Model Checking **Lots** of Systems

A. Classen, P. Heymans, P. Schobbens, A. Legay, J. Raskin

ICSE’10

Presented by Julia Rubin

October 24, 2012
Disclaimer

Some slides were borrowed from Patrick Heymans’ SPLC 2012 keynote
Overview

⇒ Model Checking ...
⇒ ... Lots of Systems

⇒ Proposed Approach
⇒ Evaluation
⇒ Comments
Overview

- Model Checking ...
- ... Lots of Systems

- Proposed Approach
- Evaluation
- Comments
1. Model of the System

System → Abstraction → Model
2. Properties to be Checked

⇒ “Don’t crash into a wall.”

¬◊ overlap(vehicle, wall)

Operators of a temporal logic

⇒ “Each request is answered.”

□ (request ⇒ ◊response)
Model Checking

Does M satisfy P?

Model M

Property P

counterexample

the property holds!

verified
Overview

- Model Checking ...
- ... Lots of Systems
- Proposed Approach
- Evaluation
- Comments
Lots of Similar Systems

A Product Line
Product Line Engineering

[Pohl et al., 2005]
Main Goal

Validate properties for the whole product line at once

Not for each product individually
Model Checking for Product Lines

Product Line
Model M

Property P

Does M satisfy P?

the property holds for all products!

counterexample + a list of violating products
Model Checking for Product Lines

Product Line
Model M

Property P

Featured Transition System

Does M satisfy P?

\text{the property holds for all products!}

counterexample + a list of violating products
Model Checking for Product Lines

Product Line
Model M

Property P

Regular safety properties and $\omega$-regular properties

counterexample + a list of violating products

the property holds for all products!
Model Checking for Product Lines

Product Line

Model M

Property P

Model Checking for FTS

Does M satisfy P?

- Yes
- No

the property holds for all products!

counterexample + a list of violating products
Overview

Model Checking ...  
... Lots of Systems

Proposed Approach

1. Featured Transition System (FTS)
2. Regular safety properties and $\omega$-regular properties
3. Model Checking these properties for FTS

Evaluation
Comments
1. FTS by Example

Sells soda

Sells soda and tea

Can cancel purchase

Drinks are free
FTS: Feature Model

Optional feature

Or-Group: Chose one or both

Relationship between features
FTS: Feature Model

- Feature model defines valid products
- This one has 12
FTS: Feature Annotations

⇒ Transitions are annotated with features
FTS: Feature Annotations

Stands for:

- One TS with both tea and soda
- One TS with tea only
- One TS with soda only
FTS: Feature Priorities

If feature \( f \) is chosen, the pay transition is not taken.
FTS: Summary

A Featured Transitions System:
1. Feature Model
2. Transition Systems (TS)
3. Annotations of transitions by features
4. Priorities between transitions

⇒ Classical TS model checking won’t work on FTS!
2. Properties

- LTL Regular Safety properties
- LTL $\omega$-regular properties
Regular Safety Properties

- **Safety**: Holds in every state
  - Every counterexample has a finite prefix
  - Infinite number of finite “bad” prefixes

- **Regular Safety**: The set of “bad” prefixes forms a regular language
  - Accepted by a finite automaton

- Not all safety properties are regular
  - “The number of inserted coins is always at least the number of served drinks”
  - Context free language ($pay^nserve^{n+1}$)
\textit{\(\omega\)-regular Properties}

\(\Rightarrow\) “Bad” prefixes form a \(\omega\)-regular language

- Accepted by an automaton of infinite words (Büchi automaton)
  - state is accepting only if visited infinitely often
- Infinite number of infinite words for which accepting state is visited infinitely often

\(\Rightarrow\) Closed under complementation
Model Checking Regular Safety and $\omega$-regular Properties

- TS
- automaton
- Finite / Büchi
Model Checking Regular Safety and ω-regular Properties

Check reachability in the synchronous product of TS and finite / Büchi automaton
Model Checking Regular Safety and $\omega$-regular Properties for FTS

Check reachability in the synchronous product of TS and finite / Büchi automaton
3. Reachability in FTS

- DFS Exploration of the State Space
- Recording reachable states + products for each state
- Recording reachable states + features that the products must have
Example

return

initial state, reachable by all products
Example

initial state, reachable by all products

Transitions:

1. pay/v

2. reachable by products having v but not f.
Example

initial state, reachable by all products

Transitions:

1. \( \text{pay/v} \) from state 1 to state 2
2. \( \text{change/v} \) from state 2 to state 3

- \( \text{pay/v} \) reachable by products having \( \text{v} \) but not \( \text{f} \).
- \( \text{change/v} \) reachable by products having \( \text{v} \) but not \( \text{f} \).
Example

initial state, reachable by all products

Transitions:

1. reachable by products having \( v \) but not \( f \).
2. reachable by products having \( v \) but not \( f \).
3. reachable by products having \( v \) but not \( f \).
4. reachable by products having \( v \) and \( s \) but not \( f \).
5. reachable by products having \( f \) (or having \( v \)).
6. reachable by products having \( f \) and \( s \) (or having \( v \) and \( s \)).

