Improved BDD-based Discrete Analysis of Timed Systems

Truong Khanh Nguyen[1], Jun Sun[2], Yang Liu[1], Jin Song Dong[1] and Yan Liu[1]

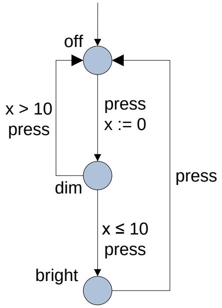
[1]School of Computing
National University of Singapore

[2]Information System Technology and Design,
Singapore University of Technology and Design

FM 2012: 18TH INTERNATIONAL SYMPOSIUM ON FORMAL METHODS

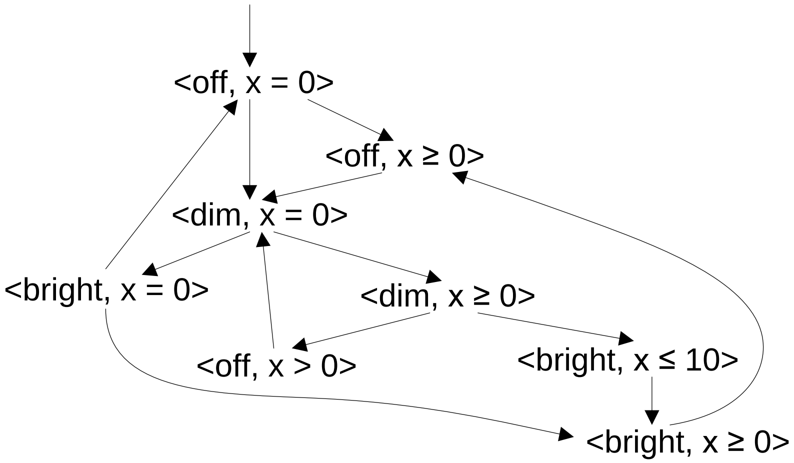
Timed Model Checking

- Timed Automata



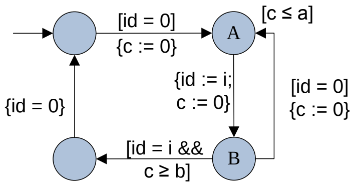
- Zone
  - Set of valuations defined by a clock constraint $\varphi = x \sim c | x - y \sim c | \varphi \wedge \varphi$ where $\sim \in \{<, \leq, =, >, \geq\}$
  - Example: $(x > 3) \wedge (x - y > 1)$
  - Representation: DBM

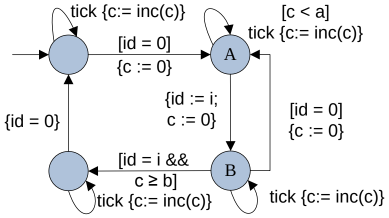
Zone Abstraction: Example



Digitization and the Use of BDD

- ‘Real-time Model Checking is really Simple’.
- Digitization and BDD
- BDD is less sensitive with the number of timed automata but very sensitive with large clock values.





where $inc(c) = return(c \leq M)?(c+1) : c$ and $M = b$

Our Results

bound		32	64	128	256	512	1024	2048	3096
time	PAT	0.5	1.4	5	17	68	293	1297	3018
	Rabbit	5.5	44	570	×	×	×	×	×
memory	PAT	16	21	41	49	104	298	494	519

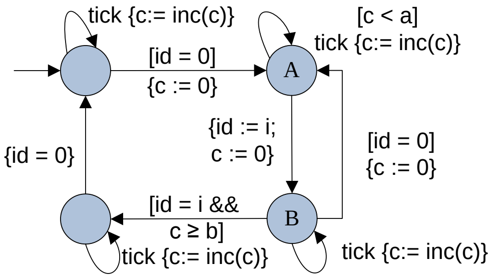
Table : Fischer's protocol with 4 processes

	#proc	8	12	16	24	32	40	50
time	PAT	0.4	1.1	4	20	61	195	531
	UPPAAL	1	200	×	×	×	×	×
	Rabbit	1.6	4.4	12	60	180	473	1142
memory	PAT	17	26	47	136	278	386	757
	UPPAAL	29	629	×	×	×	×	×

