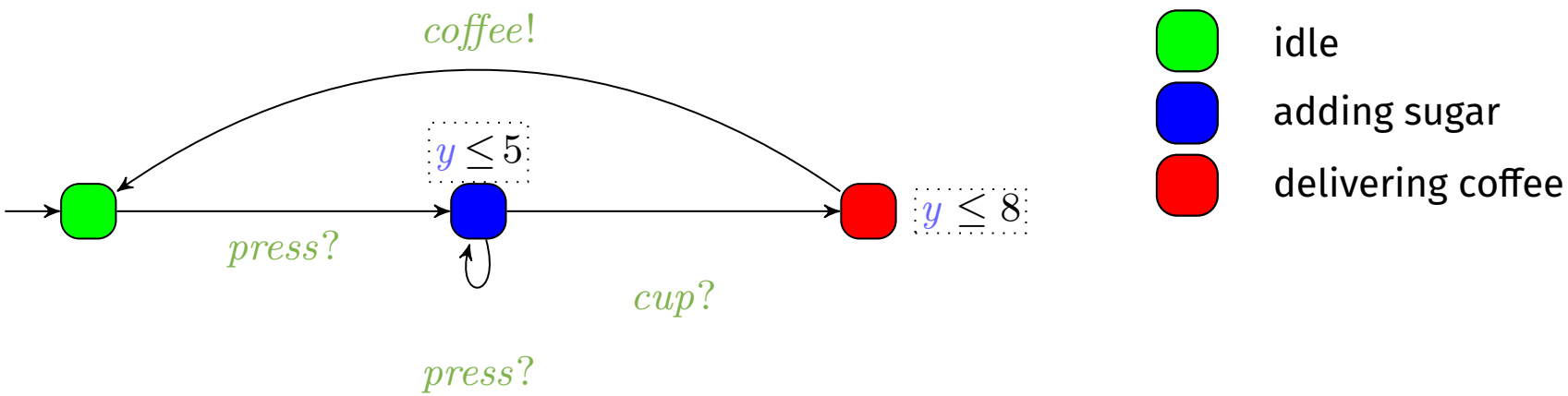


[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: Theoretical Computer Science 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants

- Features
  - Location invariant: property to be verified to stay at a location

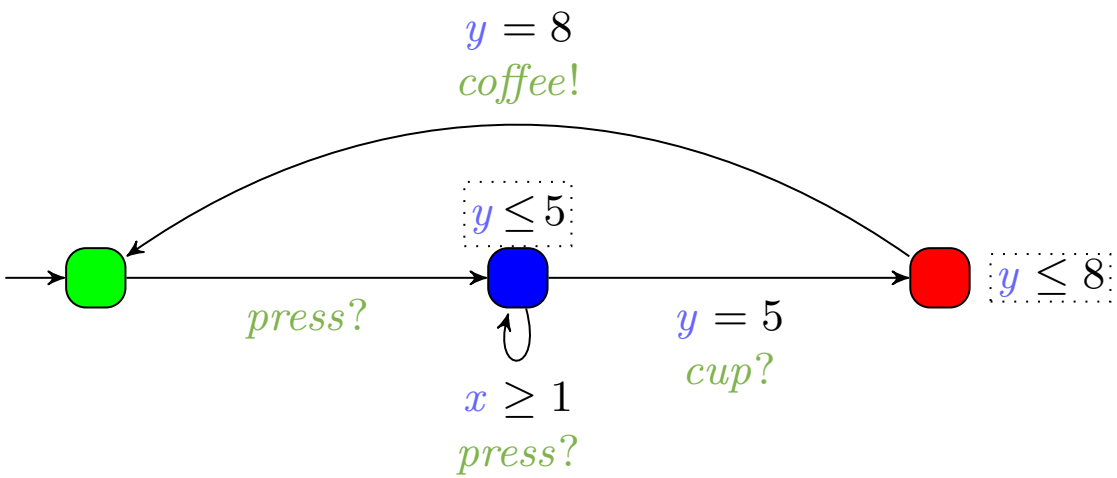


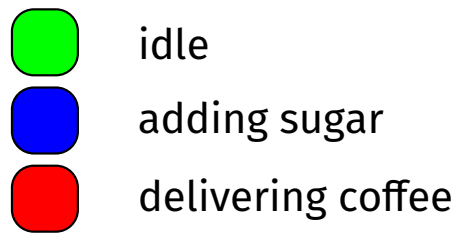
[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: Theoretical Computer Science 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition



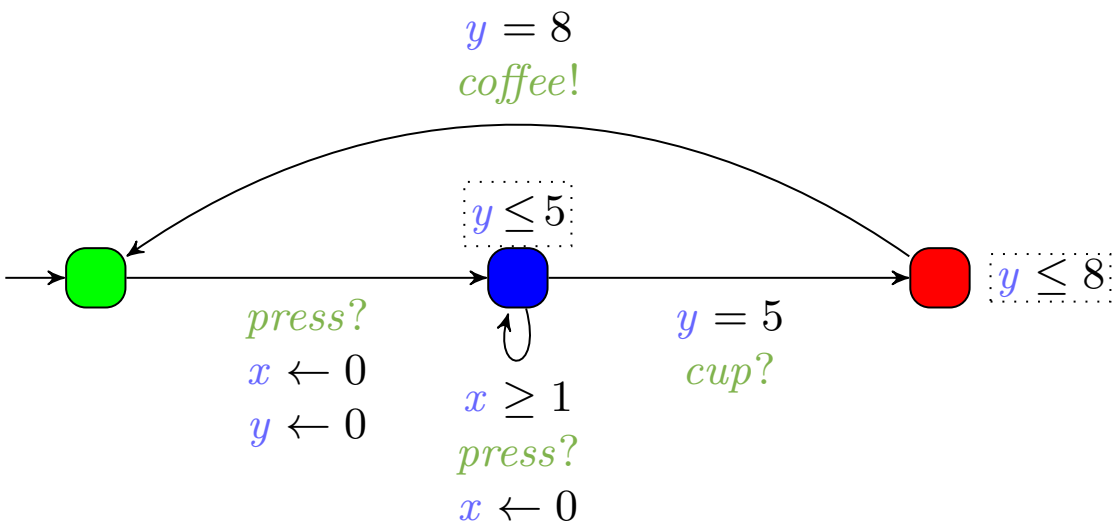


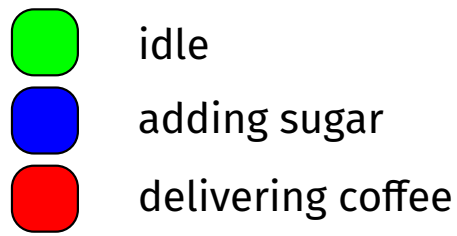
[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: Theoretical Computer Science 126.2 (Apr. 1994), pp. 183–235

Timed automaton (TA)

- Finite-state automaton (sets of locations and actions) augmented with a set X of clocks [AD94]
  - Real-valued variables evolving linearly at the same rate
  - Can be compared to integer constants in invariants and guards

- Features
  - Location invariant: property to be verified to stay at a location
  - Transition guard: property to be verified to enable a transition
  - Clock reset: some of the clocks can be set to 0 along transitions





[AD94] Rajeev Alur and David L. Dill. "A theory of timed automata". In: Theoretical Computer Science 126.2 (Apr. 1994), pp. 183–235

