PRUNING NESTED-DFS FOR PARAMETRIC TIMED AUTOMATA

LAURE PETRUCCI & JACO VAN DE POL

CNRS/LIPN, PARIS 13

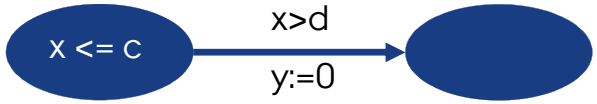
DEPT. OF CS, AARHUS

PARAMETRIC TIMED AUTOMATA

ALUR, HENZINGER, VARDI [STOC 1993]

Design of real-time systems

- Locations, transitions
- Clocks
  - Guards
  - Invariants
  - Resets
- Parameters



Networks of PTA (as in Imitator)

- Communicating automata
- Discrete variables
- Urgent locations

Analysis and Synthesis

- **Reachability of locations**
  - For all parameters
  - Synthesise correct parameters
  - Synthesise optimal parameters [TACAS 2019! Bloemen et al.]
- **Safety and Liveness properties (LTL)**
  - Parametric verification
  - Synthesise correct parameters
- Note: everything is undecidable...

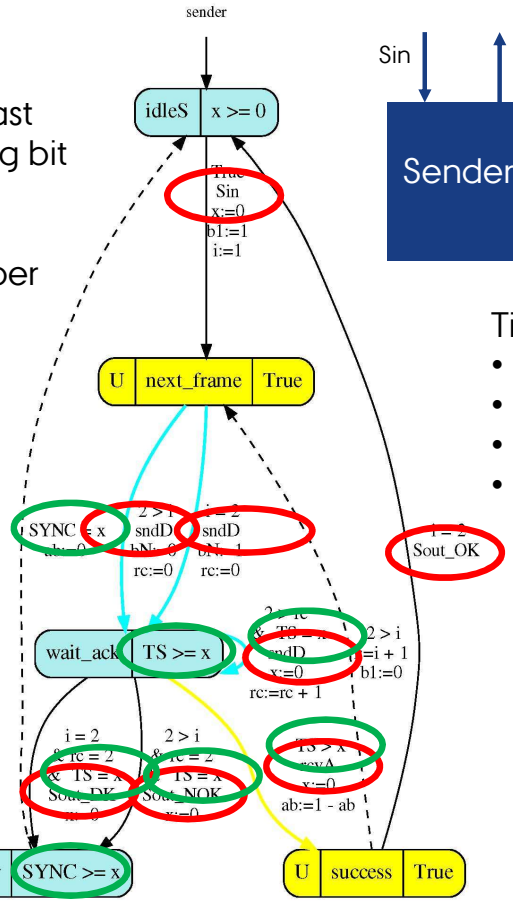
BOUNDED RETRANSMISSION PROTOCOL

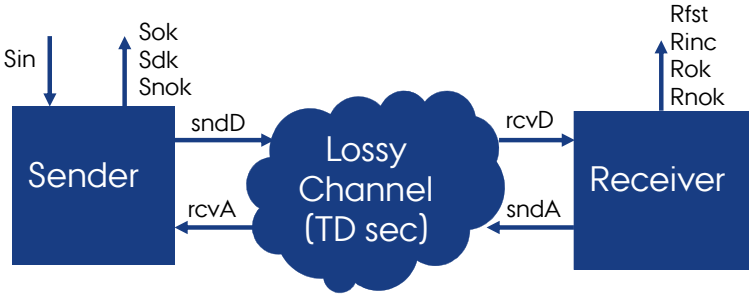
PEDRO D'ARGENIO, JOOST-PIETER KATOEN, THEO RUYS, JAN TRETMANS [TACAS 1997]

