CAV 2013

18th July 2013
Saint Petersburg, Russia

PSyHCoS

Parameter Synthesis for Hierarchical Concurrent Real-Time Systems

Étienne André, Yang Liu, Jun Sun, Jin Song Dong, Shang-Wei Lin

Temasek Laboratories
National University of Singapore

Motivation

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"
  - etc.
- Verification for one set of constants does not guarantee the correctness for other values
- Challenges
  - Numerous verifications: is the system correct for any value within $[40;60]$?
  - Optimization: until what value can we increase 10?
  - Robustness: What happens if 50 is implemented with 49.99?

Motivation

- Timed systems are characterized by a set of timing constants
  - "The packet transmission lasts for $50\,ms$"
  - "The sensor reads the value every $10\,s$"
  - etc.
- Verification for one set of constants does not guarantee the correctness for other values
- Challenges
  - Numerous verifications: is the system correct for any value within $[40; 60]$?
  - Optimization: until what value can we increase 10?
  - Robustness: What happens if 50 is implemented with 49.99?
- Parameter synthesis
  - Consider that timing constants are unknown constants (parameters)
  - Find good values for the parameters

Parametric Stateful Timed CSP

- An intuitive formal modeling language
  - English-like description
  - Formal semantics allowing verification
  - Standard constructions of CSP [Hoare, 1978]
  - User-defined data structures (C#-like code)

Parametric Stateful Timed CSP

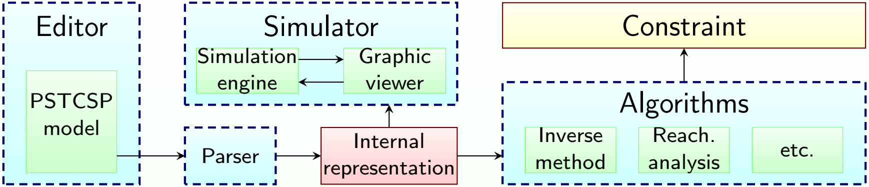
- An intuitive formal modeling language
  - English-like description
  - Formal semantics allowing verification
  - Standard constructions of CSP [Hoare, 1978]
  - User-defined data structures (C#-like code)

- Parameterized timed constructs [André et al., 2012, Sun et al., 2013]
  - `Wait`[u]: waits exactly u time units
  - P `timeout`[u] Q: the first observable event of P shall occur before u time units; otherwise, behaves like Q
  - P `interrupt`[u] Q: behaves like P until u time units; then, like Q
  - P `within`[u]: the first observable event of P shall occur before u time units
  - P `deadline`[u]: P shall terminate before u time units

Algorithms Implemented

- Implementation in PSyHCoS
  - Parameter Synthesis for Hierarchical Concurrent Systems
- Computation of the reachability graph
  - ☺ Interesting for small examples
  - ☹ Often leads to an infinite state space
    - $\rightsquigarrow$ Does not terminate (no synthesis possible)
- Synthesis using the inverse method [André and Soulat, 2013]
  - ☺ Partial exploration of the state space only
  - ☺ Outputs a constraint on the parameters: avoiding numerous verification, allowing optimization and robustness analysis
  - ☺ Often terminate in practice
- And also: classical model checking algorithms
  - LTL / deadlock-checking, etc.

Architecture of PSyHCoS



- Implemented in C# (Microsoft .NET framework)
- Each syntactic construct (with its semantics) implemented in a different class
- Algorithms implemented in a modular way
  $\rightsquigarrow$ Easy reusability and addition of new features

Experiments

Case study	$\lvert U \rvert$	*reachAll* $\lvert S \rvert$	*reachAll* $\lvert T \rvert$	*reachAll* $\lvert X \rvert$	*reachAll* t	*reachAll+* $\lvert S \rvert$	*reachAll+* $\lvert T \rvert$	*reachAll+* $\lvert X \rvert$	*reachAll+* t	*IM* $\lvert S \rvert$	*IM* $\lvert X \rvert$	*IM* t	*IM+* $\lvert S \rvert$	*IM+* $\lvert X \rvert$	*IM+* t
Bridge	4	-	-	-	M.O.	-	-	-	M.O.	2.8k	2	253	2.8k	2	455
Fischer_4	2	-	-	-	M.O.	-	-	-	M.O.	11k	4	41.9	2k	4	8.65
Fischer_5	2	-	-	-	M.O.	-	-	-	M.O.	133k	5	1176	13k	5	84.5
Fischer_6	2	-	-	-	M.O.	-	-	-	M.O.	-	-	M.O.	86k	6	1144
Jobshop	8	14k	20k	2	21.0	12k	17k	2	18.1	1112	2	17.1	877	2	22.8
RCS_5	4	5.6k	7.2k	4	10.5	5.6k	7.2k	4	9.54	5.6k	4	7.83	5.6k	4	16.7
RCS_6	4	34k	43k	4	91.7	34k	43k	4	54.5	34k	4	60.4	34k	4	91.3
TrAHV	6	7.2k	13k	6	14.2	7.2k	13k	6	15.8	227	6	0.555	227	6	0.655

- *reachAll*: computation of the reachability graph
- *IM*: inverse method
- *reachAll+* (resp. *IM+*): version with optimized encoding

Perspectives

- Integration of further state space reduction techniques [André et al., 2013]
- Improvement of the internal representation of constraints relying on the Parma Polyhedra Library [Bagnara et al., 2008]
- Parameterized refinement checking
- Extension to the multi-core setting [Laarman et al., 2013]

Try it!

