Iterative Bounded Synthesis for Efficient Cycle Detection in Parametric Timed Automata

Étienne André[1] Jaime Arias[2] Laure Petrucci[2] Jaco van de Pol[3]

[1]Université de Lorraine, CNRS, LORIA, Nancy, France
[2]LIPN, CNRS UMR 7030, Université Sorbonne Paris Nord, France
[3]Aarhus University, Denmark

SynCoP, 2 April 2022

Parametric Timed Automata

Ingredients of PTA (Alur, Henzinger, Vardi '93)

- Finite number of locations ℓ_0, ℓ_1, ℓ_2 . . . . transitions in between
- Clocks x, y, z . . . . advance at the same rate
- Guards, Invariants, Clock resets . . . . specify timing constraints
- Parameters p, q, r . . . . unknown constants used in constraints

Example: the light switch

Parametric Timed Automata

Ingredients of PTA (Alur, Henzinger, Vardi '93)

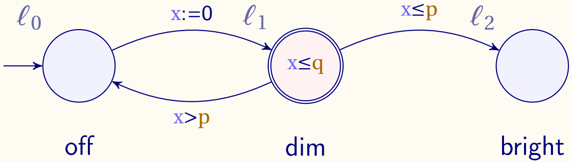
- Finite number of locations ℓ_0, ℓ_1, ℓ_2 .... transitions in between
- Clocks x, y, z .... advance at the same rate
- Guards, Invariants, Clock resets .... specify timing constraints
- Parameters p, q, r .... unknown constants used in constraints

Example: the light switch

Ingredients of PTA (Alur, Henzinger, Vardi '93)

- Finite number of locations ℓ_0, ℓ_1, ℓ_2 .... transitions in between
- Clocks x, y, z .... advance at the same rate
- Guards, Invariants, Clock resets .... specify timing constraints
- Parameters p, q, r .... unknown constants used in constraints

Example: the light switch



Parametric Synthesis

- Specification and Verification of Real-time systems
- Timing parameters unknown (at design time)
- Goal: synthesise parameter constraints for which requirements hold
- Here: Liveness properties – Büchi conditions

Example: there is an infinite accepting run if and only if $q > p$

Problem statement

- The problem is undecidable for PTA
- How to search an infinite state space for cycles?
- Examples and Semi-algorithms – completeness (in the limit)

Parametric Synthesis

- Specification and Verification of Real-time systems
- Timing parameters unknown (at design time)
- Goal: synthesise parameter constraints for which requirements hold
- Here: Liveness properties – Büchi conditions

Example: there is an infinite accepting run if and only if $q > p$

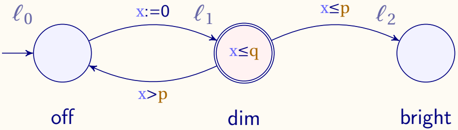
Problem statement

- The problem is undecidable for PTA
- How to search an infinite state space for cycles?
- Examples and Semi-algorithms – completeness (in the limit)

Parametric Synthesis

- Specification and Verification of Real-time systems
- Timing parameters unknown (at design time)
- Goal: synthesise parameter constraints for which requirements hold
- Here: Liveness properties – Büchi conditions

Example: there is an infinite accepting run if and only if $q > p$



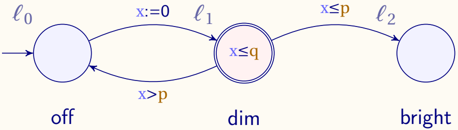
Problem statement

- The problem is undecidable for PTA
- How to search an infinite state space for cycles?
- Examples and Semi-algorithms – completeness (in the limit)

Parametric Synthesis

- Specification and Verification of Real-time systems
- Timing parameters unknown (at design time)
- Goal: synthesise parameter constraints for which requirements hold
- Here: Liveness properties – Büchi conditions

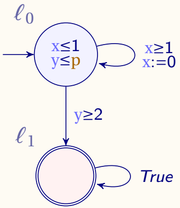
Example: there is an infinite accepting run if and only if $q > p$



Problem statement

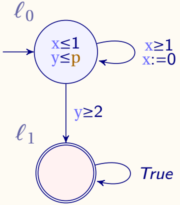
- The problem is undecidable for PTA
- How to search an infinite state space for cycles?
- Examples and Semi-algorithms – completeness (in the limit)

PTA

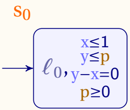


- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

PTA

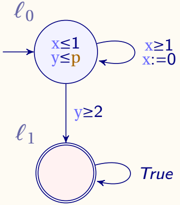


Parametric Zone Graph: (location, convex polyhedron)

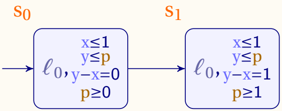


- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

PTA

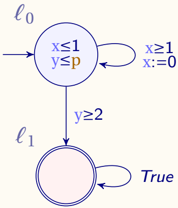


Parametric Zone Graph: (location, convex polyhedron)

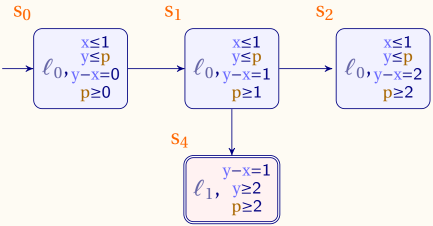


- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

PTA



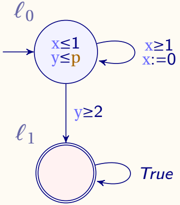
Parametric Zone Graph: (location, convex polyhedron)