Bits:
- b1, bN: first/last
- ab: alternating bit

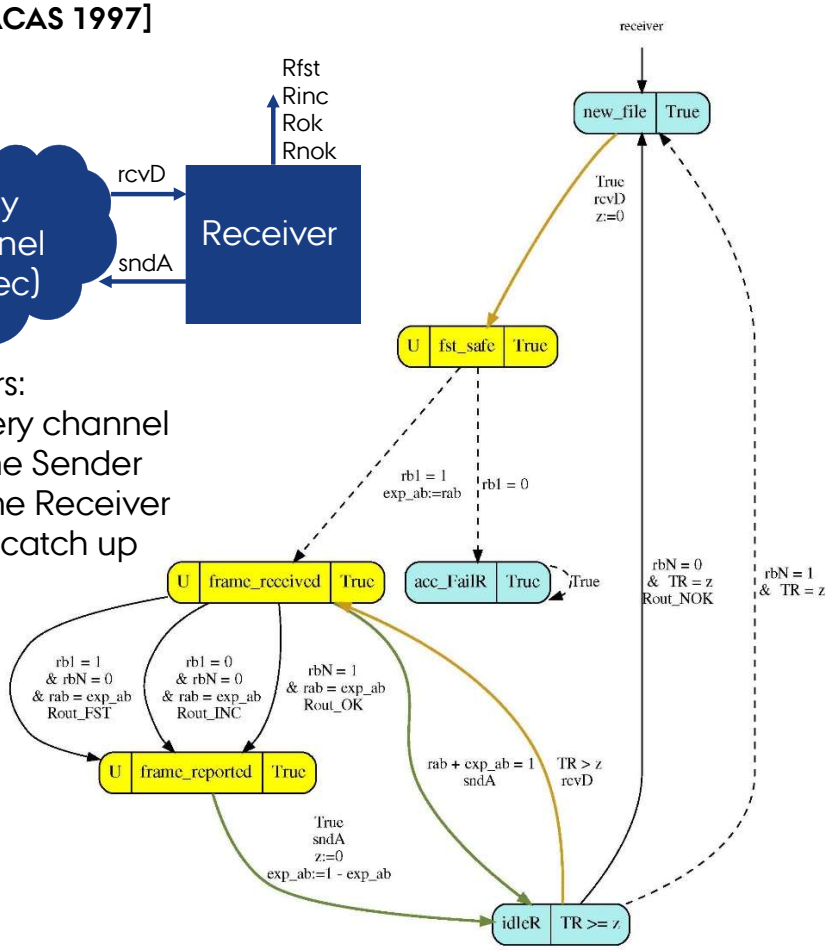
Integers:
- i: frame number
- rc: # retries

Clocks:
- x: sender
- z: receiver





Timing Parameters:
- TD: max delivery channel
- TS: waiting time Sender
- TR: waiting time Receiver
- SYNC: Sender catch up

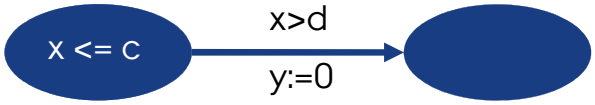


SYMBOLIC ZONE GRAPH

Semantics of Timed Automata:

- Timed Transition System (uncountably infinite)

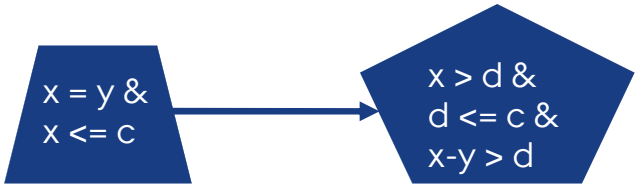
PTA:



Finite abstraction:

- Zone Automaton (extrapolation)
- Efficient DBM representation (x-y < 3)

PZG:



PTA case:

- Parametric Zone Graph (PZG): (t, Z)
- Representation: Polyhedra
- Projection: Parametric Constraint ($Z \downarrow_P$)
- Note: PZG can become infinite

PC: True d<=c

LINEAR-TIME TEMPORAL LOGIC

AMIR PNUELI [1977], COURCOUBETIS, VARDI, WOLPER, YANNAKAKIS [FMSD 1992]

LTL properties:

- Properties on execution paths through the system
- Expressivity: safety and liveness properties
- We restrict to properties over transition labels