Table : Fischer's protocol with time upper-bound 4

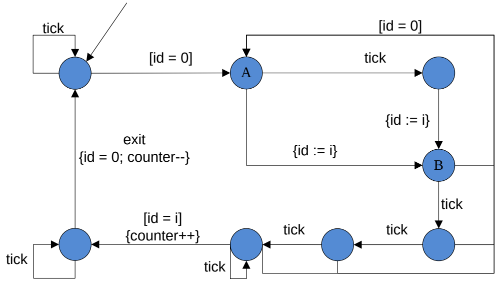
Encoding with Clocks

- Bool variables to encode clocks.
- Encoded similarly to a finite state machine.
- Commplex transition function.
- $a = 1, b = 3$: 2 boolean variables, and 3 boolean variables to encode states, and clock values respectively



Encoding with Ticks

- Generate all tick transitions explicitly and remove clock variables
- Benefit:
  - Simple transition function
  - Use less boolean variables



Clocks vs. Ticks

#proc		4	5	6	7	8
time (s)	without clock variables	0	0	0.1	0.2	0.4
	with clock variables	0.6	15	513	×	×
memory (Mb)	without clock variables	21	22	23	24	26
	with clock variables	32	70	425	×	×

Table : Compare two different approaches of encoding timing constraints

Encoding a Timed Automaton

- Generate a finite automaton without clock variable from timed automaton
- Encoding similarly as finite state machine.
- The encoding of a time automaton is a tuple $\mathcal{B} = (\overrightarrow{V}, \overrightarrow{v}, \mathit{Init}, \mathit{Trans}, \mathit{Out}, \mathit{In}, \mathit{Tick})$
  - $\overrightarrow{V}$: set of unprimed Boolean variables encoding global variables
  - $\overrightarrow{v}$: set of variables encoding local variables
  - *Init*: encoding of the initial state
  - *Out*: encoding of channel out transitions
  - *Int*: encoding of channel in transitions
  - *Tick*: encoding of tick-transitions
  - *Trans*: encoding of other transitions