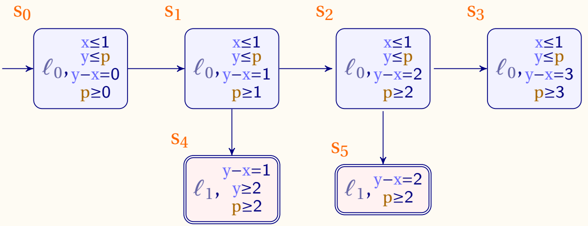
- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

Accepting-First and Pruning

PTA



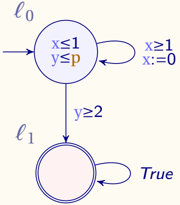
Parametric Zone Graph: (location, convex polyhedron)



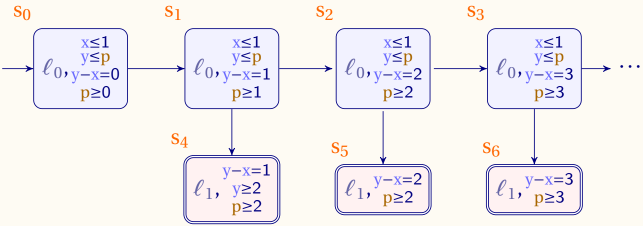
- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

Accepting-First and Pruning

PTA



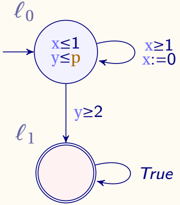
Parametric Zone Graph: (location, convex polyhedron)



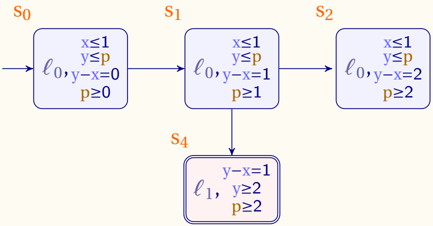
- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

Accepting-First and Pruning

PTA



Parametric Zone Graph: (location, convex polyhedron)

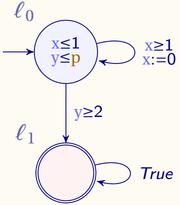


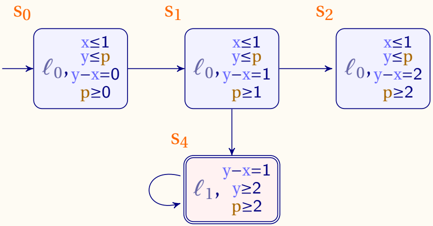
- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

Accepting-First and Pruning

PTA

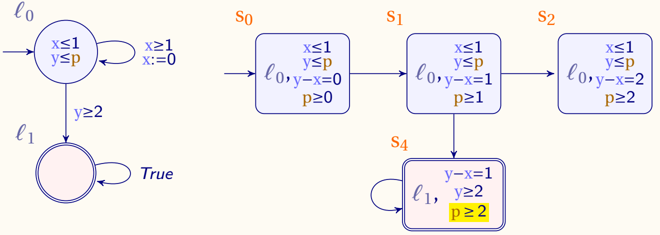
Parametric Zone Graph: (location, convex polyhedron)





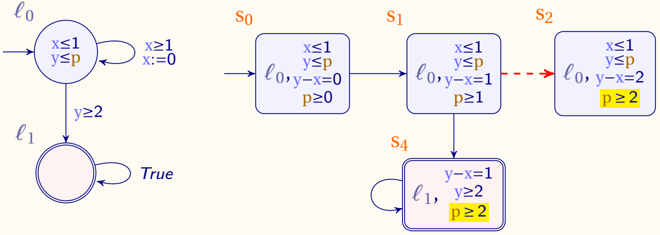
- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

PTA   Parametric Zone Graph: (location, convex polyhedron)



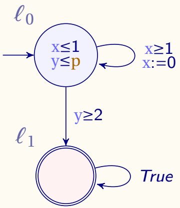
- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

PTA Parametric Zone Graph: (location, convex polyhedron)

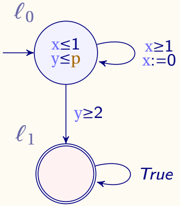


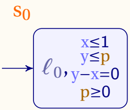
- Exploration using the self-loop on ℓ_0:
  Infinite branch without accepting cycle
- Accepting first strategy:
  Constraint synthesized: $p \geq 2$
- Exploration ends with cumulative pruning

Layering Strategy

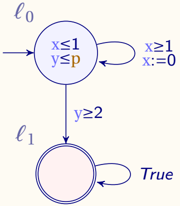


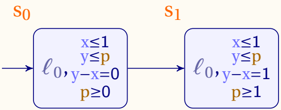
- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at ℓ_1
- Exploration ends with cumulative pruning





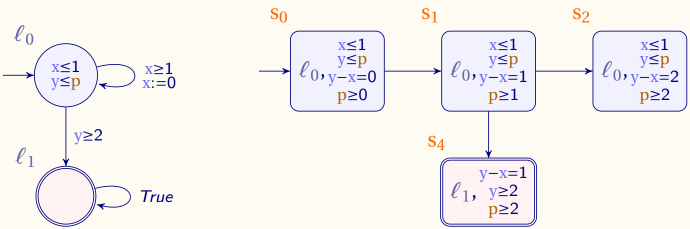
- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at s_0
- Exploration ends with cumulative pruning