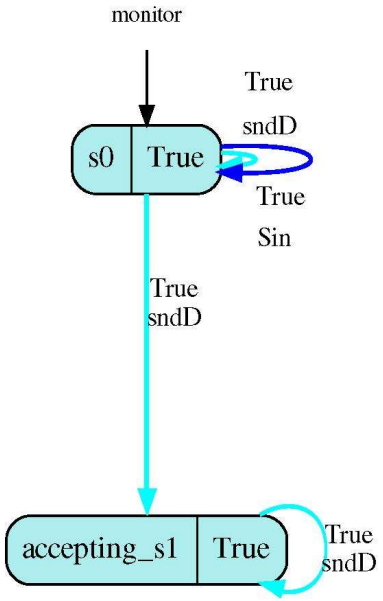
Method:

1. Take the negation of the LTL property
2. Transform it into a Büchi Automaton (in Spot)
3. Add this automaton as a component in Imitator

Correctness:

- Every infinite run through the product is:
  - ✓ An infinite run in the original system
  - ✓ An infinite run through the Büchi automaton
- Accepting runs = counter examples
- No accepting runs = LTL property holds

GF S_in

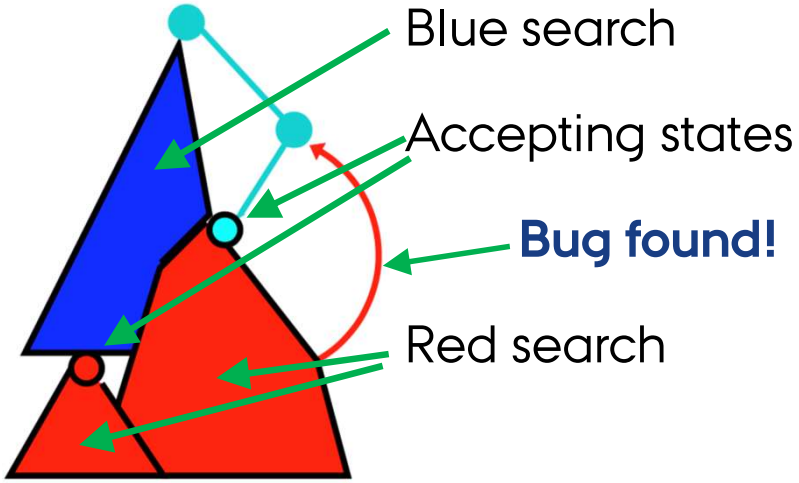


Büchi automaton for the negation

NESTED DEPTH-FIRST SEARCH

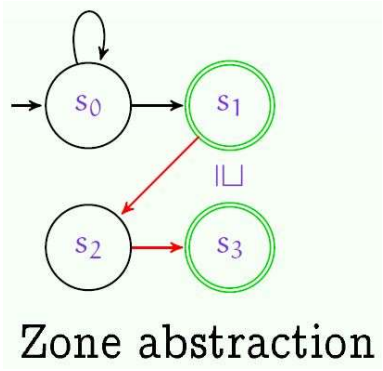
```
dfsblue(s):
    s.color1 := cyan
    for t in s.next do
        if t.color1 == white
            then dfsblue(t)
    if s.accepting
        then dfsred(s)
    s.color1 := blue
```

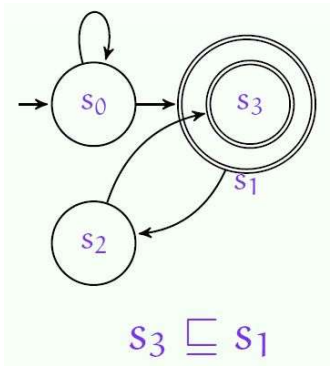
```
dfsred(s):
    s.color2 := red
    for t in s.next do
        if t.color1==cyan
            then CYCLE
        if t.color2 == white
            then dfsred(t)
```