Parametric timed model checking in a nutshell

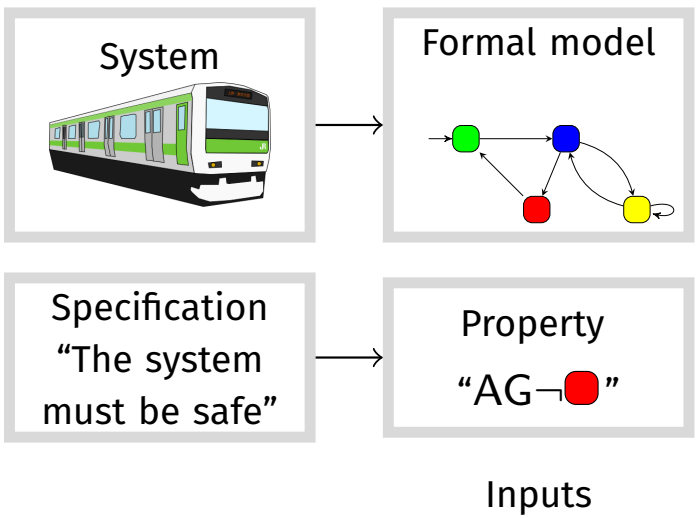
- Principle of parametric timed model checking

System

Specification
“The system must be safe”

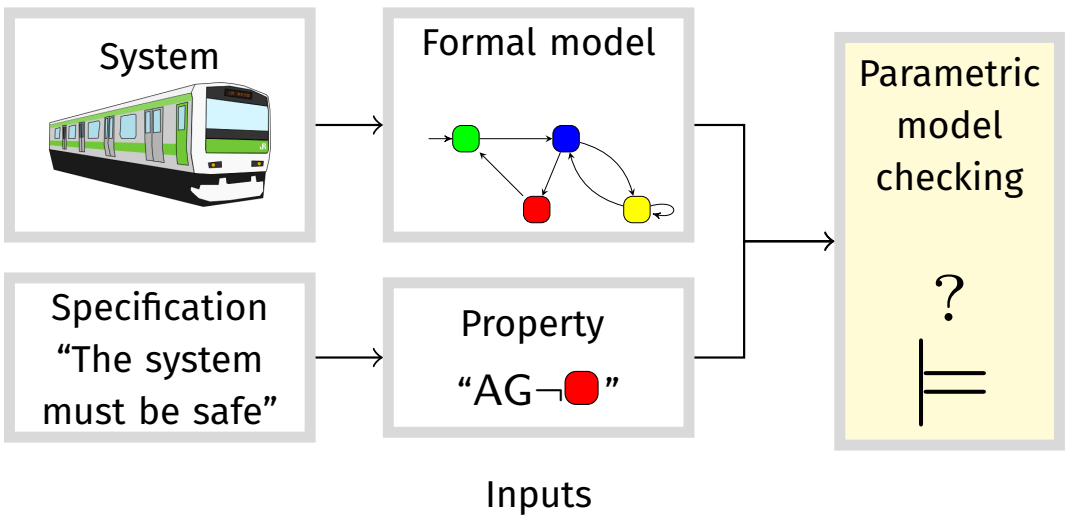
Parametric timed model checking in a nutshell

- Principle of parametric timed model checking



Parametric timed model checking in a nutshell

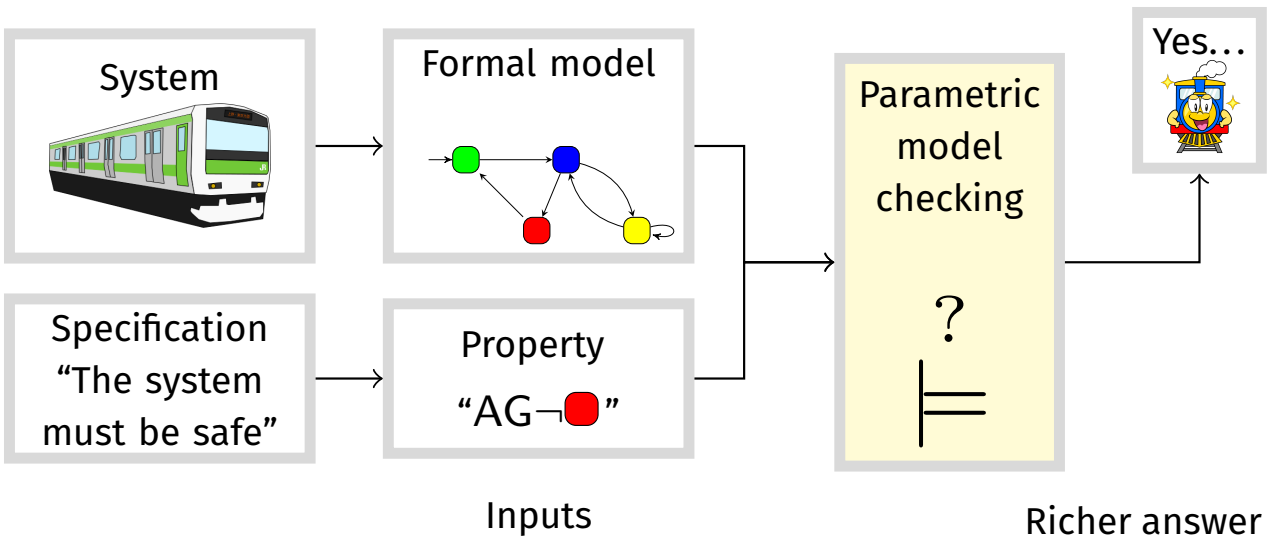
- Principle of parametric timed model checking



- Question: for which values of the design parameters does the model of the system satisfy the property?

Parametric timed model checking in a nutshell

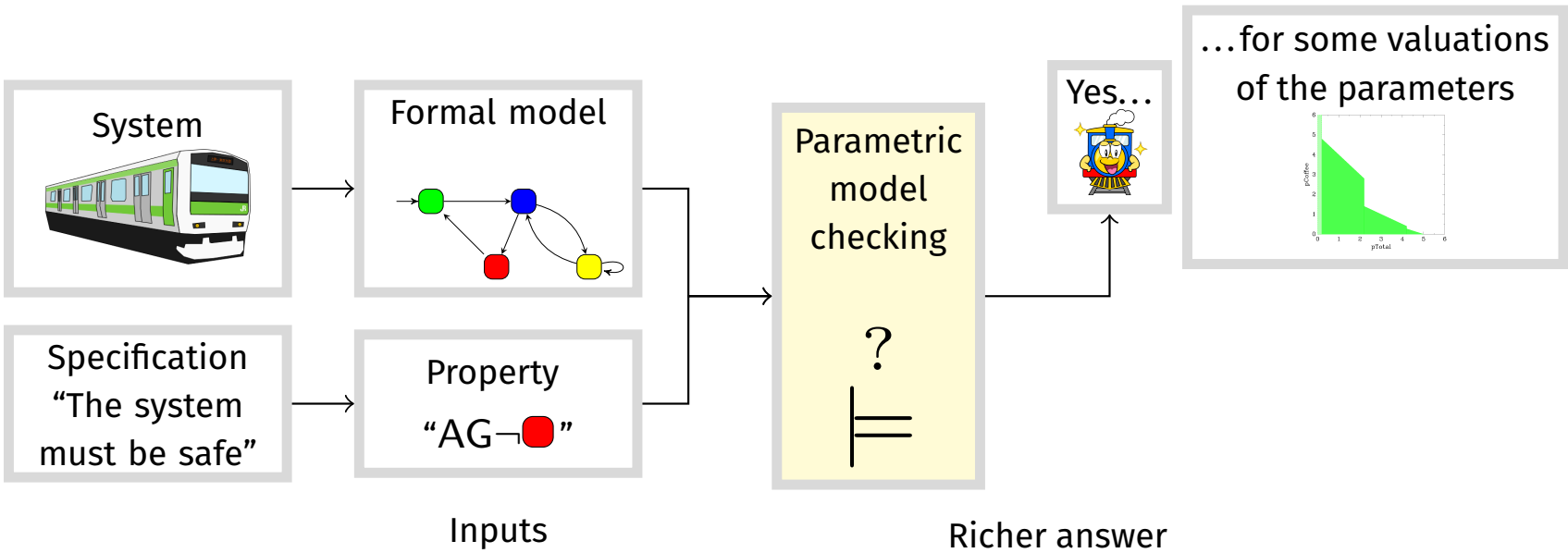
- Principle of parametric timed model checking



- Question: for which values of the design parameters does the model of the system satisfy the property?

