A Case Study on Extracting the Characteristics of the Reachable States of a State Machine
Formalizing a Communication Protocol with Inductive Logic Programming

Dung Tuan Ho*, Min Zhang**, Kazuhiro Ogata*

*Japan Advanced Institute of Science and Technology (JAIST)
**East China Normal University (ECNU)
Outline

• Interactive Theorem Proving
• Communication Protocol
• Method & Architecture of Tool Used
• Case Studies
• Conclusion and Future Work
Interactive Theorem Proving

A System \rightarrow A State Machine \: M \equiv < S, I, T >
Interactive Theorem Proving

a System $\rightarrow$ a State Machine $M \triangleq \langle S, I, T \rangle$  
State predicate $p \leftarrow$ a Property
Interactive Theorem Proving

a System $\rightarrow$ a State Machine $M \triangleq < S, I, T>$ $\models$ State predicate $p$ $\Leftarrow$ a Property

$p$ is an invariant?
Interactive Theorem Proving

A System $\rightarrow$ a State Machine $M \triangleq <S, I, T> \quad \models \quad$ State predicate $p \quad \leftarrow \quad$ a Property
Interactive Theorem Proving

A system $M \triangleq \langle S, I, T \rangle \models$ State predicate $p \iff$ a Property

Conjecture a lemma $q$
Interactive Theorem Proving

Issue!!!
How to conjecture such lemma q?
Interactive Theorem Proving

Issue!!!
How to conjecture such lemma q? Get better understanding
Interactive Theorem Proving

Issue!!!
How to conjecture such lemma q? Get better understanding
Which is the reliable source?
Interactive Theorem Proving

Issue!!!

How to conjecture such lemma q ? Get better understanding
Which is the reliable source ? Reachable States
Interactive Theorem Proving

Issue!!!

How to conjecture such lemma q? Get better understanding
Which is the reliable source? Reachable States  Machine Learning
Interactive Theorem Proving

Issue!!!

How to conjecture such lemma q? Get better understanding
Which is the reliable source? Reachable States Machine Learning
Description of Reachable states is first-order logic formulas!!!
Interactive Theorem Proving

Issue!!!

How to conjecture such lemma q? Get better understanding
Which is the reliable source? Reachable States Machine Learning
Description of Reachable states is first-order logic formulas!!!

Inductive Logic Programming
TCP – a Communication Protocol
Communication Protocol

TCP – a Communication Protocol
Communication Protocol

TCP – a Communication Protocol
Communication Protocol

Sender

Can I send the next one?

Data Channel

0 ... 0

Receiver

Plz send the new one

Ack Channel

a0 ... a0

Next 0

Buffer 0 nil
Two simplified versions of TCP
- Alternating Bit Protocol (ABP): unbounded channels
- Simple Communication Protocol (SCP): bounded channels – capacity 1
Communication Protocol

A system state \( s = \langle sb, b, rb, buf, dchan, achan \rangle \)
Communication Protocol

Reliable Communication Property The sequence of numbers sent by Sender is successfully received by Receiver, no drop or duplicated ones

\[(sb(S) = rb(S) \rightarrow mk(p(S)) = (p(S) \text{ buf}(S))) \land (sb(S) \neq rb(S) \rightarrow mk(p(S)) = \text{buf}(S))\]

Lemmas for SCP verification

\[achan(S) = c(b) \rightarrow (sb(S) = b \lor rb(S) = b)\].

\[(dchan(S) = c( < b, n > ) \land rb(S) = b) \rightarrow p(S) = n\].

\[(dchan(S) = c( < b, n > ) \land rb(S) = b) \rightarrow sb(S) = b\].

\[(dchan(S) = c( < b, n > ) \land achan(S) = c(b')) \rightarrow sb(S) = b' \lor rb(S) \neq b\]

*S : a variable of System states
Method & Architecture of Tool Used

System specification for theorem proving

YAST

System specification for model checking

Bounded model checker

Data structures

State machine

Input

Background knowledge (Types & Predicates)

E⁺ (reachable states)

E⁻ (unreachable states)

Normal learning mode
or
Learning form positive examples only

Prolog
Method & Architecture of Tool Used

**Examples**

```prolog
state(f, 0, t, [0], c(p(f, 0)), c(f)).

...:

:- state(f, s(0), f, [0], c(p(f, s(s(0))), c(t)).

...:
```

**Background knowledge**

```prolog
pnat(0).

pnat(s(X)) :- pnat(X).

...:

mk(0, [0]) :- !.

mk(s(N), [s(N) | L1]) :- pnat(N), mk(N, L1).

...:
```