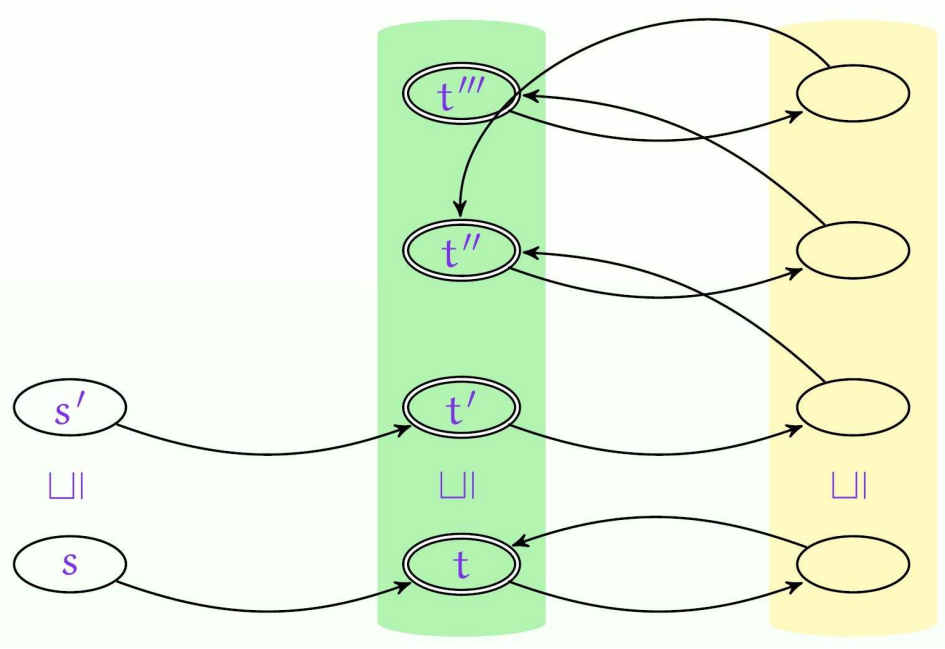
SUBSUMPTION AND LTL FOR TIMED AUTOMATA

ALFONS LAARMAN, MADS OLESEN, ANDREAS DALSGAARD, KIM LARSEN, JVDP [CAV 2013]



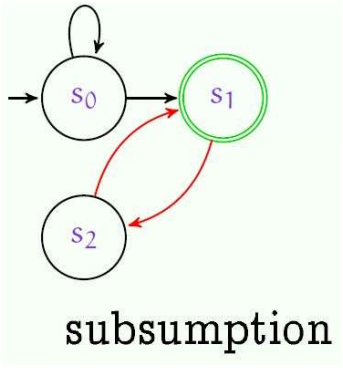


$(l,Z_1) \sqsubseteq (l,Z_2)$ if $Z_1 \subseteq Z_2$



Subsumption is:

- Sound for reachability
- Unsound for liveness:
  - Introduces cycles!



Theorem: an accepting cycle on s can be always be simulated by an accepting cycle on $s' \sqsupseteq s$

PRUNING NDFS WITH SUBSUMPTION

```
dfsblue(s):
  s.color1 := cyan
  for t in s.next do
    if t.color1 == white
      & ∄r. t ⊑ r ∈ Red
        then dfsblue(t)
  if s.accepting
    then dfsred(s)
  s.color1 := blue
```

```
dfsred(s):
  s.color2 := red
  for t in s.next do
    if ∃r. t ⊒ r ∈ Cyan
      then CYCLE
    if ∄r. t ⊑ r ∈ Red
      & t ↓p = s ↓p
      then dfsred(t)
```

Notes:

- If in the red search we encounter a state that **subsumes a cyan** state, then we can already report **an accepting cycle**
- If we encounter a state that is **subsumed by a red state**, we can backtrack, since we would **not find a new cycle**
- We can restrict the red search to **the same layer,** since parameters can **never increase again**

OPPORTUNITIES FOR PRUNING NESTED-DFS

BEZDEK, BENES, BARNAT, CERNÁ [SEFM 2016], GIA NGUYEN, LAURE PETRUCCI, JVDP [ICECCS 2018]

Prune using the collected constraints [collecting]