Parametric timed model checking in a nutshell

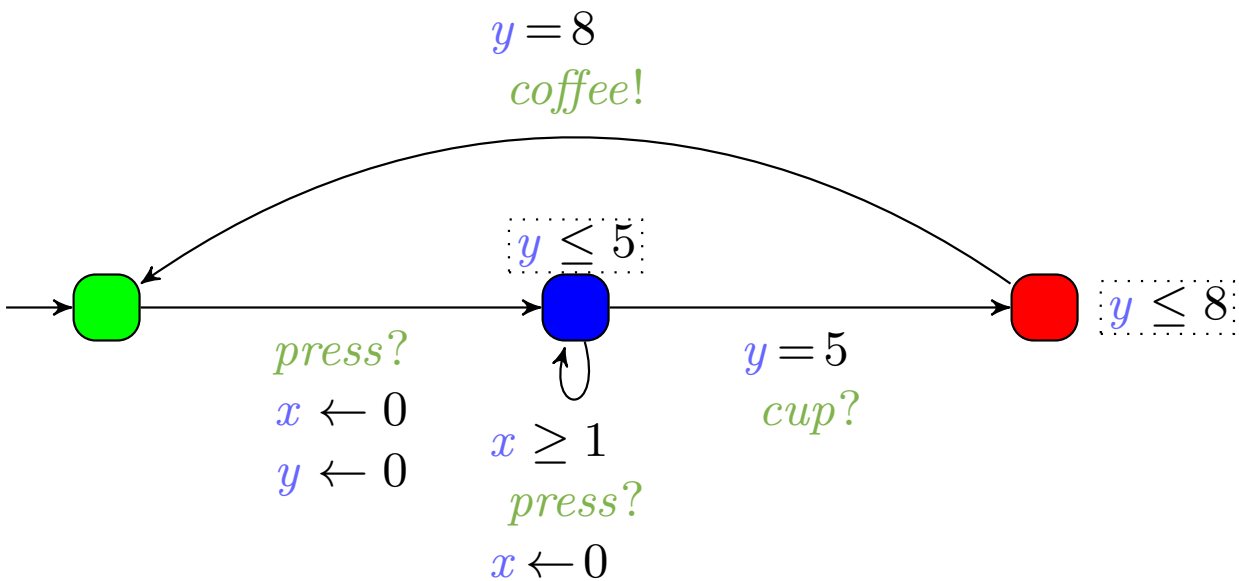
- Principle of parametric timed model checking



- Question: for which values of the design parameters does the model of the system satisfy the property?

Parametric Timed Automaton (PTA)

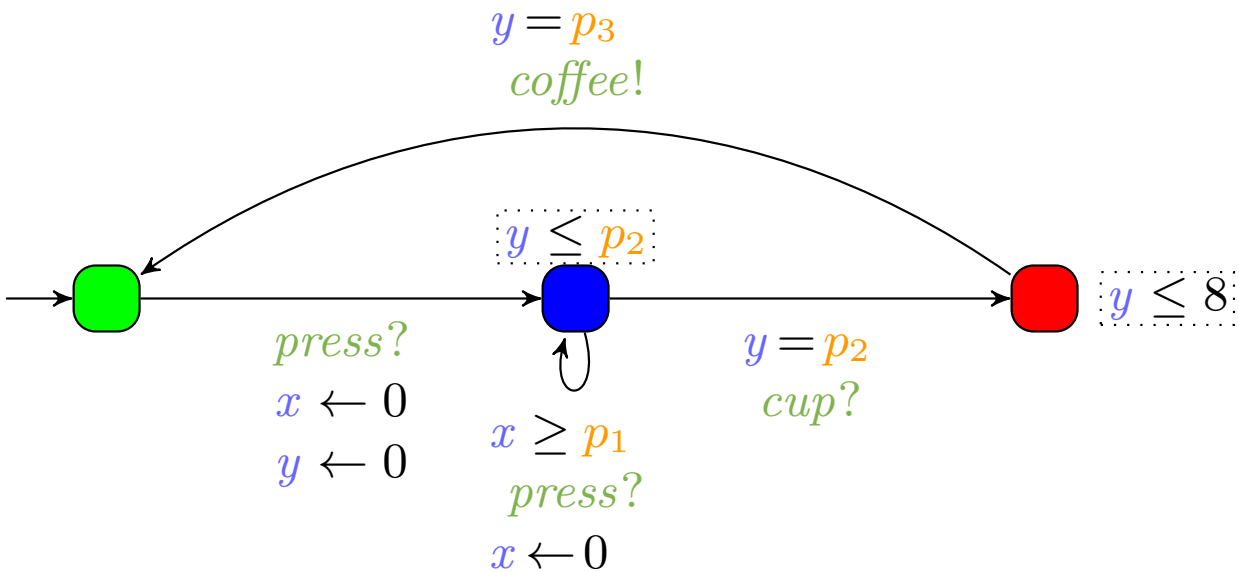
- Timed automaton (sets of locations, actions and clocks)



[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: STOC. ACM, 1993, pp. 592–601

Parametric Timed Automaton (PTA)

- Timed automaton (sets of locations, actions and clocks) augmented with a set $\mathbb{P}$ of parameters [AHV93]
  - Unknown constants compared to a clock in guards and invariants



[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: STOC. ACM, 1993, pp. 592–601

Outline

3 Translation
- (Parametric) timed model checking
- Modular translation

Overview of the translation

Modular translation

- Each train is translated into an independent (P)TA
- Each schedule constraint is translated into an independent (P)TA
- Components communicate through shared variables and synchronization actions

Verification

- A schedule exists if a given target location can be reached
  - Corresponding to all trains completing their connections, and all schedule constraints being met

Synthesis

- All timing parameters for which the target location can be reached are solutions to the problem

Translating trains and segments

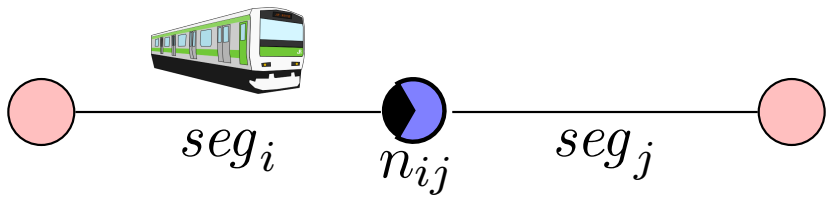
Concurrent and real-time system:

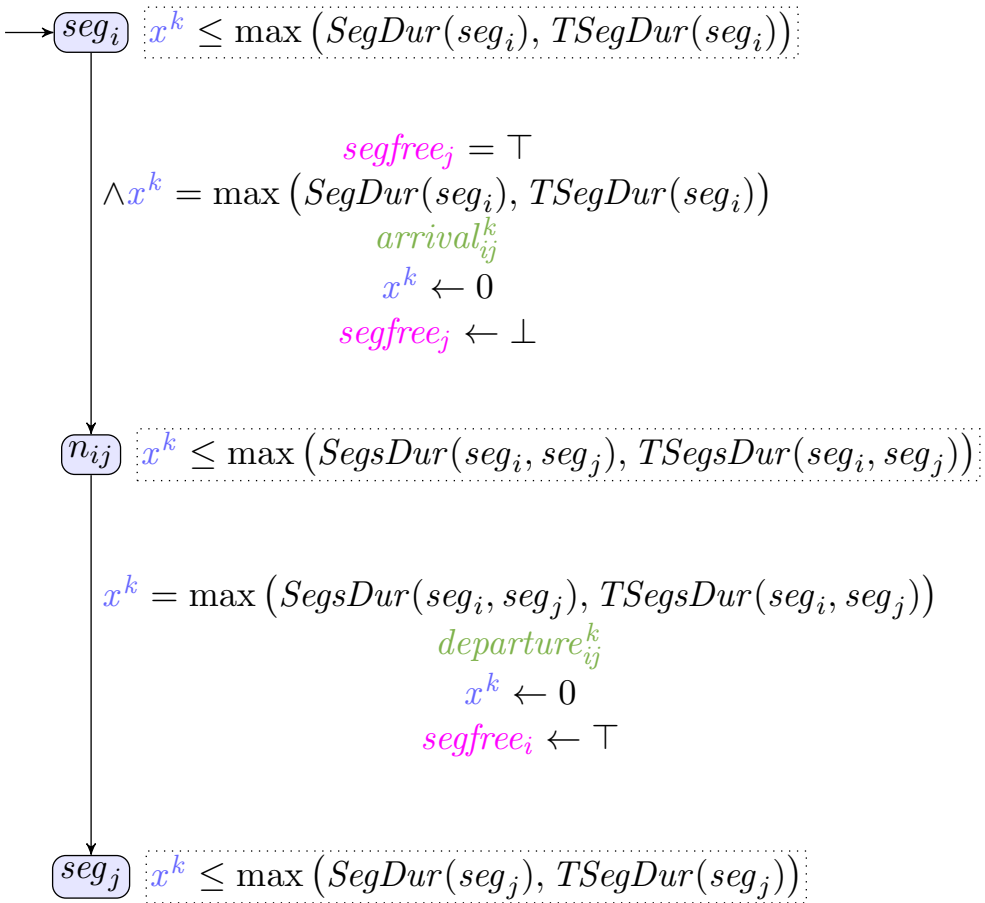
- ☹ a simple discrete graph with, e. g., a list of trains currently at each node, is not a suitable approach

Here: fully distributed approach

- each train evolves in its own representation of the rail network graph, in a continuous manner
- we define for each train k a PTA (with a single clock x^k), with the set of locations being made of the segments and the node of the network
- mutual exclusion in segments and nodes is ensured using global Boolean variables
  - tested and updated when attempting to enter, and when exiting a segment or node

Translation: train k moves from seg_i to seg_j via node n_{ij}

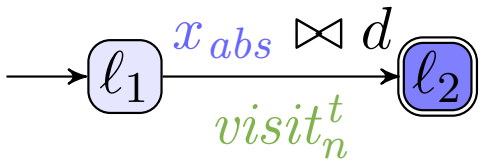




- x^k: clock of train k
- $SegDur(seg_i)$: segment traversal time at maximal speed
- $TSegDur(seg_i)$: segment traversal time at maximal speed for this train
- $segfree_i$: Boolean encoding that segment seg_i is free
- $arrival_{ij}^k$: synchronization action denoting train k reached node n_{ij}
- $departure_{ij}^k$: synchronization action denoting train k departed from node n_{ij}

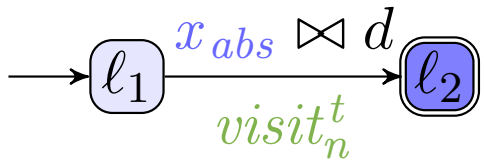
Translating schedule constraints

- Modeling constraint $visit(t, n) \bowtie d$
  - Featuring a (unique) absolute time clock (never reset) x_{abs}

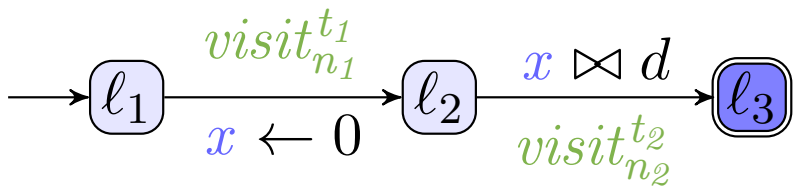


Translating schedule constraints

- Modeling constraint $visit(t, n) \bowtie d$
  - Featuring a (unique) absolute time clock (never reset) x_{abs}

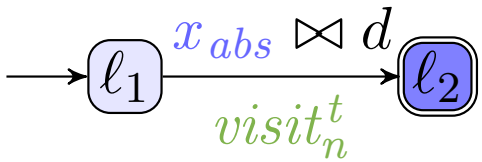


- Modeling constraint $transfer\big(visit_1(t_1, n_1), visit_2(t_2, n_2)\big) \bowtie d$
  - Featuring a dedicated (local) clock x

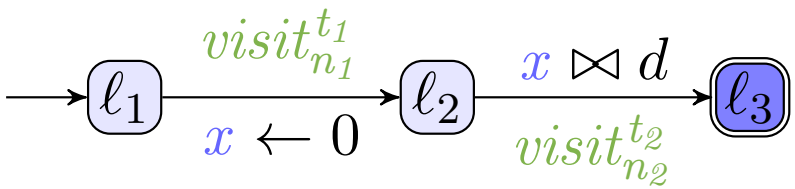


Translating schedule constraints

- Modeling constraint $visit(t, n) \bowtie d$
  - Featuring a (unique) absolute time clock (never reset) x_{abs}



- Modeling constraint $transfer\big(visit_1(t_1, n_1), visit_2(t_2, n_2)\big) \bowtie d$
  - Featuring a dedicated (local) clock x



- (Other constructs: see paper)

Outline

1 System model

2 Problem

3 Translation

4 Experiments

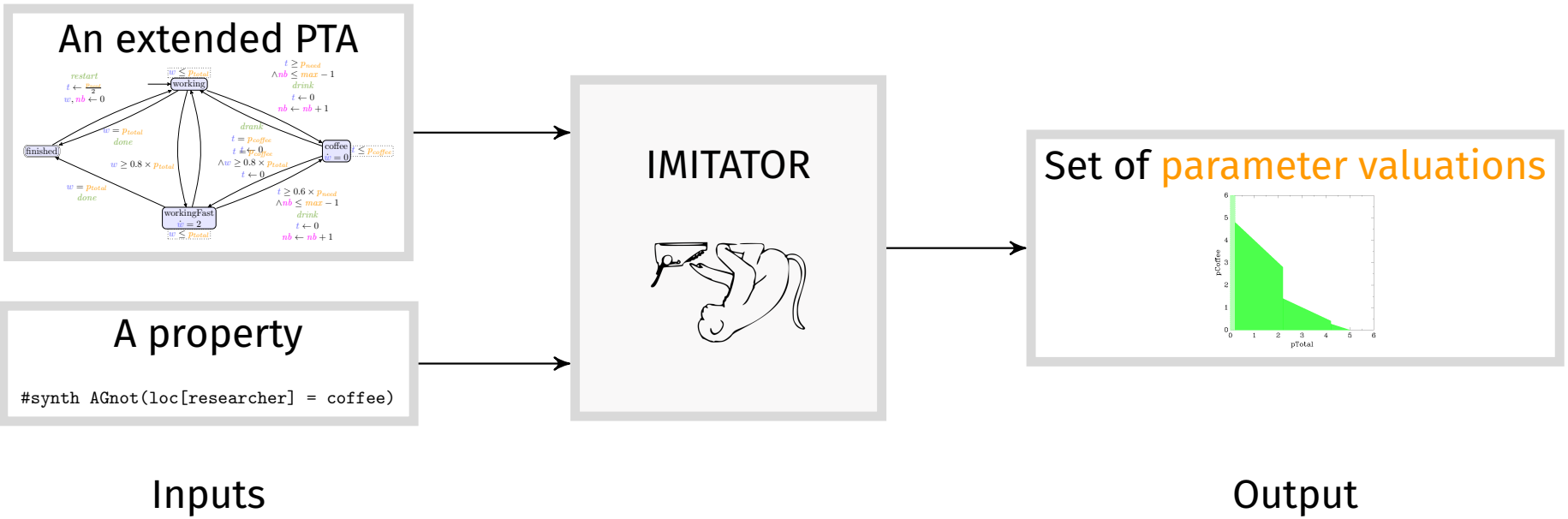
5 Perspectives

Outline

4 Experiments
- IMITATOR in a nutshell
- Experiments
- Benchmark 1: running example
- Benchmark 2: serial-parallel infrastructure

