Model Checking for Probabilistic Concurrent Systems

- DTMCs and CTL, LTL
- CTMCs and CSL
- MDPs and CTL and LTL
- CTMDPs and CSL
- MAs and CSL
- IMCs and CSL
- Probabilistic Hybrid Systems
LTL Satisfiability Checking Revisited

Jianwen Li¹, Lijun Zhang², Geguang Pu¹, Moshe Vardi³ and Jifeng He¹

¹Software Engineering Institute, East China Normal University
²State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
³Computer Science, Rice University, USA

November 6, 2013
LTL Model Checking [CGP99] has been very useful;
However, people often make mistakes in writing LTL formulas;
There are many works on “property assurances”;
Among them, satisfiability checking is a basic check.
Prior Works on LTL Satisfiability Checking

- Model-checking based
  - SPOT (+ SPIN) [RV07]
  - PANDA + CadenceSMV [RV11]
  - NuSMV-BDD, NuSMV-BMC [CCGR00]

- Temporal Resolution
  - trp++ [HK03]

- Tableau Framework
  - pltl [Sch98]
  - lwb [Sch98]

- Others
  - alaska [DDMR08]
  - tspass [LH10]

- see *Evaluating LTL Satisfiability Solvers* [SD11]
Motivation (1)

1. We follow the automata-theoretic framework
2. $\phi$ is sat? $\iff A_\phi$ is not empty?
3. The tableau construction [GPVW95] is well-known from LTL to Büchi automaton
4. But the generated automaton may be exponential.
Motivation (2)

1. Individual properties are likely to be satisfiable
2. Combined properties are likely to be unsatisfiable
3. Showing satisfiable means finding a model for the property
4. Can we take advantage of that?
5. Yes!
6. Our approach: On-the-fly search + Obligation Set
Consider the following cases:

- \((\ldots)Ub\)
- \((\ldots)Rb\)

There exists the core property set \(\{b\}\) for above formulas!

\(b^\omega\) satisfies both formulas above.

Our idea: Define the *Obligation Set*, which provides an easy way to check satisfiable formulas!
Definition (Obligation Set)

For a formula $\phi$, we define its obligation set, denoted by $Olg(\phi)$, as follows:

1. $Olg(\text{tt}) = \{\emptyset\}$ and $Olg(\text{ff}) = \{\{\text{ff}\}\}$;
2. If $\phi$ is a literal, $Olg(\phi) = \{\{\phi\}\}$;
3. If $\phi = X\psi$, $Olg(\phi) = Olg(\psi)$;
4. If $\phi = \psi_1 \lor \psi_2$, $Olg(\phi) = Olg(\psi_1) \cup Olg(\psi_2)$;
5. If $\phi = \psi_1 \land \psi_2$, $Olg(\phi) = \{O_1 \cup O_2 \mid O_1 \in Olg(\psi_1) \land O_2 \in Olg(\psi_2)\}$;
6. If $\phi = \psi_1 U\psi_2$ or $\psi_1 R\psi_2$, $Olg(\phi) = Olg(\psi_2)$;

For $O \in Olg(\phi)$, we refer to it as an obligation of $\phi$. 
Obligation Set (2)

Example

- \( Olg(aUb) = \{b\} \);
- \( Olg(G(bUc \land dUe)) = \{c, e\} \);
- \( Olg(G(bUc \lor dUe)) = \{c\}, \{e\} \).

Definition (Consistent Obligation)

We say an obligation \( O \) of \( \phi \) is consistent iff for all \( a \in O \) we have that \( \land a \neq ff \).

Theorem (Satisfiability Theorem for Consistent Obligation)

Assume \( O \in Olg(\phi) \) is a consistent obligation. Then, \( O^\omega \models \phi \).
How about if there is no consistent obligation $O \in Olg(\phi)$?

Then we introduce the on-the-fly checking on the transition system of $\phi$.

Why not check on the (generalized-)Büchi automaton?

— We want to use Obligation Sets!
Tagging formulas (1)

- Transition systems are similar to tableau-based GBA
- This makes the checking easier
- But we simplified too much, the transition system does not carry enough information
- We use formula tagging to mark satisfaction of until formulas on the edges of transition systems
Tagging formulas (2)

Given a formula $\phi$, we denote $U(\phi)$ the set of until subformulas of $\phi$. $S_a$ is the set of occurrences of atom $a$, and $\text{right}(\psi)$ is the set of right subformulas of $\psi$. Then:

Definition (Tagging Formula)

Let $a \in AP$ be an atom appearing in $\phi$. Then, the tagging function $F_a : S_a \rightarrow 2^{U(\phi)}$ is defined as: $\psi \in F_a(a_i)$ iff $a_i$ appears in $\text{right}(\psi)$. We define the tagged formula $\phi_t$ as the formula obtained by replacing $a_i$ by $a_{F_a(a_i)}$ for each $a_i \in S_a$.

Example

Consider $\phi = aU(a \land aU\neg a)$. Let $\psi_u = aU\neg a$, and $S_a = \{a_1, a_2, a_3, a_4\}$. From the definition we know $F_a(a_1) = \emptyset$, $F_a(a_2) = F_a(a_3) = \{\phi\}$, and $F_a(a_4) = \{\phi, \phi_u\}$. 
Definition (Normal Form Expansion)

The *normal form* of an LTL formula \( \phi \), denoted as \( \text{NF}(\phi) \), is:

1. \( \text{NF}(\phi) = \{ \phi \land X(tt) \} \) if \( \phi \neq \text{ff} \) is a propositional formula. If \( \phi \equiv \text{ff} \), we define \( \text{NF}(\text{ff}) = \emptyset \);
2. \( \text{NF}(X\phi) = \{ tt \land X(\psi) \mid \psi \in \text{DF}(\phi) \} \);
3. \( \text{NF}(\phi_1 U \phi_2) = \text{NF}(\phi_2) \cup \text{NF}(\phi_1 \land X(\phi_1 U \phi_2)) \);
4. \( \text{NF}(\phi_1 R \phi_2) = \text{NF}(\phi_1 \land \phi_2) \cup \text{NF}(\phi_2 \land X(\phi_1 R \phi_2)) \);
5. \( \text{NF}(\phi_1 \lor \phi_2) = \text{NF}(\phi_1) \cup \text{NF}(\phi_2) \);
6. \( \text{NF}(\phi_1 \land \phi_2) = \{ (\alpha_1 \land \alpha_2) \land X(\psi_1 \land \psi_2) \mid \forall i = 1, 2. \alpha_i \land X(\psi_i) \in \text{NF}(\phi_i) \} \);

Note: Let \( \phi = \bigvee_{1 \leq i \leq n} \phi_i \) and we define \( \text{DF}(\phi) = \{ \phi_i \mid 1 \leq i \leq n \} \).
Definition (LTL Transition System)

The labelled transition system \( T_\phi \) generated from the formula \( \phi \) is a tuple \( \langle \Sigma, S_\phi, \rightarrow, \phi \rangle \) where \( \phi \) is the initial state, and:

1. the transition relation \( \rightarrow \) is defined by: \( \psi_1 \xrightarrow{\alpha} \psi_2 \) iff there exists \( \alpha \land X(\psi_2) \in NF(\psi_1) \);

2. \( S_\phi \) is the smallest set of formulas such that \( \phi \in S_\phi \), and \( \psi_1 \in S_\phi \) and \( \psi_1 \xrightarrow{\alpha} \psi_2 \) implies \( \psi_2 \in S_\phi \).
Example

- **aUb:**
  1. $NF(aUb) = \{ b \land Xtt, a \land X(aUb) \}$;
  2. $NF(tt) = tt \land X(tt)$.

- **$\phi_1 = G(bUc \land dUe)$:**
  1. $NF(\phi_1) = \{ c \land e \land X\phi_1, b \land e \land X\phi_2, c \land d \land X\phi_3, b \land d \land X\phi_4 \}$: here $\phi_2 = bUc \land \phi_1$, $\phi_3 = dUe \land \phi_1$, and $\phi_4 = bUc \land dUe \land \phi_1$.
  2. $NF(\phi_2) = NF(\phi_3) = NF(\phi_4)$.
Theorem

SAT(\(\phi\)) \iff\text{there exists a SCC } B \text{ of } TS_{\phi} \text{ and a state } \psi \text{ in } B \text{ such that } \phi \text{ can expand to } \psi \text{ and, } L(B) \text{ is a superset of some obligation } O \in Olg(\psi) .

Note: \(L(B)\) denotes the set of literals across the SCC \(B\).
On-the-fly Satisfiability Checking

The whole framework of our new algorithm is as follows:

1. We first tag the formula $\phi$. Then we construct $T_\phi$, where we explore the states in an on-the-fly manner, by performing nested depth-first [CVWY92],

2. Whenever a formula is found, we compute the obligation set. In case that it contains a consistent obligation set, we return true,

3. If a SCC $B$ is reached, $\phi \in B$, and $L(B)$ is a superset of some obligation set $O \in Olg(\phi)$, we return true,

4. If all SCCs are explored, but do not have the property in step 3, we return false.

Tool: Aalta$^1$.

$^1$www.lab205.org/aalta
Platform: SUG@R cluster\(^2\) : 2.83GHz Intel Xeon Harpertown CPUs with 16GB RAM per node; Red Hat 4.1.2

Benchmarks:

1. From [RV07]: more than 100,000 random, 8 pattern, 3 counter formulas;
2. Random conjunction formulas: \(\bigwedge_{1\leq i\leq n} P_i\), where \(P_i\) is a random specification pattern\(^3\) (totally 44 types).

Timeout is set to be 300 seconds.

\(^2\)http://www.rcsg.rice.edu/sharecore/sugar/
\(^3\)http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml
Why random conjunctions?

- To check scaling we need large formulas
- But typical properties are not large
- Thus we propose the new random conjunction of specification patterns
- Corresponds to checking the interaction of properties
Experimental Methods

- Compare *Aalta* to model-checking-based LTL satisfiability solvers;
- Explicit: SPOT [DLP04] + SPIN [Hol03]
- Symbolic: PANDA + CadenceSMV[RV11]
- Compare the solvers’ scalability on large formulas
- Study the impact of heuristic strategies
- Separate the satisfiable and unsatisfiable formulas
Experimental Results (1)

Figure: Experimental results for random formulas with 3 variables.
Experimental Results (2)

Figure: Experimental results for $R(n) = \bigwedge_{i=1}^{n} (GFp_i \lor FGp_{i+1})$. 

$\text{LTL Satisfiability Checking Revisited}$

Lijun Zhang (ISCAS) (Beijing)

November 6, 2013 21 / 32
Figure: Experimental results for random conjunctive formulas.
Experimental Results (4)

Figure: Experimental results for 3-variable random formulas from Aalta with OFOA and OF.
Experimental Results (5)

![Graph showing experimental results for random conjunction formulas from Aalta with OFOA and OF.]

Figure: Experimental results for random conjunction formulas from Aalta with OFOA and OF.
Figure: Experimental results for satisfiable random formulas.
Experimental Results (7)

Figure: Experimental results for unsatisfiable random formulas.
Conclusion

- Pro-SAT heuristic strategies are effective
- What about pro-UNSAT heuristics?
- “Mirror Mirror on The Wall, who is the fastest of them all”?
- More work is needed.
Thanks!