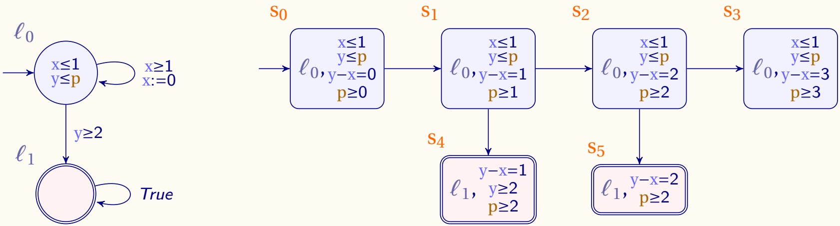
- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at s_5
- Exploration ends with cumulative pruning

Layering Strategy

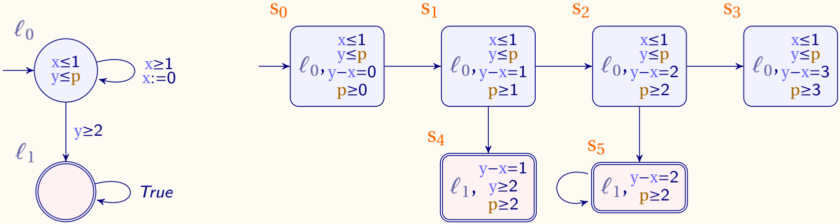


- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at s_5
- Exploration ends with cumulative pruning

Layering Strategy

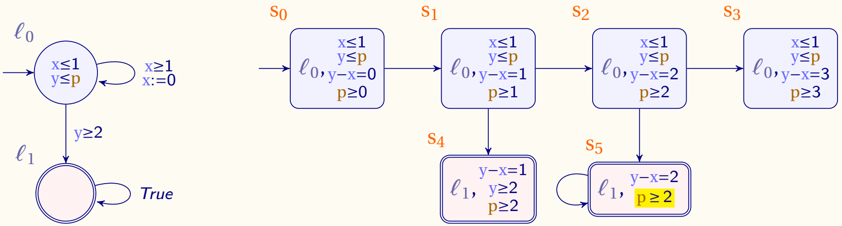


- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at s_5
- Exploration ends with cumulative pruning



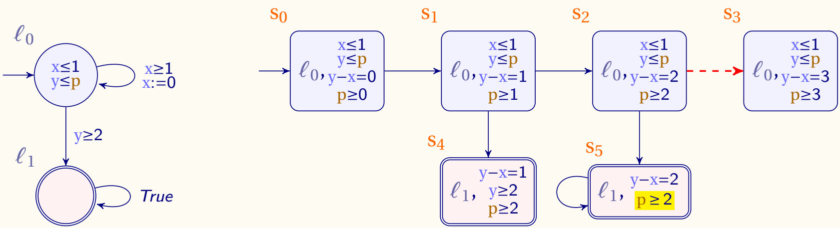
- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at s_5
- Exploration ends with cumulative pruning

Layering Strategy



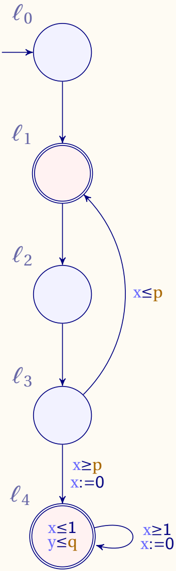
- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at s_5
- Exploration ends with cumulative pruning

Layering Strategy



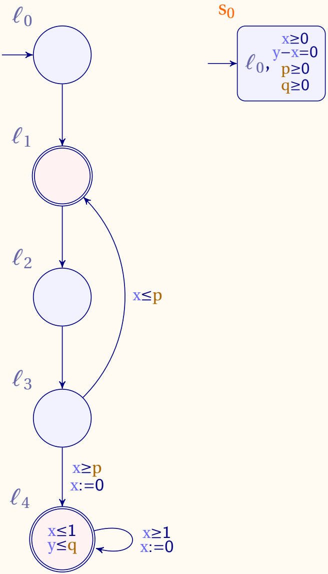
- Exploration by layers, having the same constraints on parameters
- The constraints zones decrease along a path
- Constraint synthesized at s_5
- Exploration ends with cumulative pruning

Look-Ahead Strategy



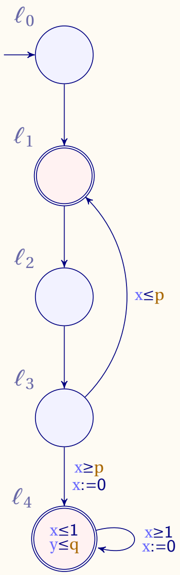
- From s_4, it is possible to engage in an infinite run .... not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \land q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4

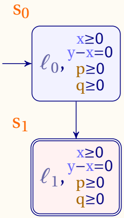
Look-Ahead Strategy



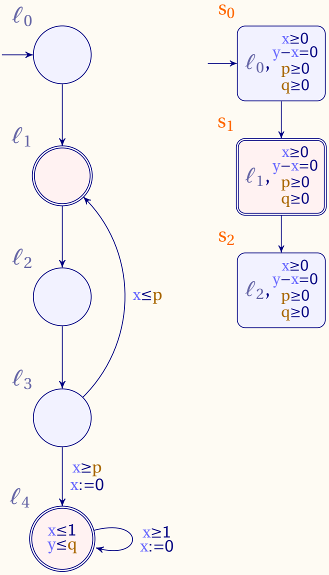
- From s_0, it is possible to engage in an infinite run .... not a lasso!
- Look-ahead strategy: s_0 has an accepting successor "on the stack"
- Constraint $p \geq 0 \wedge q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_0