Parameter synthesis using IMITATOR 3

IMITATOR is a parametric timed model checker [And21]



The set of parameter valuations is symbolic

- Symbolic: finite set of linear constraints (polyhedra)

[And21] Étienne André. "IMITATOR 3: Synthesis of timing parameters beyond decidability". In: CAV. vol. 12759. Lecture Notes in Computer Science. Springer, 2021, pp. 1–14

Some algorithms implemented in IMITATOR

- Reachability-synthesis
- Safety-synthesis
- Liveness-synthesis
- Subset of TCTL-synthesis (including AF, AG, EU, EW…)
- Robustness analysis
- Minimal time synthesis
- Optimal parameter synthesis

Some algorithms implemented in IMITATOR

- Reachability-synthesis
- Safety-synthesis
- Liveness-synthesis
- Subset of TCTL-synthesis (including AF, AG, EU, EW…)
- Robustness analysis
- Minimal time synthesis
- Optimal parameter synthesis

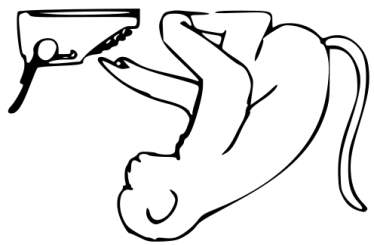
Distribution

Free and open source software: Available under the GNU-GPL license

Distribution:

- Binaries available for Linux platforms (no dependency, no install)
- Docker version
- Version 3.0 integrated as a virtual machine doi.org/10.5281/zenodo.4723415
- Comes with a user manual and an extensive benchmarks library [AMP21]

Try it!



www.imitator.fr

[AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. "A Benchmarks Library for Extended Timed Automata". In: TAP. vol. 12740. Lecture Notes in Computer Science. Springer, 2021, pp. 39–50

Outline

4 Experiments
- IMITATOR in a nutshell
- Experiments
- Benchmark 1: running example
- Benchmark 2: serial-parallel infrastructure

Experiments

Two case studies

1. The running example without and with timing parameters
2. A serial-parallel infrastructure to test scalability

Environment and data

- Version used: IMITATOR 3.4-alpha “Cheese Durian” (build `feat/forall_actions/788f551`)

- Models and results can be found and reproduced online

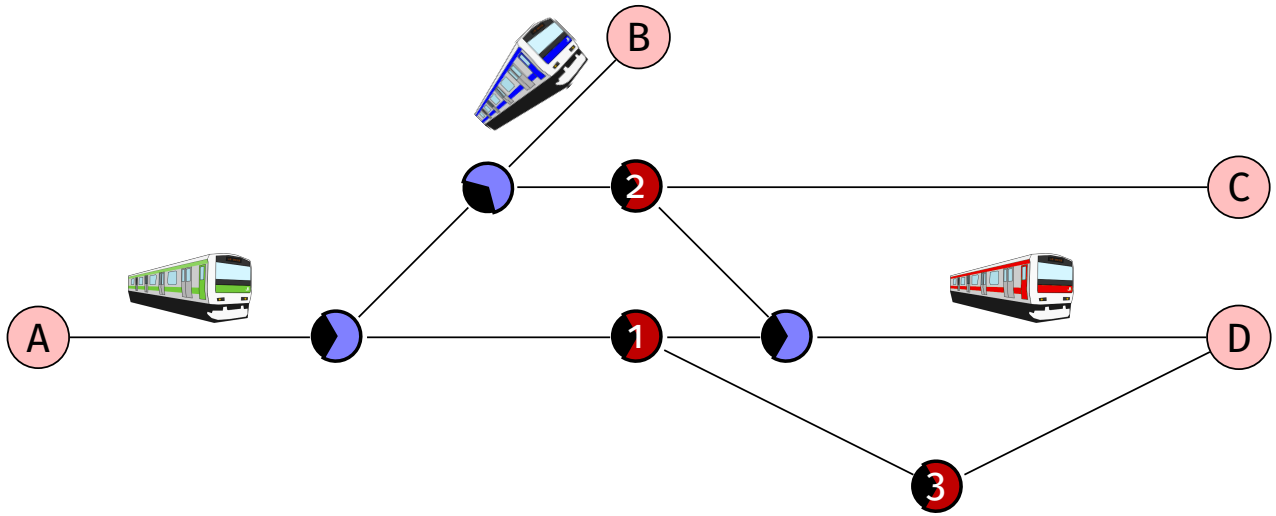
`https://www.imitator.fr/static/ICFEM24` ; long-term at `https://doi.org/10.5281/zenodo.13789618`

Outline

4 Experiments

- IMITATOR in a nutshell
- Experiments
- Benchmark 1: running example
- Benchmark 2: serial-parallel infrastructure

Running example: without timing parameterss



IMITATOR result:

- ☺ the train trajectory problem is satisfied, i. e., there exists a schedule such that all trains meet their connections and schedule constraints

IMITATOR execution time: 1.24 s

Running example: with timing parameters (1/2)

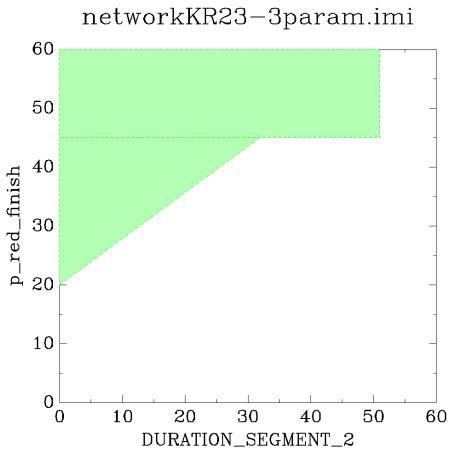
- We add a parametric schedule constraint $visit(t_{red}, A) \leq p_R$, with $p_R \in \mathbb{P}$
  - p_R: upper bound such that the red train reaches its destination
  - IMITATOR result: $p_R \geq 36$
    i. e., the red train cannot be faster than 36 time units
  - execution time: 1.91 s

Running example: with timing parameters (2/2)

- we parametrize in addition the minimum duration for segments 2 and 7
  - $SegDur(seg_2) = p_2$ and $SegDur(seg_7) = p_7$, with $p_2, p_7 \in \mathbb{P}$.
  - IMITATOR result:

$$(p_R \geq p_2 + 28) \vee (p_R \geq p_2 + p_7 + 20) \vee (p_R \geq 45)$$

  - execution time: 9.83 s

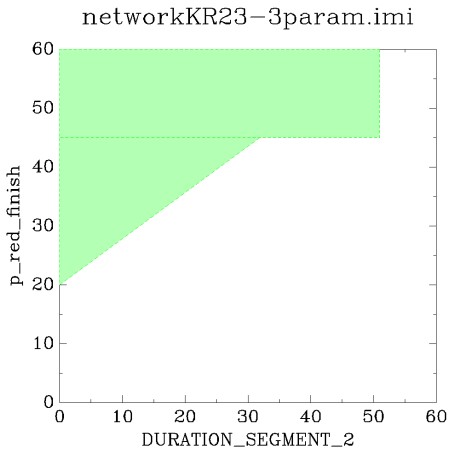


Running example: with timing parameters (2/2)

