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...
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

Clocks:
- x: sender
- z: receiver

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

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

Clocks:
- x: sender
- z: receiver

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)

Finite abstraction:
- Zone Automaton (extrapolation)
- Efficient DBM representation \((x-y < 3)\)

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

PTA:
- \(x \leq c\)
- \(y := 0\)
- \(x > d\)

PZG:
- \(x = y \& x \leq c\)
- \(x > d \& d \leq c \& x - y > d\)

PC:
- True
- \(d \leq c\)
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

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 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 $s' \sqsubseteq s$
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 $\neg s \downarrow_p \subseteq \text{Constr}$
  s.color1 := cyan
  for t in s.next do
    if $t \downarrow_p \subseteq s \downarrow_p$
      then Pending += t
  else if t.color1 == white
    & $\nexists r. t \subseteq r \in \text{Red}$
      then dfsblue(t)
if s.accepting
  then dfsred(s)
s.color1 := blue

dfsred(s):
  s.color2 := red
  for t in s.next do
    if $\exists r. t \supseteq r \in \text{Cyan}$
      then Constr += $t \downarrow_p$
    if $\nexists r. t \subseteq r \in \text{Red}$
    & $t \downarrow_p = s \downarrow_p$
      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)

<table>
<thead>
<tr>
<th>Benchmark Models</th>
<th># Models</th>
<th>EC-Algorithm</th>
<th>ECC-EX</th>
<th>STATESPACE Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRP</td>
<td>7</td>
<td>NDFS (s)</td>
<td>0.035</td>
<td>NDFS PRIOR (s)</td>
</tr>
<tr>
<td>coffee</td>
<td>2</td>
<td>TO</td>
<td>0.008</td>
<td>TO</td>
</tr>
<tr>
<td>critical-region</td>
<td>2</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>F3</td>
<td>3</td>
<td>0.026</td>
<td>0.068</td>
<td>0.003</td>
</tr>
<tr>
<td>F4</td>
<td>4</td>
<td>TO</td>
<td>TO</td>
<td>0.006</td>
</tr>
<tr>
<td>FDDI4</td>
<td>13</td>
<td>0.010</td>
<td>0.010</td>
<td>7.004</td>
</tr>
<tr>
<td>FischerAHV93</td>
<td>2</td>
<td>0.010</td>
<td>0.013</td>
<td>0.013</td>
</tr>
<tr>
<td>flipflop</td>
<td>5</td>
<td>0.060</td>
<td>0.057</td>
<td>68.223</td>
</tr>
<tr>
<td>mtv1A1-v2</td>
<td>3</td>
<td>0.063</td>
<td>0.062</td>
<td>1129.284</td>
</tr>
<tr>
<td>mtv1A3-v2</td>
<td>3</td>
<td>0.063</td>
<td>0.062</td>
<td>1129.284</td>
</tr>
<tr>
<td>JLR13</td>
<td>2</td>
<td>TO</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>Lynch</td>
<td>2</td>
<td>0.007</td>
<td>0.010</td>
<td>0.010</td>
</tr>
<tr>
<td>Lynch5</td>
<td>5</td>
<td>0.012</td>
<td>0.012</td>
<td>0.019</td>
</tr>
<tr>
<td>Pipeline-KP12-2-3</td>
<td>4</td>
<td>0.540</td>
<td>0.508</td>
<td>492.995</td>
</tr>
<tr>
<td>Pipeline-KP12-2-5</td>
<td>4</td>
<td>0.791</td>
<td>0.787</td>
<td>0.034</td>
</tr>
<tr>
<td>Pipeline-KP12-3-3</td>
<td>5</td>
<td>1.960</td>
<td>1.962</td>
<td>0.034</td>
</tr>
<tr>
<td>RCP</td>
<td>6</td>
<td>0.024</td>
<td>0.023</td>
<td>7</td>
</tr>
<tr>
<td>Sched2.50.0</td>
<td>6</td>
<td>0.011</td>
<td>0.011</td>
<td>4.168</td>
</tr>
<tr>
<td>Sched2.50.2</td>
<td>6</td>
<td>0.011</td>
<td>0.011</td>
<td>0.417</td>
</tr>
<tr>
<td>Sched2.100.0</td>
<td>6</td>
<td>0.008</td>
<td>0.008</td>
<td>3.769</td>
</tr>
<tr>
<td>Sched2.100.2</td>
<td>6</td>
<td>0.008</td>
<td>0.008</td>
<td>3.769</td>
</tr>
<tr>
<td>simop</td>
<td>8</td>
<td>0.244</td>
<td>0.310</td>
<td>36.549</td>
</tr>
<tr>
<td>spsmall</td>
<td>11</td>
<td>0.016</td>
<td>0.047</td>
<td>0.268</td>
</tr>
<tr>
<td>train-gate</td>
<td>5</td>
<td>0.023</td>
<td>0.021</td>
<td>13</td>
</tr>
<tr>
<td>WFAS</td>
<td>4</td>
<td>0.023</td>
<td>0.021</td>
<td>0.056</td>
</tr>
</tbody>
</table>

**Table 1. Experimental comparison of Nested DFS algorithms**
# NEW RESULTS ON IMITATOR BENCHMARKS

<table>
<thead>
<tr>
<th></th>
<th>NDFS sub</th>
<th>NDFS layer</th>
<th>NDFS collect</th>
<th>Layers + Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical</td>
<td>XXX</td>
<td>XXX</td>
<td>XXX</td>
<td>Solved!!</td>
</tr>
<tr>
<td>F4</td>
<td>XXX</td>
<td>0.007</td>
<td>0.006</td>
<td>Solved!!</td>
</tr>
<tr>
<td>JLR13</td>
<td>XXX</td>
<td>XXX</td>
<td>XXX</td>
<td>Solved!!</td>
</tr>
<tr>
<td>Sched2.50.2</td>
<td>0.011</td>
<td>XXX</td>
<td>XXX</td>
<td>XXX</td>
</tr>
</tbody>
</table>

**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”
- \( \rightarrow \) sharper bounds than in original paper [d’Argenio, TACAS 1997]

Original constraints: \( T_1 > 2.TD && SYNC >= TR > 2.MAX.T1 + 3.TD \)

Instantiated for MAX=2: \( T_1 > 2.TD && SYNC >= TR > 4.T1 + 3.TD \) \( (1) \)

Imitator result (MAX=2): \( T_1 > 2.TD && SYNC + T1 >= TR + TD && TR > 4.T1 + 3.TD \) \( (2) \)

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