Look-Ahead Strategy



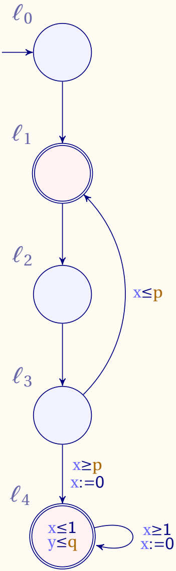


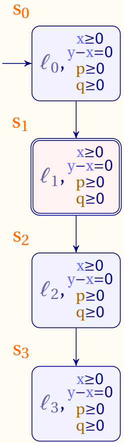
- From s_4, it is possible to engage in an infinite run .... not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \wedge q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4



- From s_4, it is possible to engage in an infinite run .... not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \land q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4

Look-Ahead Strategy





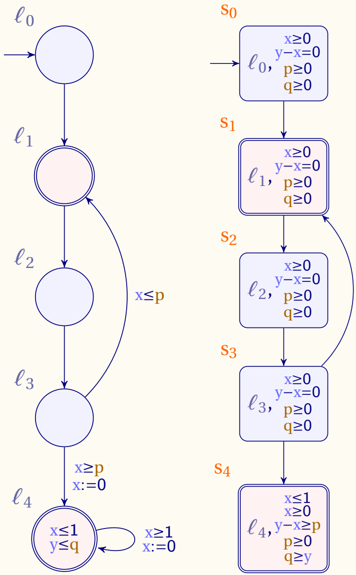
From s_4, it is possible to engage in an infinite run . . . . not a lasso!

Look-ahead strategy: s_9 has an accepting successor "on the stack"

Constraint $p \geq 0 \wedge q \geq 0$ is synthesized immediately

Cumulative pruning prunes away the infinite run starting in s_4

Look-Ahead Strategy

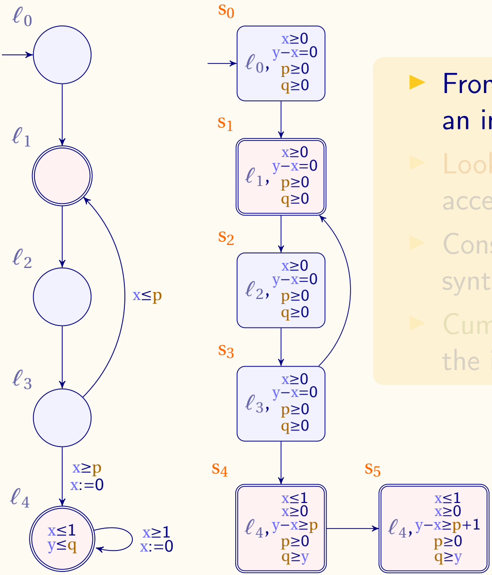


From s_0, it is possible to engage in an infinite run .... not a lasso!

Look-ahead strategy: s_1 has an accepting successor "on the stack"

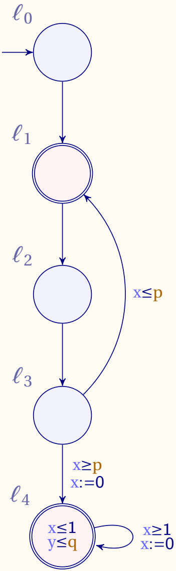
Constraint $p \geq 0 \wedge q \geq 0$ is synthesized immediately

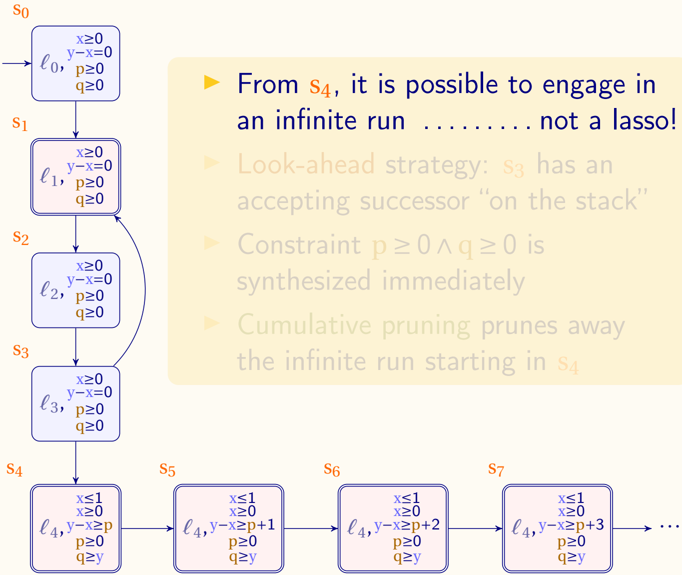
Cumulative pruning prunes away the infinite run starting in s_0



- From s_4, it is possible to engage in an infinite run .... not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \land q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4

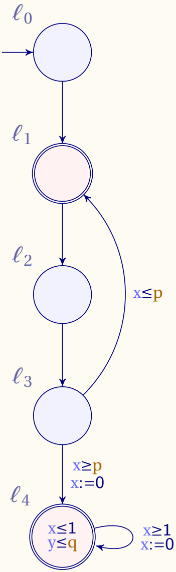
Look-Ahead Strategy

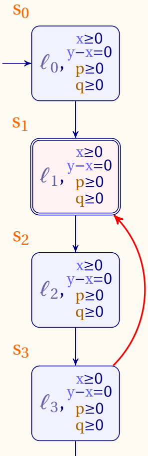




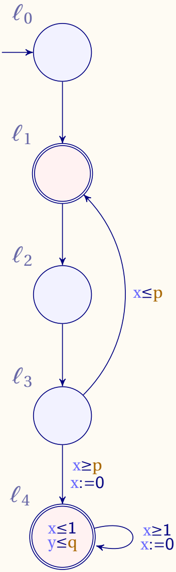
- From s_4, it is possible to engage in an infinite run .... not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \wedge q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4