- we parametrize in addition the minimum duration for segments 2 and 7
  - $SegDur(seg_2) = p_2$ and $SegDur(seg_7) = p_7$, with $p_2, p_7 \in \mathbb{P}$.
  - IMITATOR result:

$$(p_R \geq p_2 + 28) \vee (p_R \geq p_2 + p_7 + 20) \vee (p_R \geq 45)$$

  - execution time: 9.83 s



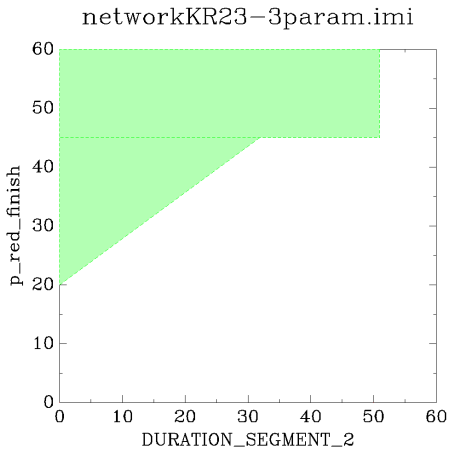
Correct (sound and complete) condition over the segment durations and schedule constraints parameters such that all train connections and schedule constraints are met

Running example: with timing parameters (2/2)

- we parametrize in addition the minimum duration for segments 2 and 7
  - $SegDur(seg_2) = p_2$ and $SegDur(seg_7) = p_7$, with $p_2, p_7 \in \mathbb{P}$.
  - IMITATOR result:

$$(p_R \geq p_2 + 28) \vee (p_R \geq p_2 + p_7 + 20) \vee (p_R \geq 45)$$

  - execution time: 9.83 s



Correct (sound and complete) condition over the segment durations and schedule constraints parameters such that all train connections and schedule constraints are met

Major advantage over, e. g., SMT solvers, that would typically derive (non-necessarily complete) discrete sets of specific valuations

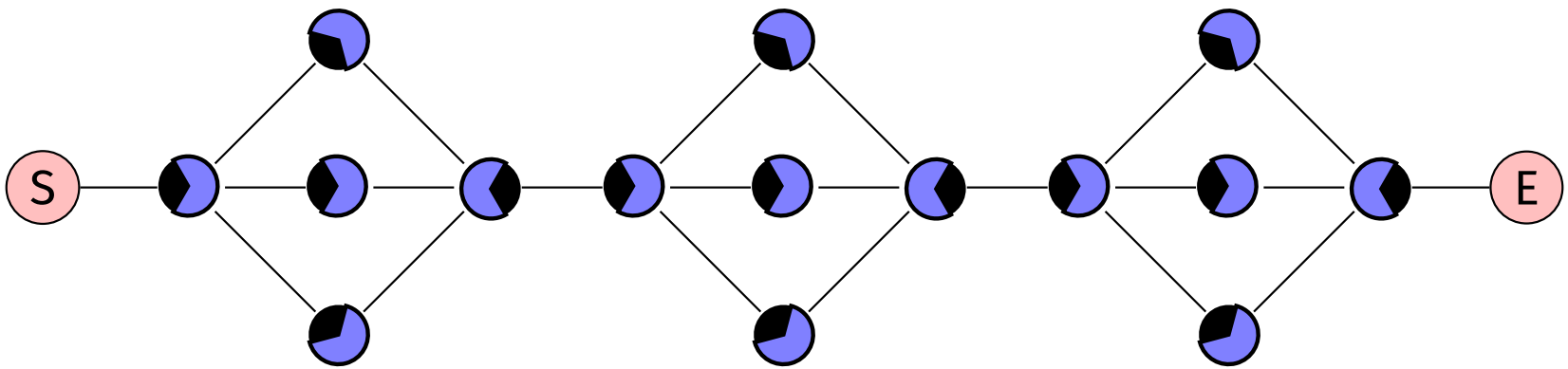
Outline

4 Experiments
- IMITATOR in a nutshell
- Experiments
- Benchmark 1: running example
- Benchmark 2: serial-parallel infrastructure

A serial-parallel infrastructure [KR23]

Goal: evaluate the scalability

- network with N_S serially connected groups of N_P identical parallel tracks with a station



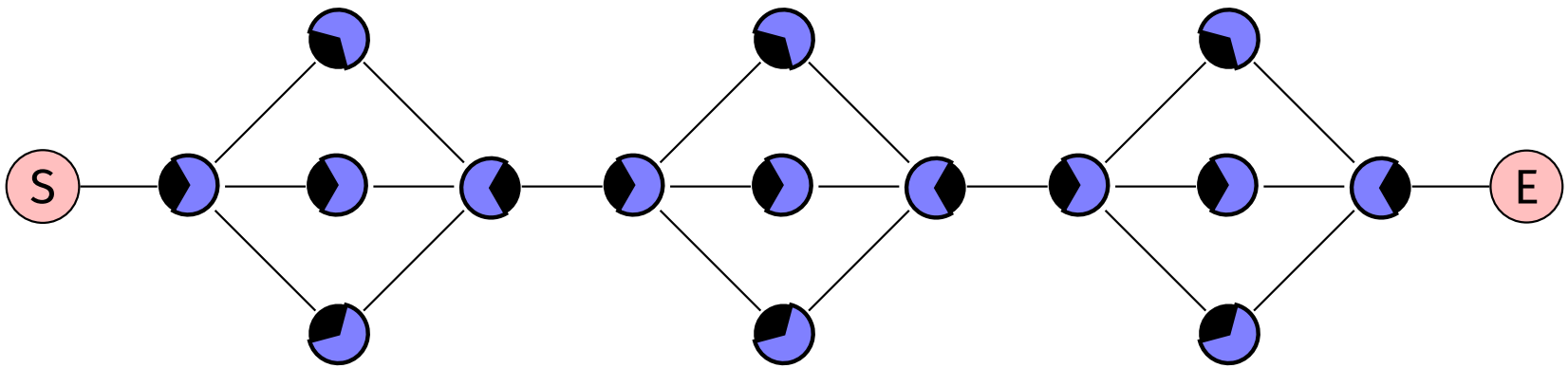
Here: $N_S = N_P = 3$

[KR23] Tomáš Kolárik and Stefan Ratschan. "Railway Scheduling Using Boolean Satisfiability Modulo Simulations". In: FM. vol. 14000. Lecture Notes in Computer Science. Springer, 2023, pp. 56–73

A serial-parallel infrastructure [KR23]

Goal: evaluate the scalability

- network with N_S serially connected groups of N_P identical parallel tracks with a station
- connections only include $[S, E]$, i. e., trains cannot stop at any station, and are free to use any path
  - Variety of choices for each train: exponential blowup



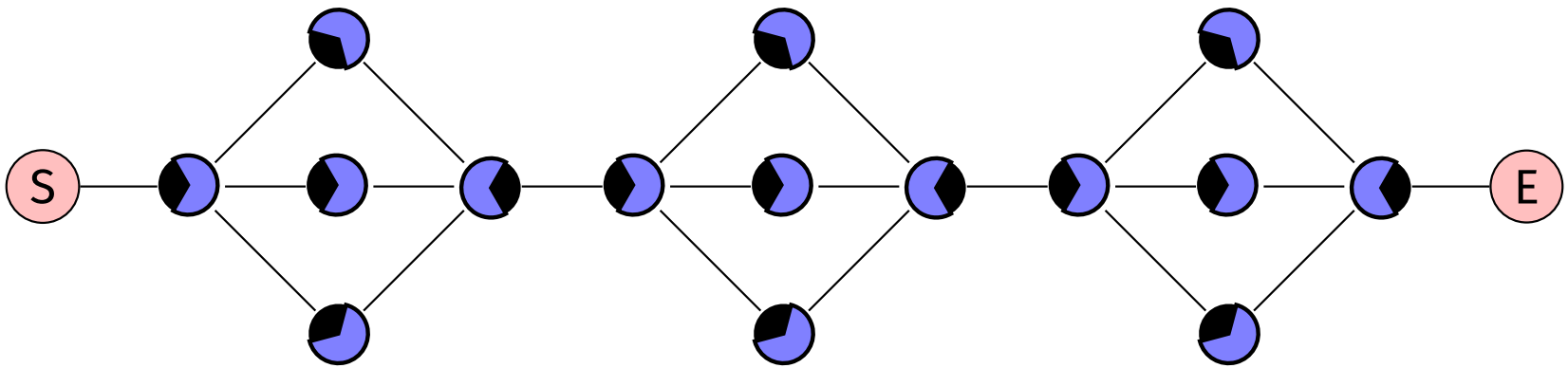
Here: $N_S = N_P = 3$

[KR23] Tomáš Kolárik and Stefan Ratschan. "Railway Scheduling Using Boolean Satisfiability Modulo Simulations". In: FM. vol. 14000. Lecture Notes in Computer Science. Springer, 2023, pp. 56–73