**Mode declaration**

```prolog
%scp
:- modeh(1, state(+bool, +pnat, +bool, +nlist, c(p(+bool, +pnat)), c(+bool))).

%abp
:- modeh(1, state(+bool, +pnat, +bool, +nlist, [p(+bool, +pnat) | +pqueue], [+bool | +bqueue))).

:- modeb(1, neg(+bool, +bool))?

:- modeb(1, mk(+pnat, +nlist))?

...:
```

```prolog
* < sb, p, rb, buf, dchan, achan>
```

5
# Case Studies

<table>
<thead>
<tr>
<th>Input</th>
<th>输出</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCP</td>
<td>Background knowledge B</td>
</tr>
<tr>
<td>- Boolean</td>
<td>- Original learning mode</td>
</tr>
<tr>
<td>- Natural numbers</td>
<td></td>
</tr>
<tr>
<td>- List of natural numbers</td>
<td>- Learning form positive examples only</td>
</tr>
<tr>
<td>ABP</td>
<td>-(SCP’s B)</td>
</tr>
<tr>
<td>- Queue of boolean values</td>
<td></td>
</tr>
<tr>
<td>- Queue of &lt; b, n &gt;</td>
<td>- Learning form positive examples only</td>
</tr>
<tr>
<td>- User-defined predicates</td>
<td></td>
</tr>
</tbody>
</table>
SCP Case Study

• Set 1
  – state(A,B,C,D,c(p(A,B)),c(E)) :- mk(B,D).
  – state(A,B,A,C,c(p(D,E)),c(A)) :- neg(A,D), mk(B,[B | C]).
  – state(A,B,A,C,c(p(A,B)),c(A)) :- mk(B,[B | C]).

• Set 2
  – state(A,B,C,D,c(p(A,B)),c(A)) :- neg(A,C).
  – state(A,B,C,D,c(p(A,B)),c(C)) :- neg(A,C), mk(B,D).
  – state(A,B,A,C,c(p(D,E)),c(A)) :- neg(A,D), mk(B,[B | C]).

* < sb, p, rb, buf, dchan, achan>
SCP Case Study

- **Set 1**
  - \( \text{state}(A,B,C,D,c(p(A,B)),c(E)) :- \text{mk}(B,D). \)
  - \( \text{state}(A,B,A,C,c(p(D,E)),c(A)) :- \text{neg}(A,D), \text{mk}(B,[B \mid C]). \)
  - \( \text{state}(A,B,A,C,c(p(A,B)),c(A)) :- \text{mk}(B,[B \mid C]). \)

- **Set 2**
  - \( \text{state}(A,B,C,D,c(p(A,B)),c(A)) :- \text{neg}(A,C). \)
  - \( \text{state}(A,B,C,D,c(p(A,B)),c(C)) :- \text{neg}(A,C), \text{mk}(B,D). \)
  - \( \text{state}(A,B,A,C,c(p(D,E)),c(A)) :- \text{neg}(A,D), \text{mk}(B,[B \mid C]). \)

Manually conjecture
\[
( \text{achan}(S) = c(b) \land \text{sb}(S) \neq b ) \rightarrow \text{rb}(S) = b
\]

* < sb, p, rb, buf, dchan, achan>
ABP Case Study
ABP Case Study

state(A,B,C,D,[p(A,B)|E],[A|F]) :- neg(A,C), gap0(p(A,B),E).
state(A,B,C,D,[p(A,B)|E],[C|F]) :- mk(B,D), gap0(p(A,B),E).
state(A,B,A,C,[p(D,E)|F],[A|G]) :- neg(A,D), succ(E,B), mk(B,[B|C]), gap1(p(A,B),F).
state(A,B,A,C,[p(A,B)|D],[A|E]) :- mk(B,[B|C]), gap0(p(A,B),D).
state(A,B,A,C,[p(A,B)|D],[E|F]) :- neg(A,E), mk(B,C), gap0(p(A,B),D).

gap0(P,[]) :- bnpair(P).
gap0(P,[P|T]) :- bnpair(P), gap0(P,T).

gap1(P,[]) :- bnpair(P).
gap1(P1,[P2|T]) :- bnpair(P1), bnpair(P2), ((P1 \= P2, next(P2,P1), gap1(P1,T)); gap0(P1,T)).
Conclusion

• ILP can characterize reachable states from a system specification
• Some useful lemmas are manually conjectured
• Some non-trivial characteristics need a non-trivial background knowledge
Future work

• Systematically find non-trivial predicates
  – Predicate Invention
• Systematically conjecture useful lemmas from the characteristics