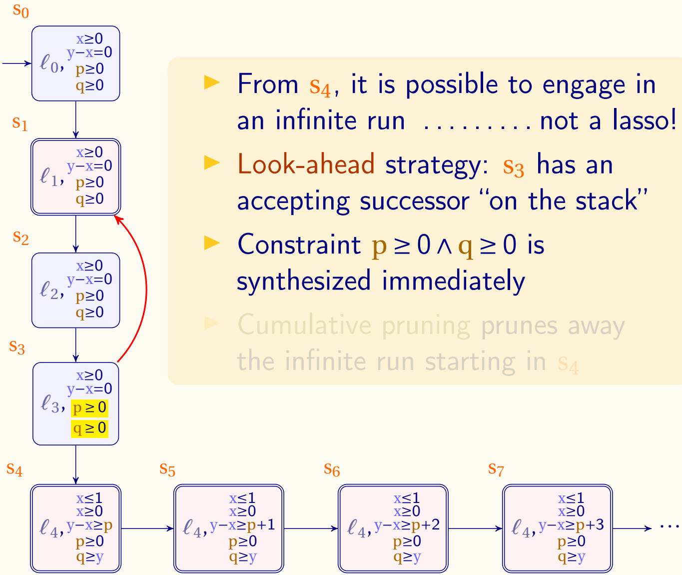
Look-Ahead Strategy



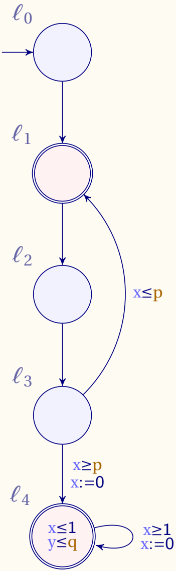


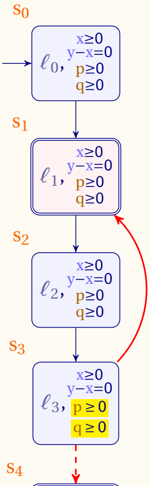
- From s_4, it is possible to engage in an infinite run . . . . not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \land q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4

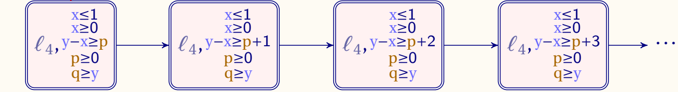




- From s_4, it is possible to engage in an infinite run . . . . not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \land q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4

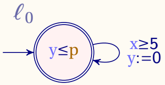






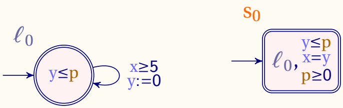
- From s_4, it is possible to engage in an infinite run . . . . not a lasso!
- Look-ahead strategy: s_3 has an accepting successor "on the stack"
- Constraint $p \geq 0 \wedge q \geq 0$ is synthesized immediately
- Cumulative pruning prunes away the infinite run starting in s_4

Subsumption Strategy



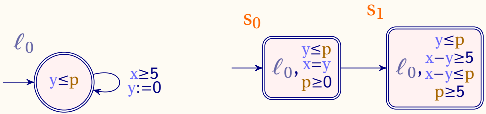
- s_1 is subsumed by s_2, so s_2 simulates s_1
- There must be an infinite accepting run when $p \geq 5$!
  (but there is no "accepting lasso")
- Note: A depth-first strategy would have diverged anyway
- Solution: Bounded Synthesis, Iterative Deepening!

Subsumption Strategy

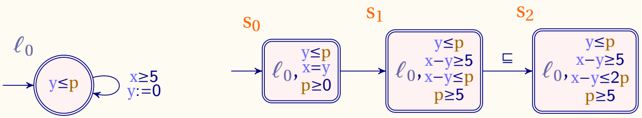


- s_1 is subsumed by s_2, so s_2 simulates s_1
- There must be an infinite accepting run when $p \geq 5$!
  (but there is no "accepting lasso")
- Note: A depth-first strategy would have diverged anyway
- Solution: Bounded Synthesis, Iterative Deepening!

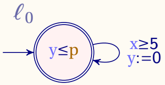
Subsumption Strategy

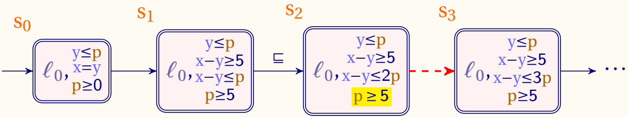


- s_1 is subsumed by s_2, so s_2 simulates s_1
- There must be an infinite accepting run when $p \geq 5$!
  (but there is no "accepting lasso")
- Note: A depth-first strategy would have diverged anyway
- Solution: Bounded Synthesis, Iterative Deepening!

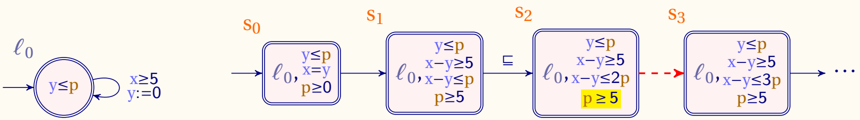


- s_1 is subsumed by s_2, so s_2 simulates s_1
- There must be an infinite accepting run when $p \geq 5$!
  (but there is no "accepting lasso")
- Note: A depth-first strategy would have diverged anyway
- Solution: Bounded Synthesis, Iterative Deepening!