Model Checking Lots of Systems
Overview

- Model Checking ...
- ... Lots of Systems

- Proposed Approach
  - Featured Transition System (FTS)
  - Regular safety properties and $\omega$-regular properties
  - Model Checking these properties for FTS

- Evaluation

- Comments

Limitations and Future Work
Limitations and Future Work

- Should take the structure of a feature model into account
  - Currently – false positive results for non-existing products

- Should allow labeling each transition with an arbitrary Boolean expression
  - Currently – transition can not be annotated by multiple features

- Should allow modeling the system by multiple transition systems or hierarchical statecharts
  - Currently – limited applicability in real life industrial settings
Evaluation

- Implementation available online!
- Complexity
- Speedup for 6 properties
  - comparing to naïve model checking
  - for one example

---

Table 1: Benchmark results for exhaustive counter example search $ExtMC(FTS, \phi)$.  

<table>
<thead>
<tr>
<th>Formula $\phi$</th>
<th>4 features, 4 products</th>
<th>9 features, 64 products</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.1) $\square \Diamond (\text{start} \land \Box \text{msg} \land (\text{methane} \Rightarrow \Diamond \text{palarm}))$</td>
<td>$\checkmark$</td>
<td>9.389 s 5.563 s 1.69</td>
</tr>
<tr>
<td>$\Rightarrow (\square \Diamond (\text{methane} \Rightarrow \Diamond \text{pumpoff}))$</td>
<td>$\times$</td>
<td>25.741 s 37.663 s 0.68</td>
</tr>
<tr>
<td>(1.3) $\Diamond \Box (\text{start} \land \Box \text{msg}) \Rightarrow \Box \Diamond \text{pumpon} \Rightarrow \Diamond \text{running}$</td>
<td>$\checkmark$</td>
<td>5.084 s 4.308 s 1.18</td>
</tr>
<tr>
<td>(1.4) $\Diamond \Box (\text{msg} \land \Box \text{level}) \Rightarrow \Box \Diamond (\text{lowwater} \Rightarrow \Diamond \text{pumpoff})$</td>
<td>$\checkmark$</td>
<td>4.970 s 4.156 s 1.20</td>
</tr>
<tr>
<td>(1.5) $\Diamond (\text{msg} \land \Box \text{level} \land \text{ready}) \Rightarrow \Box (\text{highwater} \land \Box \neg \text{methane}) \Rightarrow \Diamond \text{pumpon}$</td>
<td>$\checkmark$</td>
<td>5.172 s 4.462 s 1.16</td>
</tr>
<tr>
<td>(1.6) $\Diamond (\text{msg} \land \Box \text{level} \land \text{ready}) \Rightarrow \Box (\text{highwater} \land \Box \neg \text{methane}) \Rightarrow \Diamond \text{pumpon}$</td>
<td>$\times$</td>
<td>5.437 s 4.405 s 1.23</td>
</tr>
</tbody>
</table>
Evaluation

- Implementation available online!
- Complexity
- Speedup for 6 properties
  - comparing to naïve model checking
  - for one example

Table 1: Benchmark results for exhaustive model checking of SetMC(FTS, ϕ).

<table>
<thead>
<tr>
<th>Formula ϕ</th>
<th>Cur.</th>
<th>Our</th>
<th>Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.1)</td>
<td>25.741 s</td>
<td>8.162 s</td>
<td>7.07</td>
</tr>
<tr>
<td>(1.2)</td>
<td>138.970 s</td>
<td>102.716 s</td>
<td>3.34</td>
</tr>
<tr>
<td>(1.3)</td>
<td>5.084 s</td>
<td>5.317 s</td>
<td>0.62</td>
</tr>
<tr>
<td>(1.4)</td>
<td>4.970 s</td>
<td>4.926 s</td>
<td>0.35</td>
</tr>
<tr>
<td>(1.5)</td>
<td>5.172 s</td>
<td>4.462 s</td>
<td>1.16</td>
</tr>
<tr>
<td>(1.6)</td>
<td>5.437 s</td>
<td>4.405 s</td>
<td>1.23</td>
</tr>
</tbody>
</table>
Overview

黛 Model Checking ...
黛 ... Lots of Systems
黛 Proposed Approach
黛 Evaluation
黛 Comments
Comments

⇒ Well written paper
⇒ Cited by 77
⇒ Conclusions: “This paper lays the foundations for scalable modeling and efficient verification of software product lines”

⇒ Are they really first?
Novelty

- Model Checking of Domain Artifacts in Product Line Engineering, ASE 2009
  - less cited (only 47)

- “While similar to FTS, their modeling language
  - does not support priorities between features
  - and uses a non-standard definition of the parallel composition (which adds transitions that were not in the original automata)”
Does One Need Priorities?

Priorities

Model Checking Lots of Systems
Does One Need Priorities?

That’s what SPLC 2012 keynote uses

Part of listed future work

Priorities are not used in the evaluation

Many related work is “dismissed” because of the lack of priorities
Evaluation

Why not building an experiment that compares to other works?

E.g., MTS model checker

- “Modal transition systems were first proposed by Fischbein et al. [18] to model SPL behavior.
- Transitions in an MTS are mandatory or optional. An MTS thus specifies a family of behaviors since optional transitions may or may not be read when executing.
- Similarly to our approach, a single MTS model check allows to verify all possible products at once.
- Yet, MTS lack the notion of feature and priority between transitions.”

Can one claim “substantial gains” when comparing to their own naïve implementation?

What about fundamental contribution?
Thank You!

Model Checking Lots of Systems