A serial-parallel infrastructure [KR23]

Goal: evaluate the scalability

- network with N_S serially connected groups of N_P identical parallel tracks with a station
- connections only include $[S, E]$, i. e., trains cannot stop at any station, and are free to use any path
  - Variety of choices for each train: exponential blowup
- We consider various values for the number N_T of trains in the infrastructure, and for the number of groups N_S and parallel tracks N_P
  - we fix $N_S = N_P$



Here: $N_S = N_P = 3$

[KR23] Tomáš Kolárik and Stefan Ratschan. "Railway Scheduling Using Boolean Satisfiability Modulo Simulations". In: FM. vol. 14000. Lecture Notes in Computer Science. Springer, 2023, pp. 56–73

Scenario “nop”

“nop” scenario

- no parametric duration, nor any schedule constraint
- we add one parametric absolute timing constraint: $arrival(t_{N_T}, E) \leq J$,
  $\leadsto$ We measure the end-to-end time from the first train leaving S to the last train reaching E

Scenario “nop”

“nop” scenario

- no parametric duration, nor any schedule constraint
- we add one parametric absolute timing constraint: $arrival(t_{N_T}, E) \leq J$,
  $\rightsquigarrow$ We measure the end-to-end time from the first train leaving S to the last train reaching E

N_S	N_T	$\lvert\mathcal{A}\rvert$	$\lvert\mathbb{X}\rvert$	$\lvert\mathbb{P}\rvert$	$\lvert S\rvert$	J	$t(s)$
2	1	2	2	1	25	63	0.01
2	2	3	3	1	238	75	0.08
2	3	4	4	1	2323	87	2.68
2	4	5	5	1	22450	99	96.02
3	1	2	2	1	51	92	0.01
3	2	3	3	1	1237	104	0.68
3	3	4	4	1	27385	116	137.61
3	4	-	-	-	-	-	T.O.
4	1	2	2	1	87	121	0.03
4	2	3	3	1	3195	133	3.28
4	3	-	-	-	-	-	T.O.

- Clearly exponential (expected!)
- Interesting results nevertheless
- Automatic synthesis of J
  - instead of testing it

Scenario “last”

“last” scenario

- additionally ensures that the last train takes less than bnd time units between its departure and arrival
- Again, we parametrize this value (instead of manually enumerating it), by adding the following relative timing constraint:
  $transfer\big(departure(t_{N_T}, S), arrival(t_{N_T}, E)\big) \leq bnd$,

Scenario “last”

“last” scenario

- additionally ensures that the last train takes less than bnd time units between its departure and arrival
- Again, we parametrize this value (instead of manually enumerating it), by adding the following relative timing constraint:
  $transfer\big(departure(t_{N_T}, S), arrival(t_{N_T}, E)\big) \leq bnd$,

N_S	N_T	$\lvert\mathcal{A}\rvert$	$\lvert\mathbb{X}\rvert$	$\lvert\mathbb{P}\rvert$	$\lvert S\rvert$	J	bnd	$t(s)$
2	1	2	3	2	25	63	63	0.01
2	2	3	4	2	238	75	63	0.12
2	3	4	5	2	2323	87	63	3.40
2	4	5	6	2	22450	99	63	113.41
3	1	2	3	2	51	92	92	0.02
3	2	3	4	2	1052	104	92	0.91
3	3	4	5	2	27385	116	92	157.78
3	4	-	-	-	-	-	-	T.O.
4	1	2	3	2	87	121	121	0.04
4	2	3	4	2	3195	133	121	4.64
4	3	-	-	-	-	-	-	T.O.

- Automatic synthesis of J and bnd (instead of manually iterating as in the literature)

Outline

1 System model

2 Problem

3 Translation

4 Experiments

5 Perspectives

Perspectives: improving integration

So far

- ad hoc input format
- ad hoc template-based encoding

Perspectives

- industry-standard input
- fully automated translation and verification
- schedule automatically derived in a standard formalism
- selections of “best” schedule (optimal / robust / …)

Perspectives: improving expressiveness

- Add optimality criteria (partially done)
- Finer encoding of acceleration and deceleration, as in [KR23]
- Add energy
  - with optimality criteria
  - encode energy transfer from deceleration into acceleration of other trains
- Moving block system (MBS)

[KR23] Tomáš Kolárik and Stefan Ratschan. "Railway Scheduling Using Boolean Satisfiability Modulo Simulations". In: FM. vol. 14000. Lecture Notes in Computer Science. Springer, 2023, pp. 56–73

Bibliography

References I

[AD94] Rajeev Alur and David L. Dill. “A theory of timed automata”. In: Theoretical Computer Science 126.2 (Apr. 1994), pp. 183–235. doi: 10.1016/0304-3975(94)90010-8.

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: STOC (May 16–18, 1993). Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. San Diego, California, USA: ACM, 1993, pp. 592–601. doi: 10.1145/167088.167242.

[AMP21] Étienne André, Dylan Marinho, and Jaco van de Pol. “A Benchmarks Library for Extended Timed Automata”. In: TAP (June 21–25, 2021). Ed. by Frédéric Loulergue and Franz Wotawa. Vol. 12740. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 39–50. doi: ABenchmarksLibraryforExtendedParametricTimedAutomata.

[And21] Étienne André. “IMITATOR 3: Synthesis of timing parameters beyond decidability”. In: CAV (July 18–23, 2021). Ed. by Rustan Leino and Alexandra Silva. Vol. 12759. Lecture Notes in Computer Science. virtual: Springer, 2021, pp. 1–14. doi: 10.1007/978-3-030-81685-8_26.

[Bud+17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. “JANI: Quantitative Model and Tool Interaction”. In: TACAS (Apr. 22–29, 2017). Ed. by Axel Legay and Tiziana Margaria. Vol. 10206. Lecture Notes in Computer Science. Uppsala, Sweden, 2017, pp. 151–168. doi: 10.1007/978-3-662-54580-5_9.

[CL00] Franck Cassez and Kim Guldstrand Larsen. “The Impressive Power of Stopwatches”. In: CONCUR (Aug. 22–25, 2000). Ed. by Catuscia Palamidessi. Vol. 1877. Lecture Notes in Computer Science. University Park, PA, USA: Springer, 2000, pp. 138–152. doi: 10.1007/3-540-44618-4_12.

References II

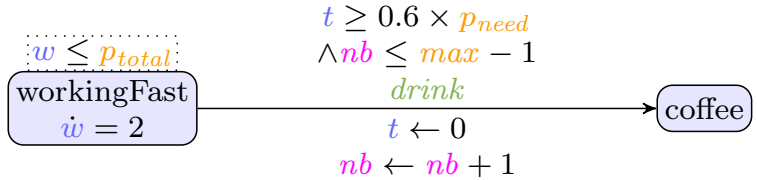
[KR23] Tomáš Kolárik and Stefan Ratschan. “Railway Scheduling Using Boolean Satisfiability Modulo Simulations”. In: FM (Mar. 6–10, 2023). Ed. by Marsha Chechik, Joost-Pieter Katoen, and Martin Leucker. Vol. 14000. Lecture Notes in Computer Science. Lübeck, Germany: Springer, 2023, pp. 56–73. doi: 10.1007/978-3-031-27481-7_5.