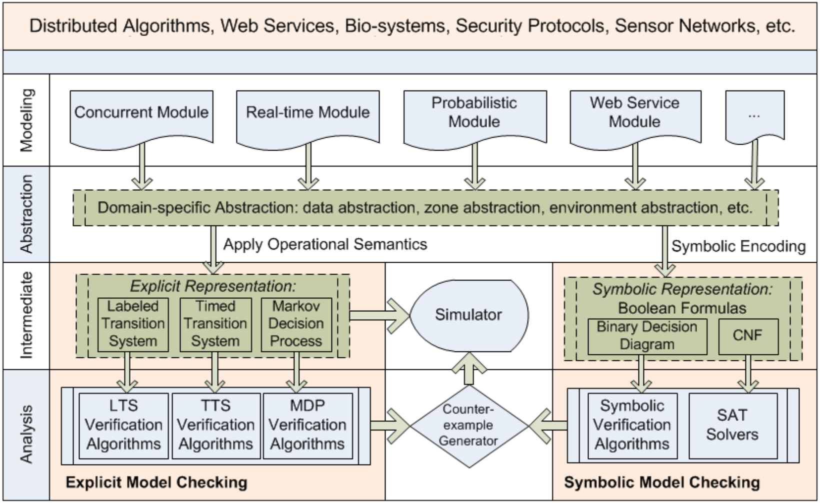
More than a Trick

- Systems are composed hierarchically.
- Compositional functions: Parallel, Interleave, Unconditional Choice, Deadline, Timeout . . .
- Example of Interleave of two BDD machines
  $\mathcal{B}_i = (\overrightarrow{V}, \overrightarrow{v}_i, Init_i, Trans_i, Out_i, In_i, Tick_i)$, $i \in \{0, 1\}$
  - $\overrightarrow{v} = \overrightarrow{v}_0 \cup \overrightarrow{v}_1$;
  - $Init = Init_0 \wedge Init_1$.
  - $Trans = \bigvee_{i\in\{0,1\}}[(Trans_i \wedge \overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i}) \vee (In_i \wedge Out_{1-i})]$ where $(\overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$ denotes that the local variables of $\mathcal{B}_{1-i}$ are unchanged.
  - $In = \bigvee_{i\in\{0,1\}}(In_i \wedge \overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$
  - $Out = \bigvee_{i\in\{0,1\}}(Out_i \wedge \overrightarrow{v}_{1-i} = \overrightarrow{v}'_{1-i})$
  - $Tick = Tick_0 \wedge Tick_1$

Implementation in PAT

- Use CUDD package
- Implemented in PAT framework
- PAT is available at `http://www.patroot.com/`
- 1M lines of C# code, 21 modules with 100+ build in examples
- Used as an educational tool in e.g. York Univ., Univ. of Auckland, NII (Japan), NUS . . .
- 2000+ registered users from 400+ organizations in 52 countries and regions.

Implementation in PAT



Distributed Algorithms, Web Services, Bio-systems, Security Protocols, Sensor Networks, etc.
Modeling
Concurrent Module
Real-time Module
Probabilistic Module
Web Service Module
...
Abstraction
Domain-specific Abstraction: data abstraction, zone abstraction, environment abstraction, etc.
Apply Operational Semantics
Symbolic Encoding
Intermediate
Explicit Representation:
Labeled Transition System
Timed Transition System
Markov Decision Process
Simulator
Symbolic Representation: Boolean Formulas
Binary Decision Diagram
CNF
Analysis
LTS Verification Algorithms
TTS Verification Algorithms
MDP Verification Algorithms
Counter-example Generator
Symbolic Verification Algorithms
SAT Solvers
Explicit Model Checking
Symbolic Model Checking

More Experiments

bound		8/248	12/372	16/497	20/621	26/808	40/1243
time	PAT	5	10	21	35	67	205
	Rabbit	10	32.7	67	90	342	1160
memory	PAT	31	72	126	245	468	518

Table : CSMA/CD with 4 processes

	#proc	8	10	12	14	16	32	64	128
time	PAT	0.3	0.3	0.4	0.6	0.8	5	45	593
	UPPAAL	0.4	3.0	22.9	163	×	×	×	×
	Rabbit	1	1	1.3	1.4	1.5	3	16.1	80
memory	PAT	16	17	18	25	28	73	312	661
	UPPAAL	29	51	292	1894	×	×	×	×

Table : CSMA/CD with time upper-bound 1/4

More Experiments

bound		20	40	80	160	320	640	1280	2560
time	PAT	0.5	1.3	4	9	29	105	428	1853
	Rabbit	2.6	5.3	13.4	54.4	256	1510	×	×
memory	PAT	17	24	31	35	62	122	303	446

Table : Railway control system with 4 stations

	#proc	6	7	8	9	10
time	PAT	1.8	6	16	58	169
	UPPAAL	0.2	1.1	7.9	83.1	×
	Rabbit	53	805	×	×	×
memory	PAT	33	64	170	460	715
	UPPAAL	26	36	111	835	×

Table : Railway control system with time upper-bound 5

More Experiments

Model		Fischer						Railway Control				CSMA/CD			
#proc		**6**	**8**	**10**	**12**	**14**	**16**	**6**	**7**	**8**	**9**	**4**	**6**	**8**	**9**
+Zeno	PAT	5	39	177	599	1653	4345	14	48	157	887	0.2	3	24	106
	UPPAAL	2.3	6711	×	×	×	×	0.4	2.6	24.1	242	0	0.6	662	×
-Zeno	PAT	9	59	269	980	3014	×	21	66	207	1006	0.4	5	55	368

Table : LTL model checking with/without non-Zenoness

Conclusion and Future Work

- Develop a BDD library for timed verification in PAT.
- Applied to 2 different languages.
- Our approach is efficient by not using clock variables.
- Extend our library for probabilistic verification.