- s_1 is subsumed by s_2, so s_2 simulates s_1
- There must be an infinite accepting run when $p \geq 5$!
  (but there is no "accepting lasso")
- Note: A depth-first strategy would have diverged anyway
- Solution: Bounded Synthesis, Iterative Deepening!



- s_1 is subsumed by s_2, so s_2 simulates s_1
- There must be an infinite accepting run when $p \geq 5$!
  (but there is no "accepting lasso")
- Note: A depth-first strategy would have diverged anyway
- Solution: Bounded Synthesis, Iterative Deepening!

- Apply Nested DFS up to a certain depth; keep increasing this depth

- Strategies:
  - Cumulative Pruning
  - Lookahead
  - Layering
  - Accepting First
  - Subsumption

- Three scenarios:
  - Completely explored state: .... color green to avoid recomputation
  - Cycle found: .... cumulative pruning eliminates all successors
  - Otherwise: .... re-explore in the next iteration of NDFS (cache the successor-computations)

- Apply Nested DFS up to a certain depth; keep increasing this depth

- Strategies:
  - Cumulative Pruning
  - Lookahead
  - Layering
  - Accepting First
  - Subsumption

- Three scenarios:
  - Completely explored state: . . . . color green to avoid recomputation
  - Cycle found: . . . . cumulative pruning eliminates all successors
  - Otherwise: . . . . re-explore in the next iteration of NDFS (cache the successor-computations)

- Apply Nested DFS up to a certain depth; keep increasing this depth

- Strategies:
  - Cumulative Pruning
  - Lookahead
  - Layering
  - Accepting First
  - Subsumption

- Three scenarios:
  - Completely explored state: .... color green to avoid recomputation
  - Cycle found: .... cumulative pruning eliminates all successors
  - Otherwise: .... re-explore in the next iteration of NDFS (cache the successor-computations)

Experimental Evaluation

- Implemented all strategies in IMITATOR[1]
  (open source on github + web-interface[2])
  - Nested DFS (with all strategies)
  - BFS + SCC (Tarjan)
  - BSID: Bounded Synthesis with Iterative Deepening

- Evaluated and compared all strategies on 26 benchmarks[3]

- Applied to a case study: Bounded Retransmission Protocol
  - Synthesized more liberal constraints than in previous work (D'Argenio, Vaandrager)

[1] https://www.imitator.fr/

[2] https://imitator.lipn.univ-paris13.fr/

[3] https://zenodo.org/record/4115919

Experimental Evaluation

- Implemented all strategies in IMITATOR[1]
  (open source on github + web-interface[2])
  - Nested DFS (with all strategies)
  - BFS + SCC (Tarjan)
  - BSID: Bounded Synthesis with Iterative Deepening

- Evaluated and compared all strategies on 26 benchmarks[3]

- Applied to a case study: Bounded Retransmission Protocol
  - Synthesized more liberal constraints than in previous work (D'Argenio, Vaandrager)

[1] https://www.imitator.fr/

[2] https://imitator.lipn.univ-paris13.fr/

[3] https://zenodo.org/record/4115919

Experimental Evaluation

- Implemented all strategies in IMITATOR[1]
  (open source on github + web-interface[2])
  - Nested DFS (with all strategies)
  - BFS + SCC (Tarjan)
  - BSID: Bounded Synthesis with Iterative Deepening

- Evaluated and compared all strategies on 26 benchmarks[3]

- Applied to a case study: Bounded Retransmission Protocol
  - Synthesized more liberal constraints than in previous work (D'Argenio, Vaandrager)

TACAS ★ Artifact ★ AEC ★ Evaluated ★ Consistent ★ Complete ★ Well Documented ★ Easy to Reuse ★

[1] https://www.imitator.fr/

[2] https://imitator.lipn.univ-paris13.fr/

[3] https://zenodo.org/record/4115919

Partial Correctness (too weak)

- Partial Soundness: if the algorithm terminates, then all parameters within the generated constraints lead to accepting runs
- Partial Completeness: if the algorithm terminates, then any parameter that leads to accepting runs is in some generated constraint

Correctness in the limit (ideal)

- Soundness in the limit:
  the algorithm only enumerates constraints that lead to accepting runs
- Completeness in the limit:
  all parameters that lead to accepting runs are eventually enumerated

Complete for lassos (realistic)

- All parameters that lead to an accepting lasso in the Parametric Zone Graph are eventually enumerated

Partial Correctness (too weak)

- Partial Soundness: if the algorithm terminates, then all parameters within the generated constraints lead to accepting runs
- Partial Completeness: if the algorithm terminates, then any parameter that leads to accepting runs is in some generated constraint

Correctness in the limit (ideal)

- Soundness in the limit: the algorithm only enumerates constraints that lead to accepting runs
- Completeness in the limit: all parameters that lead to accepting runs are eventually enumerated

Complete for lassos (realistic)

- All parameters that lead to an accepting lasso in the Parametric Zone Graph are eventually enumerated

Soundness and Completeness

Partial Correctness (too weak)

- Partial Soundness: if the algorithm terminates, then all parameters within the generated constraints lead to accepting runs
- Partial Completeness: if the algorithm terminates, then any parameter that leads to accepting runs is in some generated constraint

Correctness in the limit (ideal)

- Soundness in the limit:
  the algorithm only enumerates constraints that lead to accepting runs
- Completeness in the limit:
  all parameters that lead to accepting runs are eventually enumerated