Available under the GNU-GPL license

http://lipn.univ-paris13.fr/~andre/software/PSyHCoS/

$$\Psi$$

Try it!

Available under the GNU-GPL license

http://lipn.univ-paris13.fr/~andre/software/PSyHCoS/

$$\Psi$$

Demo today at 5pm

FM 2014

19th International Symposium on Formal Methods (FM 2014)

- 12-16, May, 2014
- Singapore
- http://www.comp.nus.edu.sg/~pat/FM2014/

Bibliography

References I

André, É., Fribourg, L., and Soulat, R. (2013).
Merge and conquer: State merging in parametric timed automata.
In *ATVA'13*, Lecture Notes in Computer Science. Springer.

André, É., Liu, Y., Sun, J., and Dong, J. S. (2012).
Parameter synthesis for hierarchical concurrent real-time systems.
In *ICECCS'12*, pages 253–262. IEEE Computer Society.

André, É. and Soulat, R. (2013).
The Inverse Method.
FOCUS Series in Computer Engineering and Information Technology. ISTE Ltd and John Wiley & Sons Inc.

Bagnara, R., Hill, P. M., and Zaffanella, E. (2008).
The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.
Science of Computer Programming, 72(1–2):3–21.

Hoare, C. (1978).
Communicating sequential processes.
Communications of the ACM, 21:666–677.

References II

Laarman, A., Olesen, M. C., Dalsgaard, A. E., Larsen, K. G., and Van De Pol, J. (2013).
Multi-core emptiness checking of timed buchi automata using inclusion abstraction.
In *CAV'13*, volume 8044 of *Lecture Notes in Computer Science*. Springer.

Sun, J., Liu, Y., Dong, J. S., Liu, Y., Shi, L., and André, É. (2013).
Modeling and verifying hierarchical real-time systems using Stateful Timed CSP.
ACM Transactions on Software Engineering and Methodology, 22(1):3.1–3.29.

Configuration encoding

Configuration encoding

- Encoding
  - Process (ID)
  - Value for variables
  - List of clocks
  - Constraint: definition of a normal form
- Example
  - $(\mathtt{Wait}[u_3]_{x_3} \| \mathtt{Wait}[u_5]_{x_3} \| \mathtt{Wait}[u_5]_{x_2}, x_3 \leq x_2)$
  - Encoding:
    - Process: $\mathtt{Wait}[u_3] \| \mathtt{Wait}[u_5] \| \mathtt{Wait}[u_5]$
    - List of clocks: $\{x_3, x_3, x_2\}$
    - Constraint: $x_3 \leq x_2$
- Justification for the list of clocks
  - Distinguishes between $(\mathtt{Wait}[u_3]_{x_3} \| \mathtt{Wait}[u_5]_{x_3} \| \mathtt{Wait}[u_5]_{x_2}, x_3 \leq x_2)$ and $(\mathtt{Wait}[u_3]_{x_2} \| \mathtt{Wait}[u_5]_{x_3} \| \mathtt{Wait}[u_5]_{x_2}, x_3 \leq x_2)$

Configuration encoding: optimization

- Actual equivalence between
  $(\mathtt{Wait}[u_3]_{x_1} \| \mathtt{Wait}[u_5]_{x_2}, x_1 \leq x_2)$ and
  $(\mathtt{Wait}[u_3]_{x_2} \| \mathtt{Wait}[u_5]_{x_1}, x_2 \leq x_1)$

Configuration encoding: optimization

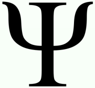
- Actual equivalence between $(\mathtt{Wait}[u_3]_{x_1} \| \mathtt{Wait}[u_5]_{x_2}, x_1 \leq x_2)$ and $(\mathtt{Wait}[u_3]_{x_2} \| \mathtt{Wait}[u_5]_{x_1}, x_2 \leq x_1)$
- Idea $\leadsto$ rename clocks
  - Example: $(\mathtt{Wait}[u_3]_{x_3} \| \mathtt{Wait}[u_5]_{x_3} \| \mathtt{Wait}[u_5]_{x_2}, x_3 \leq x_2)$
    New encoding:
    - Process: $\mathtt{Wait}[u_3] \| \mathtt{Wait}[u_5] \| \mathtt{Wait}[u_5]$
    - List of clocks: $\{x_1, x_1, x_2\}$
    - Constraint: $x_1 \leq x_2$
- This method is time consuming
  - Numerous string and list sorts
  - But often leads to efficient state space reduction

Licensing

Source of the pictures used

Title: Smiley green alien big eyes (aaah)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain

Title: Smiley green alien big eyes (cry)
Author: LadyofHats
Source: https://commons.wikimedia.org/wiki/File:Smiley_green_alien_big_eyes.svg
License: public domain



Title: Psi uc lc
Author: Dcoetzee, F l a n k e r
Source: https://commons.wikimedia.org/wiki/File:Psi_uc_lc.svg
License: public domain

License of this document