Inferring Temporal System Properties

Samuel Huang, joint work with Rance Cleaveland

University of Maryland, College Park

April 28th, 2011

Model Checking Problem

$$\text{Ask:} \quad M \models \phi \quad ?$$

- M is a model
- ϕ is a property/requirement
- $\models$ is a satisfaction relation

Related Problems

Synthesis Problem

Find a suitable M: $\quad \Box \models \phi$

Property extraction problem

Find all suitable ϕ: $\quad M \models \{\Box\}$

Related Problems

Synthesis Problem

$$\text{Find a suitable } M\text{:} \quad \Box \models \phi$$

Property extraction problem

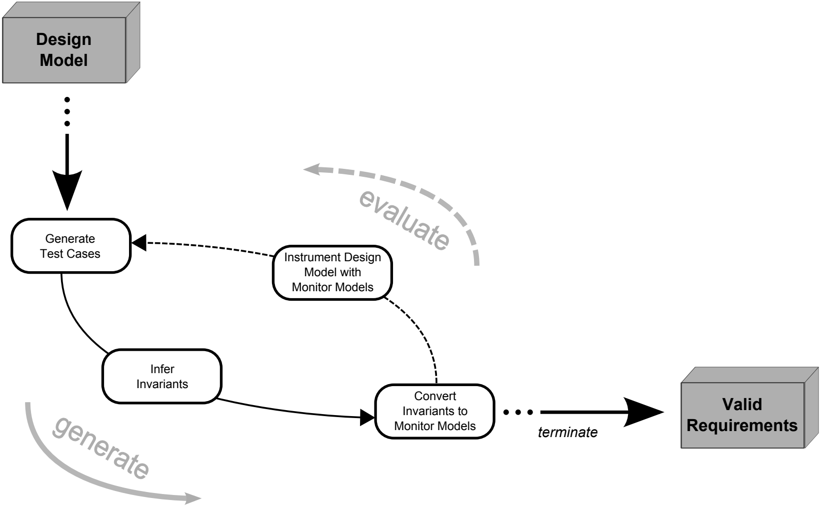
$$\text{Find all suitable } \phi\text{:} \quad M \models \{\Box\}$$

Motivation for Requirements Extraction

- System comprehension
- System reconstruction
  - Incomplete/missing/out-dated documentation
  - "Implicit" (and sometimes unintended) requirements (during construction of system)
- Requirements extraction can serve as a way to estimate high level behavior of a system in terms of the properties that it exhibits.

Automatic Requirement Extraction from Test Cases [ACH+10]

(joint work with Fraunhofer and Robert Bosch)



Observations

- By varying the method by which test cases are generated, we extracted different degrees of requirements
  - Randomized - yielded sparse and lower total number of requirements
  - Structurally guided (MCDC coverage) - more complete overall requirement set
- Iterating requirement extraction process helped lead to refinement of final results

Observations

- By varying the method by which test cases are generated, we extracted different degrees of requirements
  - Randomized - yielded sparse and lower total number of requirements
  - Structurally guided (MCDC coverage) - more complete overall requirement set
- Iterating requirement extraction process helped lead to refinement of final results

Observations

- By varying the method by which test cases are generated, we extracted different degrees of requirements
  - Randomized - yielded sparse and lower total number of requirements
  - Structurally guided (MCDC coverage) - more complete overall requirement set
- Iterating requirement extraction process helped lead to refinement of final results

Observations

- By varying the method by which test cases are generated, we extracted different degrees of requirements
  - Randomized - yielded sparse and lower total number of requirements
  - Structurally guided (MCDC coverage) - more complete overall requirement set
- Iterating requirement extraction process helped lead to refinement of final results

A Caveat, and Moving On

- In this work we assumed the model was known to us, and a test suite was generated to satisfy some coverage criterion on the model. What can be done without knowledge of the model?
- Given a set of a system's executions E, what properties can be discovered of the system that hold "true?"
- Here a "true" property means one with some measure of accuracy over the execution set E, such as satisfying some *support*. [AIS93]
- The properties discovered should be in some understandable and usable format, such as a temporal logic.

Learning from executions

- Treat set E as a sequence database, and incorporate sequential pattern mining. [AS95, YHA03, Moe07]
- Can mine patterns of the form

$$A \rightarrow B \rightarrow C \rightarrow \ldots$$

- Which can be rewritten as

$$\mathrm{F}\,(A \rightarrow \mathrm{XF}\,(B \rightarrow \mathrm{XF}\,(\ldots)))$$

Support Metric

- Sequential pattern mining algorithms do not only return patterns that are correct 100% of the time. Typically they require a support parameter, which specifies how often rules must be correct to be considered significant.
- The previous rule is more properly written as

$$P_{=s}\left[\mathrm{F}\left(A \rightarrow \mathrm{XF}\left(B \rightarrow \mathrm{XF}\left(\ldots\right)\right)\right)\right]$$

  Here, the rule has been written in probabilistic temporal logic expressing uncertainty in its occurrence.

Related work

- Recent work [LKL07, LKL08] discovers rules of a software code base (JBoss Application Server) in an effort to uncover underlying program design and identify bugs. Characterization of temporal logic fragments that are covered is unclear.
- BIOCHAM [CFS06] - ad hoc machine learning inference of temporal logic formulae for bio-molecular interaction networks.

Next Steps

- Expand supported fragement of temporal logic as much as possible. How far can we go?
- Different fragments are useful for different application domains
  - Software engineering/program analysis:

$$event \longrightarrow F\neg(\mathrm{power\_stays\_on})$$

  - Metabolic pathways:

$$protA \longrightarrow protB \longrightarrow \neg protC \longrightarrow protD$$

$$protA \longrightarrow protB \dashv protC \longrightarrow protD$$

Thanks!

Christopher Ackermann, Rance Cleaveland, Samuel Huang, Arnab Ray, Charles P. Shelton, and Elizabeth Latronico.
Automatic requirement extraction from test cases.
In *RV*, pages 1–15, 2010.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami.
Mining association rules between sets of items in large databases.
In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM.

Rakesh Agrawal and Ramakrishnan Srikant.
Mining sequential patterns.
In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

Laurence Calzone, François Fages, and Sylvain Soliman.

Biocham: an environment for modeling biological systems and formalizing experimental knowledge.

Bioinformatics, 22:1805–1807, July 2006.

David Lo, S.-C Khoo, and C Liu.

Mining temporal rules from program execution traces.

In *Proc. of Int. Work. on Program Comprehension through Dynamic Analysis*, 2007.

David Lo, Siau-Cheng Khoo, and Chao Liu.

Mining past-time temporal rules from execution traces.

In *Proceedings of the 2008 international workshop on dynamic analysis: held in conjunction with the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2008)*, WODA '08, pages 50–56, New York, NY, USA, 2008. ACM.

Fabian Moerchen.
Unsupervised pattern mining from symbolic temporal data.
SIGKDD Explor. Newsl., 9:41–55, June 2007.

Xifeng Yan, Jiawei Han, and Ramin Afshar.
CloSpan: Mining Closed Sequential Patterns in Large Datasets.
In *In SDM*, pages 166–177, 2003.