Complete for lassos (realistic)

- All parameters that lead to an accepting lasso in the Parametric Zone Graph are eventually enumerated

Pathological Cases

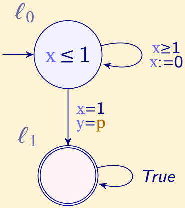
Solution set is not a finite set of convex polyhedra

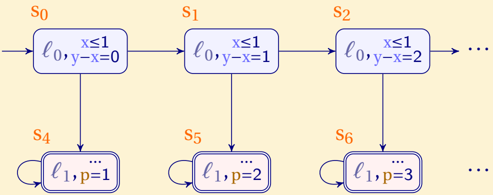
The set of solutions is not a finite union: $p = 1 \vee p = 2 \vee p = 3 \vee \cdots$

An infinite accepting run that is not feasible

The PZG has an infinite accepting run, but its constraint $\bigcap_i (p \geq i) = \emptyset$

Solution set is not a finite set of convex polyhedra



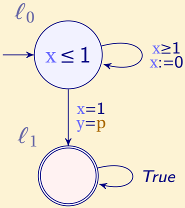


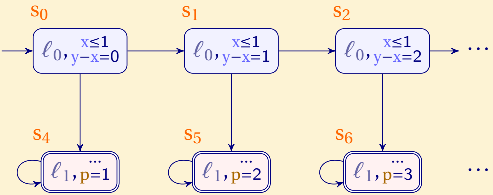
The set of solutions is not a finite union: $p = 1 \lor p = 2 \lor p = 3 \lor \cdots$

An infinite accepting run that is not feasible

The PZG has an infinite accepting run, but its constraint $\bigcap_i (p \geq i) = \emptyset$

Solution set is not a finite set of convex polyhedra





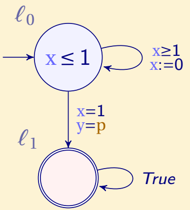
The set of solutions is not a finite union: $p = 1 \vee p = 2 \vee p = 3 \vee \cdots$

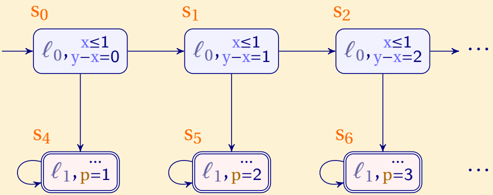
An infinite accepting run that is not feasible

The PZG has an infinite accepting run, but its constraint $\bigcap_i (p \geq i) = \emptyset$

Pathological Cases

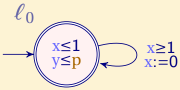
Solution set is not a finite set of convex polyhedra

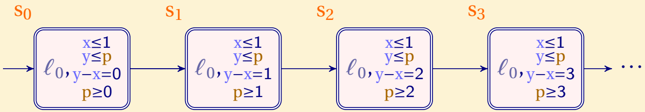




The set of solutions is not a finite union: $p = 1 \vee p = 2 \vee p = 3 \vee \cdots$

An infinite accepting run that is not feasible

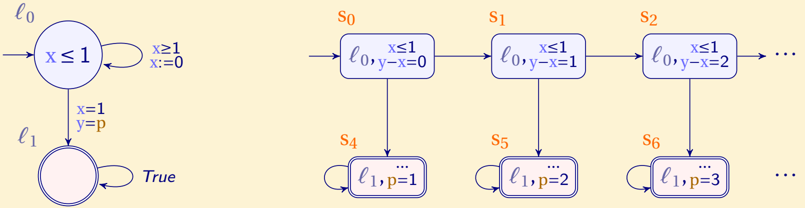




The PZG has an infinite accepting run, but its constraint $\bigcap_i (p \geq i) = \emptyset$

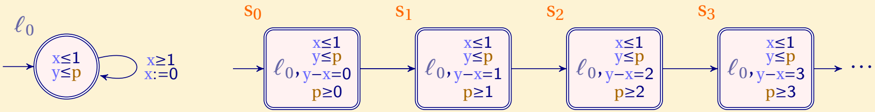
Pathological Cases

Solution set is not a finite set of convex polyhedra



The set of solutions is not a finite union: $p = 1 \lor p = 2 \lor p = 3 \lor \cdots$

An infinite accepting run that is not feasible



The PZG has an infinite accepting run, but its constraint $\bigcap_i (p \geq i) = \emptyset$

Summary of the Results

Algorithm	terminates	partially sound	partially complete	sound in the limit	complete in limit	complete for lassos	$\mathscr{A}_1$	$\mathscr{A}_2$	$\mathscr{A}_3$	$\mathscr{A}_4$	$\mathscr{A}_5$	$\mathscr{A}_6$
NDFS + strategies	×	✓	✓	✓	×	×	✓	×	×	(✓)	(L)	×
BFS + SCC	×	✓	✓	✓	×	✓	✓	×	×	(✓)	(✓)	×
Bound + deepening	×	✓	✓	✓	×	✓	✓	✓	×	(✓)	✓	×
Bounded (fixed n)	✓	✓	×	✓	×	×						
Naïve enumQ	××	(✓)	(✓)	✓	✓	✓						

Naïve enumeration:

- Enumerate all rational parameter values .... (Cantor)
- Check the resulting Timed Automaton for cycles .... (Alur & Dill)

Sender

- requests to transmit a large data package (size N=2)
- transmits data, with alternating bit and indications (lossy channel)
- retransmits messages not acknowledged (bound MAX=2)

Receiver

- reports the received packets to the other user
- sends acknowledgements back through a lossy channel