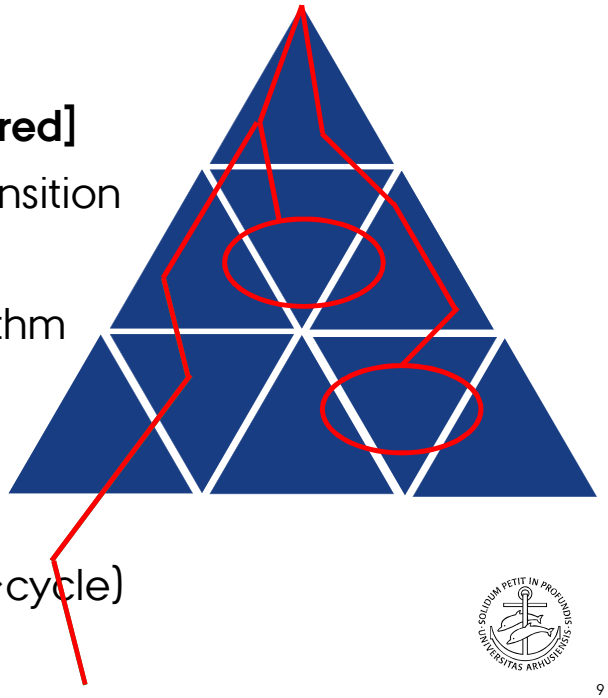
- Assume: so far we found parametric constraints C
- Assume: current state's parametric constraint s is subsumed by C
- ➔ search from s will not contribute to C

Prune or prioritize based on decreasing parametric constraint [layered]

- Assume: parametric constraint strictly decreases along some transition
- ➔ this transition cannot be on a cycle: abort the red search
- ➔ safe to postpone this transition in blue search: layering algorithm

Prune based on subsumption by previous states [subsumption]

- ➔ prune blue search on states that are subsumed by red states
- ➔ prune red search on states that subsume cyan states (spiral→cycle)



COLLECTING AND LAYERED NDFS

```
dfsblue(s):
if ¬s ↓p ⊑ Constr
    s.color1 := cyan
    for t in s.next do
        if t ↓p⊏ s ↓p
            then Pending += t
        else if t.color1 == white
                & ∄r. t ⊑ r ∈ Red
        then dfsblue(t)
    if s.accepting
        then dfsred(s)
    s.color1 := blue
```

```
dfsred(s):
    s.color2 := red
    for t in s.next do
        if ∃r. t ⊒ r ∈ Cyan
            then Constr += t ↓p
        if ∄r. t ⊑ r ∈ Red
          & t ↓p= s ↓p
          then dfsred(t)
```

Main loop:

```
while s from Pending:
    dfsblue(s)
```

Notes:

- We collect **all constraints** that lead to an accepting cycle
- We can **prune states** contained in the constraint, since they **cannot contribute** to the constraint
- Heuristic: all states in the **next parametric layer** can be safely postponed in the **pending list**

OTHER SEARCH STRATEGIES

HERBRETEAU, SRIVATHSAN, TRAN, WALUKIEWICZ [FSTTCS 2016], ÉTIENNE ANDRÉ, GIA NGUYEN, LAURE PETRUCCI [ICECCS 2017]

Search strategy matters for effective subsumption

- BFS tends to find "large" zones earlier
  - Priority queue for frontier of next states
- For NDFS:
  - at least reorder successor states
  - for layered NDFS: reorder the Pending set

Abstraction & Refinement

- Search accepting cycles in abstract PZG
- No cycles: LTL formula holds
- Cycle found? Refine search (per SCC)

IMITATOR BENCHMARK (ICECCS 2018)