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. “UPPAAL in a Nutshell”. In: International Journal on Software Tools for Technology Transfer 1.1-2 (1997), pp. 134–152. doi: 10.1007/s100090050010.

Backup slides on IMITATOR

Input syntax

- Text-based (originally inspired by HyTech)
- Human-friendly




```
loc workingFast: invariant w <= pTotal flow{w' = 2}
  when t >= 0.6 * pNeed & nb <= max - 1 sync drink do {t <- 0,
     nb <- nb + 1} goto coffee;
```

- Conversions to other formats
  - UppAal [LPY97] (losing parameters!)
  - JANI [Bud+17]
    - An interchange format for automata-based formalisms

[LPY97] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. "UPPAAL in a Nutshell". In: International Journal on Software Tools for Technology Transfer 1.1-2 (1997), pp. 134–152

[Bud+17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. "JANI: Quantitative Model and Tool Interaction". In: TACAS. vol. 10206. Lecture Notes in Computer Science. 2017, pp. 151–168

Beyond decidability

- ☺ Timed automata benefit from (some) decidability results

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. “Parametric real-time reasoning”. In: STOC. ACM, 1993, pp. 592–601

[CL00] Franck Cassez and Kim Guldstrand Larsen. “The Impressive Power of Stopwatches”. In: CONCUR. vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 138–152

Beyond decidability

- ☺ Timed automata benefit from (some) decidability results
- ☹ Adding parameters yields undecidability [AHV93]
- ☹ Adding stopwatches yields undecidability [CL00]
- ☹ Adding unbounded rational variables yields undecidability

---

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: STOC. ACM, 1993, pp. 592–601

[CL00] Franck Cassez and Kim Guldstrand Larsen. "The Impressive Power of Stopwatches". In: CONCUR. vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 138–152

Beyond decidability

- ☺ Timed automata benefit from (some) decidability results
- ☹ Adding parameters yields undecidability [AHV93]
- ☹ Adding stopwatches yields undecidability [CL00]
- ☹ Adding unbounded rational variables yields undecidability

IMITATOR paradigm: "best effort"

Try to synthesize parameter valuations

- No guarantee of termination, or
- Under or over-approximations and inform the user about them
  - Evaluate whether a result is exact, over-approximated, under-approximated, or possibly invalid

---

[AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. "Parametric real-time reasoning". In: STOC. ACM, 1993, pp. 592–601

[CL00] Franck Cassez and Kim Guldstrand Larsen. "The Impressive Power of Stopwatches". In: CONCUR. vol. 1877. Lecture Notes in Computer Science. Springer, 2000, pp. 138–152

Safety

Synthesize all parameter valuations for which the following property holds:

"It is impossible to drink any coffee"

(i. e., the coffee location is unreachable)

```
#synth AGnot(loc[researcher] = coffee)
```

Safety

Synthesize all parameter valuations for which the following property holds:

"It is impossible to drink any coffee"

(i. e., the coffee location is unreachable)

```
#synth AGnot(loc[researcher] = coffee)
```

Result:

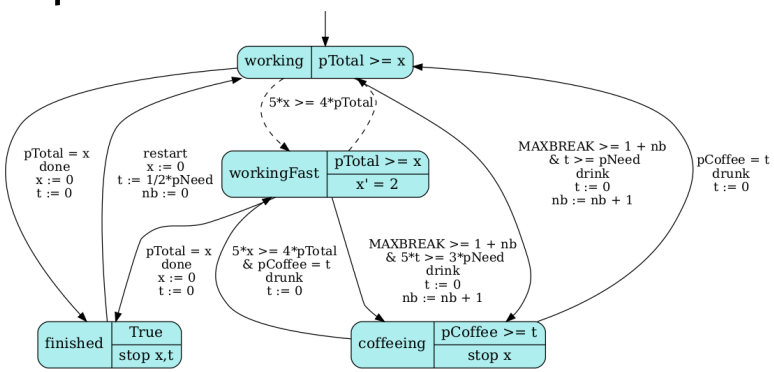
$$max \in [0, 1) \vee \left(max \geq 1 \wedge p_{total} < \frac{p_{need}}{10}\right)$$

Results

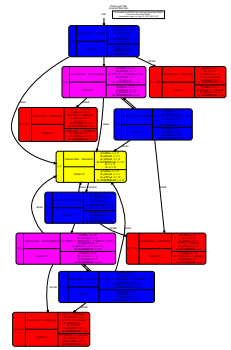
- Normalized text results

```
 * Result by: IMITATOR 3.0 "Cheese" (build HEAD/ea560fd)
 * Model    : 'researcher.imi'
 * Generated: Mon Feb 1, 2021 14:57:17
 * Command  : imitator3 researcher.imi researcher-AGnotcoffee.imiprop
 ************************************************************)

BEGIN CONSTRAINT
 pTotal >= 0
& pNeed >= 1
& MAXBREAK >= 0
& pCoffee >= 0
& 1 > MAXBREAK
OR
  pNeed > 10*pTotal
& pTotal >= 0
& pNeed >= 1
& MAXBREAK >= 1
& pCoffee >= 0
END CONSTRAINT

------------------------------------------------------------
Constraint soundness                    : exact
Termination                             : regular termination
------------------------------------------------------------
```

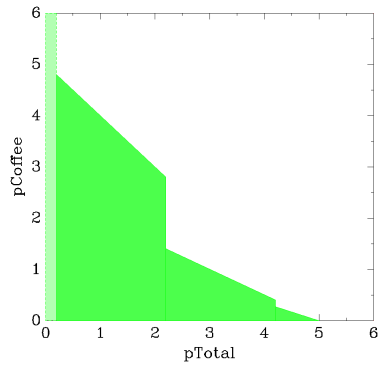
- Graphical results



PTA visualization



State space (zone graph)



Constraints representation

Licensing

Source of the graphics used I

Title : 6 Tote bei einem Zugunglück in Zoufftgen, eine Regionalbahn kollidierte mit einem Güte
Author : Jürgen Jung
Source : https://commons.wikimedia.org/wiki/File:Zugunglück_zoufftgen.jpg
License : Creative Commons cc-by

Title : Night scene of the wreck resulting from the Wenzhou train collision
Author : China Youth Online
Source : https://en.wikipedia.org/wiki/File:Wenzhou_PDL_wreck_at_night.jpg
License : Copyright

Title : 2021 Hualien train derailment
Author : Taiwan Government
Source : https://en.wikipedia.org/wiki/2021_Hualien_train_derailment
License : Taiwan Government Website Open Information Announcement

Title : Vector image of Tokyo train on the Yamanote line
Author : j4p4n
Source : https://freesvg.org/japanese-train
License : Public domain

Title : Glitter Train Cartoon
Author : DSIGNALIKIE

Source of the graphics used II

Source : https://charatoon.com/?id=7048
License : Copyright

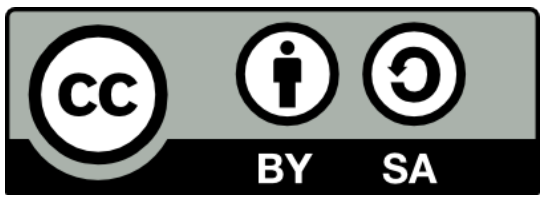
Title : Crying Train Cartoon
Author : DSIGNALIKIE
Source : https://charatoon.com/?id=7053
License : Copyright

License of this document

This presentation can be published, reused and modified under the terms of the license Creative Commons Attribution-ShareAlike 4.0 Unported (CC BY-SA 4.0)

(LaTeX source available upon request)

Author: Étienne André



creativecommons.org/licenses/by-sa/4.0/