Investigating impact of noise on results, as well as scaling up

NSF – Expeditions in Computing

Christopher Ackermann, Rance Cleaveland, Samuel Huang, Arnab Ray,

Success for recovering properties on small examples such as:

Currently using a genetic programming approach to search

Practically, we would only have a sample of streams from

Consider the set of all data streams

This fitness can help guide our search through the hypothesis

Of all possible starting conditions, only some may be observed: draw

What about noise introduced by erroneous

Given a hypothesis property

Simulators could also be used (e.g. BioNetGen [3]).

Model checking as an Oracle

Model checking traditionally answers “does model \( M \models \phi \)?”

Can modify this to ask “does data stream \( d \) satisfy property \( \phi \)?”

Equivalent to verifying if one particular execution trace of \( M \) (path through the program’s state machine) complies with \( \phi \).

Can use existing model checking algorithms/solvers for data stream verification (e.g. NuSMV [4]). Simplicity of data stream structure aids in efficiency of this model checking.

Simulators could also be used (e.g. BioNetGen [3]).

Learning from Data Streams

Given a hypothesis property \( \phi \), we can assign it a fitness (potential) based upon its success in satisfying each of the data streams.

This fitness can help guide our search through the hypothesis space (such as space of all LTL formulas).

Setting

- Data streams are pervasive!
- Many result from outputs of structured processes

Question: Can we reason about a process’s internal structure via reasoning over these outputs?

Early Efforts

- Pilot study [1]: real-time recovery of invariants from execution traces of known programs in the automotive domain.
- Used combination of data mining-based techniques [2] and Instrument-Based Verification (IBV) for discovering rules based on set of test cases (input output pairs) from a Matlab/Simulink model:

Allowed for verification of specifications that could be represented as:

\[ a \land b \land c \land \ldots \rightarrow \alpha \land \beta \land \gamma \land \ldots \]

Incorporated notions of a rule’s support and confidence to select significant and accurate rules. The approach was shown to be robust to noise, and allowed for detection of incorrect implementations/specifications.

Expanding Expressiveness

- Previous work towards mining properties, (e.g. [5]), but all recover restricted classes of properties and are not “complete.”
- Different domains necessitate different classes of interest:
  - Software engineering:
    \[ G(\text{lock} \rightarrow \text{release} \_ \text{lock}) \]
  - Metabolic pathways:
    \[ G(\downarrow \text{protA} \uparrow \text{protB}) \]
- Currently would have to use different techniques/approaches to handle per-domain properties. In more complex domains, multiple techniques would be required to cover the span of all “interesting” properties.
- Would like to mine properties from a larger space of more interesting properties, such as all of LTL or CTL.

Sampling/Handling Noise

- Consider the set of all data streams \( \mathcal{D} \) capable of being emitted from a model \( M \).
- Practically, we would only have a sample of streams from \( \mathcal{D} \) that have been observed.
- Biasing of this sample can lead to interesting situations. How is it biased?
  - Of all possible starting conditions, only some may be observed: draw firm conclusions for only this class of scenarios.
  - What about noise introduced by erroneous/buggy systems? Or an adaptation to the underlying process for a fraction of the streams?

Initial Results

- Currently using a genetic programming approach to search space of possible CTL solutions.
- Success for recovering properties on small examples such as:

\[ a \lor b \]
\[ a \rightarrow F b \]

Investigating impact of noise on results, as well as scaling up to larger applications (e.g. software systems, metabolic pathways) and more complex patterns.

References