Timing parameters

TD	time delay of the communication channels
TS	time between subsequent transmissions by the sender
TR	time between subsequent acknowledgements by the receiver
SYNC	the waiting time in case sender and receiver get out of sync

Sender

- requests to transmit a large data package (size N=2)
- transmits data, with alternating bit and indications (lossy channel)
- retransmits messages not acknowledged (bound MAX=2)

Receiver

- reports the received packets to the other user
- sends acknowledgements back through a lossy channel

Timing parameters

TD	time delay of the communication channels
TS	time between subsequent transmissions by the sender
TR	time between subsequent acknowledgements by the receiver
SYNC	the waiting time in case sender and receiver get out of sync

The channels will not be used simultaneously

Exact constraint (2 s): $TS > 2 * TD$ (R1)

The receiver starts each session with a frame with “first-bit” indication

- Add (R1) to the initial constraint of the model
- Exact constraint (R2) (1 s):
  $SYNC + TS >= TR + TD \ \& \ TS > 2 * TD \ \& \ TR > 4 * TS + 3 * TD$

  More liberal (proved with Z3) than earlier one:
  $SYNC >= TR \ \& \ TS > 2 * TD \ \& \ TR > 2 * MAX * TS + 3 * TD$

  Also confirmed up to MAX=20, but we cannot handle a parametric MAX

BRP: reachability properties

The channels will not be used simultaneously

Exact constraint (2 s): $TS > 2 * TD$ (R1)

The receiver starts each session with a frame with "first-bit" indication

- Add (R1) to the initial constraint of the model
- Exact constraint (R2) (1 s):
  $SYNC + TS >= TR + TD \ \& \ TS > 2 * TD \ \& \ TR > 4 * TS + 3 * TD$

  More liberal (proved with Z3) than earlier one:
  $SYNC >= TR \ \& \ TS > 2 * TD \ \& \ TR > 2 * MAX * TS + 3 * TD$

  Also confirmed up to MAX=20, but we cannot handle a parametric MAX

The channels will not be used simultaneously

Exact constraint (2 s): $TS > 2 * TD$ (R1)

The receiver starts each session with a frame with "first-bit" indication

- Add (R1) to the initial constraint of the model
- Exact constraint (R2) (1 s):
  $SYNC + TS >= TR + TD \ \& \ TS > 2 * TD \ \& \ TR > 4 * TS + 3 * TD$

  More liberal (proved with Z3) than earlier one:
  $SYNC >= TR \ \& \ TS > 2 * TD \ \& \ TR > 2 * MAX * TS + 3 * TD$

  Also confirmed up to MAX=20, but we cannot handle a parametric MAX

- Make the failureR location an accepting cycle
- Infinite zone graph → use BSID with step 5 and depth limit 25

- Constraint (6 s): $4 * TS + 3 * TD >= TR \ \& \ TS > 2 * TD$
  OR $TR + TD > SYNC + TS \ \& \ TS > 2 * TD$
- Complement of property (R2), thus exact constraint

- Make the failureR location an accepting cycle
- Infinite zone graph → use BSID with step 5 and depth limit 25

- Constraint (6 s): $4 * TS + 3 * TD >= TR \ \& \ TS > 2 * TD$
  OR $TR + TD > SYNC + TS \ \& \ TS > 2 * TD$
- Complement of property (R2), thus exact constraint

- Spot generates a Büchi automaton for the negation of the formula
- Add it as a monitor, synchronising with the sender
- Add previous results to the initial constraint

GF S_in

Exact constraint (1 s): False — with inclusion ⚠
⇒ no accepting cycle
⇒ **GF** S_in holds

Exact Constraints

G (S_in ⇒ **F** (S_ok ∨ S_nok)) : (R1) — search of a counter-example (0.04s)

G (S_in ⇒ **F** (S_ok ∨ S_nok ∨ S_dk)) holds — with inclusion (16s)

BRP: LTL properties

- Spot generates a Büchi automaton for the negation of the formula
- Add it as a monitor, synchronising with the sender
- Add previous results to the initial constraint

GF S_in

Exact constraint (1 s): False — with inclusion ⚠
⇒ no accepting cycle
⇒ **GF** S_in holds

Exact Constraints

G (S_in ⇒ **F** (S_ok ∨ S_nok)) : (R1) — search of a counter-example (0.04s)

G (S_in ⇒ **F** (S_ok ∨ S_nok ∨ S_dk)) holds — with inclusion (16s)

- Spot generates a Büchi automaton for the negation of the formula
- Add it as a monitor, synchronising with the sender
- Add previous results to the initial constraint

GF S_in

Exact constraint (1 s): False — with inclusion ⚠
⇒ no accepting cycle
⇒ **GF** S_in holds

Exact Constraints

G (S_in ⇒ **F** (S_ok ∨ S_nok)) : (R1) — search of a counter-example (0.04s)

G (S_in ⇒ **F** (S_ok ∨ S_nok ∨ S_dk)) holds — with inclusion (16s)

Conclusion and Perspectives

- Designed different and complementary Parametric Zone Graph exploration strategies
- Implemented them within IMITATOR
- Experimented on benchmarks
- Revisited the Bounded Retransmission Protocol analysis

- Improve the states merging approach
- Handle more cases with automaton pre-analysis

Conclusion and Perspectives

- Designed different and complementary Parametric Zone Graph exploration strategies
- Implemented them within IMITATOR
- Experimented on benchmarks
- Revisited the Bounded Retransmission Protocol analysis

- Improve the states merging approach
- Handle more cases with automaton pre-analysis