Efficient Model Checking of PSL Safety Properties

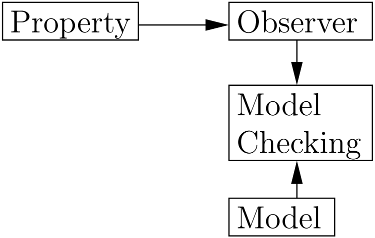
Tuomas Launiainen, Keijo Heljanko, Tommi Junttila

April 3, 2011

Motivation

- Safety properties are an important subclass of temporal properties. Intuitively they are properties whose violation is of the type "something bad happened" as opposed to "something good did not happen".
- Model checking safety properties, especially with BDDs, can be done more efficiently than general properties (Hardin et al. Formal Methods in System Design vol. 18, no. 2, 2001).
- Our approach handles PSL safety properties more efficiently than a general purpose checker.

Overview



- Known way of dealing with safety properties
- Observer is a finite state automaton that accepts counter-examples

PSL

PSL (Property Specification Language) is an industry standard language for specifying temporal properties, i.e. statements about values (or signals) that change over time.

PSL operators

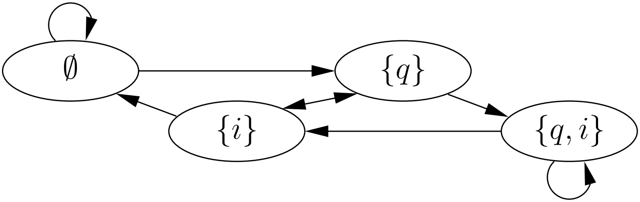
Boolean connectives: $\vee$, $\neg$, etc.					
G	globally	**F**	finally	**X**	next
U	until	**R**	releases		
Sequential regular expressions: catenation, Kleene star, etc.					
$\mapsto$	tail implication	$\diamond\!\!\rightarrow$	tail conjunction		

Transducers

- Transducers are a variant of finite state automata; their state is represented by a set of boolean variables.
- Their transition relation is symbolic.
- Transducers also have initial and final states, but the final states cannot be thought of as accepting.
- In addition to state variables, transducers have input variables. One of the state variables is the output variable, through which accepting is done.
- Transducers can be combined by plugging the output variable of one to an input variable of another.

Transducer example

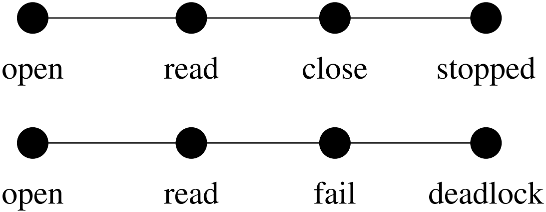
Transducer for the formula **X** `i` (q is the output variable and the only state variable):



All states are initial, $\emptyset$ and $\{i\}$ are final.

Tail implication

- The tail implication $r \mapsto \phi$ states that whenever a match is found for r, ϕ must hold
- Example: $\texttt{open}; \texttt{read}^*; \texttt{close} \mapsto \textbf{X}\ \textbf{G}\ \texttt{stopped}$



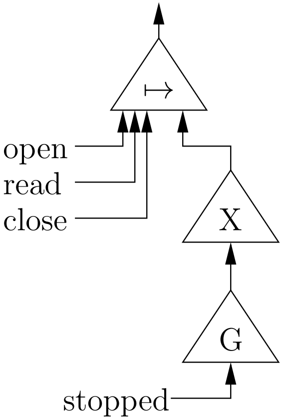
Transducer for tail implication

The tail implication $r \mapsto \phi$ states that whenever a match for r is found, ϕ must hold. A transducer for the formula can be constructed in the following way:

- An automaton is constructed for r.
- Multiple copies are simulated in the transducer.
- When a simulated copy reaches an accepting state, ϕ must hold.
- The states of the automaton for r are the state variables. The variable corresponding to the initial state is the output variable.

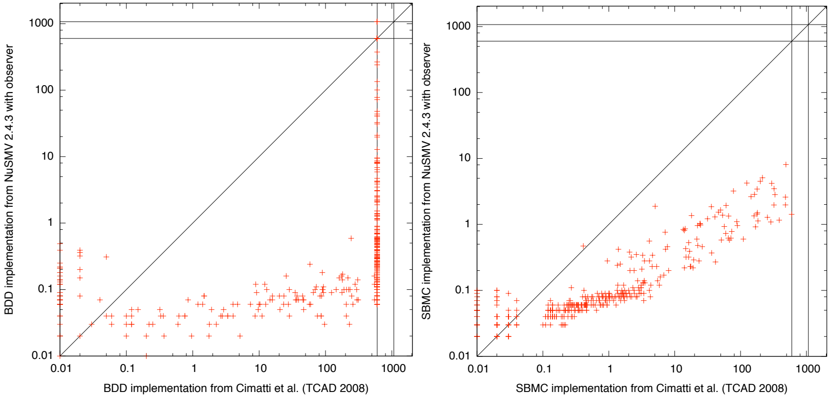
Transducer for tail implication example

Example formula: $\texttt{open}; \texttt{read}^*; \texttt{close} \mapsto \textbf{X}\ \textbf{G}\ \texttt{stopped}$



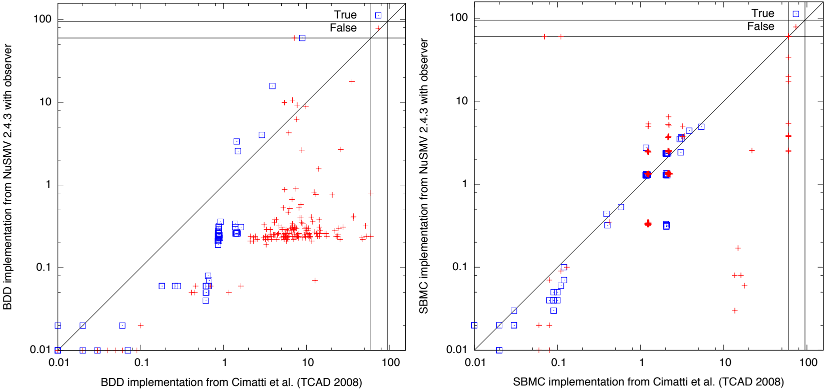
Results

Results with real life PSL safety formulas with random-generated parts:



Results

Results with real life models and random-generated safety formulas:



Conclusion

- Safety properties are an important subclass of specifications that can be dealt with more efficiently than general properties.
- PSL is an industry standard specification language whose safety properties have been specially dealt with only if they are written in the syntactically restricted safety simple subset.
- Our approach is an efficient way to deal with PSL safety properties.
- Our implementation is freely available and works with the open source model checking tool NuSMV.

References

- Cindy Eisner and Dana Fisman. Structural contradictions. In Proc. HVC 2008, volume 5394 of Lecture Notes in Computer Science, pages 164–178. Springer, 2008.
- Alessandro Cimatti, Marco Roveri, and Stefano Tonetta. Symbolic compilation of PSL. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(10):1737–1750, 2008.
- A. Pnueli and A. Zaks. On the merits of temporal testers. 25 Years of Model Checking, pages 172–195, 2008.