				EC Algorithms			StateSpace Algorithms		
Benchmark	#	#	#	ECcex			ECcycles		StateSpace
Models	*X*	*P*	*L*	**NDFS (s)**	**NDFSsub (s)**	**LayerNDFSsub (s)**	**LayerCollectNDFSsub (s)**	**#Zones**	**BFS PRIOR incl (s)**
BRP	7	2	22	0.231	0.237	0.035	0.043	4	176.535
coffee	2	3	4	**TO**	**TO**	0.008	**TO**	2	0.006
critical-region	2	2	20	**TO**	**TO**	**TO**	**TO**	0	**TO**
F3	3	0	15	0.026	0.026	0.008	0.003	1	0.255
F4	4	1	23	**TO**	**TO**	0.007	0.006	1	**TO**
FDDI4	13	2	34	0.305	0.235	1.260	7.004	136	1.319
FischerAHV93	2	4	13	0.010	0.009	0.013	0.013	1	0.040
flipflop	5	2	52	0.010	0.010	0.010	0.012	1	0.015
fmtv1A1-v2	3	3	15	0.060	0.057	46.723	68.223	29	14.040
fmtv1A3-v2	3	3	15	0.063	0.062	302.061	1129.284	67	215.020
JLR13	2	2	2	**TO**	**TO**	**TO**	**TO**	0	**TO**
lynch	2	1	18	0.007	0.007	0.010	0.010	5	0.016
lynch5	5	1	45	0.012	0.012	0.016	0.019	9	3.126
Pipeline-KP12-2-3	4	6	14	0.510	0.508	7.738	492.995	369	**TO**
Pipeline-KP12-2-5	4	6	19	0.791	0.787	**TO**	**TO**	0	**TO**
Pipeline-KP12-3-3	5	6	19	1.960	1.962	**TO**	**TO**	0	**TO**
RCP	6	5	48	0.024	0.013	0.023	0.034	7	10.095
Sched2.50.0	6	2	17	0.011	0.011	0.539	4.168	71	1.940
Sched2.50.2	6	2	17	0.011	0.011	**TO**	**TO**	0	**TO**
Sched2.100.0	6	2	17	0.008	0.008	0.417	3.769	31	2.425
Sched2.100.2	6	2	17	0.008	0.008	**TO**	**TO**	0	**TO**
simop	8	2	46	**TO**	**TO**	**TO**	**TO**	0	**TO**
spsmall	11	2	51	0.244	0.053	0.310	36.549	1026	5.650
train-gate	5	9	11	0.016	0.015	0.047	0.268	15	0.018
WFAS	4	2	10	0.023	0.021	0.056	**TO**	13	**TO**

Table 1. Experimental comparison of Nested DFS algorithms

NEW RESULTS ON IMITATOR BENCHMARKS

	NDFS sub	NDFS layer	NDFS collect	Layers + Pruning
Critical	XXX	XXX	XXX	**Solved!!**
F4	XXX	0.007	0.006	**Solved!!**
JLR13	XXX	XXX	XXX	**Solved!!**
Sched2.50.2	0.011	XXX	XXX	XXX

Relatively simple ideas:

- Giving priority to accepting successors
- Checking for self-loops
- Handling "early termination" cases
  - Cyan successor is accepting

RESULTS ON BRP: REACHABILITY

- Imitator (with –incl and –merge) can easily generate constraints for timing parameters
- Imitator cannot handle discrete parameters like “number of retries”, “length of message”
- ➔ sharper bounds than in original paper [d’Argenio, TACAS 1997]

Original constraints: **T1 > 2.TD && SYNC >= TR > 2.MAX.T1 + 3.TD**

Instantiated for MAX=2: **T1 > 2.TD && SYNC >= TR > 4.T1 + 3.TD** **(1)**

Imitator result (MAX=2): **T1 > 2.TD && SYNC + T1 >= TR + TD && TR > 4.T1 + 3.TD** **(2)**

Note: (1) implies (2), but (2) does not imply (1), so Imitator found more solutions

RESULTS ON BRP: REACHABILITY BY LTL

- All old approaches fail
  - NDFS + subsumption /collecting / layering: cannot handle the simplest case

- NDFS + subsumption + dedicated pruning: **finds some constraints**

- NDFS + abstraction refinement: **finds more constraints (maybe all)**
  1. Run NDFS on full subsumption (unsound for counter-examples)
  2. Confirm found counter-examples
  3. Add negation of found constraints to the initial state, and rerun the procedure

- On arbitrary LTL formulas (e.g. **GF** S_in): currently unsuccessful…

CONCLUSION

Herbretau et al.: LTL model checking for TAs is inherently harder than Reachability

The reachability problem for PTAs is already undecidable

What can we expect?

- We have improved search space pruning
- We can still explore more search order heuristics (like layering, priorities, BMC)
- We will further explore Abstraction Refinement, including acceleration techniques

Currently, Bounded Retransmission Protocol as a (modest) challenge


AARHUS
UNIVERSITY