Using automated theory formation to discover invariants of Event-B models*

Maria Teresa Llano, Andrew Ireland
Heriot-Watt University

Alison Pease
University of Edinburgh

Simon Colton, John Charnley
Imperial College London

1 Introduction

Formal methods have been successfully used for the development of safety-critical systems; however, the need for skilled knowledge when writing formal models and reasoning about them represents a major barrier in the adoption of formal methodologies for the development of non-critical applications. A key aspect in the verification of formal models and in the development of reliable systems is the identification of invariants. However, finding correct and meaningful invariants for a model represents a significant challenge. We have used automated theory formation (ATF) techniques to automatically discover invariants of Event-B models. In particular, we use Colton's HR system [2] to explore the domain of Event-B models and suggest potential invariants.

2 Automated theory formation with HR

HR [2] performs descriptive induction to form a theory about a set of objects of interest which are described by a set of background concepts. The theories HR produces contain concepts that relate the objects of interest, and conjectures that relate the concepts. HR builds new concepts from old ones using a set of 15 generic production rules, including the *negate* rule that finds the complement of a concept and the *split* rule that extracts the list of examples of a concept which share a given value. For each new concept formed, HR calculates the set of examples which have the properties described by the new concept definition. Using these examples, the definition, and information about how the concept was constructed and how it compares to other concepts, HR estimates how interesting the concept is via a weighted set of interestingness measures, and this drives a heuristic search. As it constructs concepts, it looks for empirical relationships between them, and formulates conjectures whenever such a relationship is found. In particular, HR forms equivalence conjectures whenever it finds two concepts with exactly the same examples, implication conjectures whenever it finds a concept with a proper subset of the examples of another, and non-existence conjectures whenever a new concept has an empty set of examples.

3 Discovering invariants of Event-B models

The first step in the analysis of candidate invariants of Event-B models is the generation of a domain file that provides HR with background knowledge about the system being modelled. In the context of Event-B, sets, constants, variables and events represent concepts of the domain while the list of examples for each of these concepts is given through the generation of animation traces. We use the ProB animator [4] to animate Event-B models. Each step of a trace represents the state of the system at a particular time point. A step can be a *good* or a *bad* state, representing desirable or undesirable behaviour: these are represented as concepts in HR and used to identify general conjectures that may be invariants of the system. States form the objects of interest in the theory, *i.e.*, we are interested in concepts and conjectures that are associated with states. A conjecture is considered as a candidate invariant if either it applies to all states, or it applies only to all good or all bad states. Concepts can be considered as invariants, or part of an invariant, if they represent properties which remain unchanged in the domain: this is captured by HR's interestingness measure *variety*.

3.1 Illustrative example

In this example we study a fragment of Abrial's cars on a bridge model [1]. The model consists of an island connected to the mainland by a bridge. The bridge has one lane; meaning that cars can travel only in one direction. HR forms the conjecture that represents this invariant as follows:

1. An animation trace with 9 steps is generated through ProB. Each step represents a state (s00 - s08).

2. For each state, we supply HR with the concepts of cars going to the island (*cti*) and cars going to the mainland (*ctm*). The data tables of these concepts are shown in figure 1(a) (where A is the state and B is the number of cars).

*This work is supported by EPSRC grants EP/F036647, EP/F035594 and EP/F037058 and BAE Systems.

3. The split production rule is applied to extract the states for which the number of cars is 0. The data tables obtained are shown in figure 1(b).
4. HR then applies the negate rule to both concepts, producing the concept of the states for which the number of cars is different from 0. The resulting data tables are shown in figure 1(c).

cti(A,B)	
s00	0
s01	1
s02	2
s03	1
s04	0
s05	0
s06	0
s07	0
s08	0

ctm(A,B)	
s00	0
s01	0
s02	0
s03	0
s04	0
s05	1
s06	2
s07	1
s08	0

(a) Core concepts

cti(A,0)	ctm(A,0)
s00	s00
s04	s01
s05	s02
s06	s03
s07	s04
s08	s08

(b) Split production rule

¬cti(A,0)	¬ctm(A,0)
s01	s05
s02	s06
s03	s07

(c) Negate production rule

Figure 1: Theory formation steps.

5. Finally, HR finds that the concept $\neg cti(A,0)$ is subsumed by the concept $ctm(A,0)$, and the concept $\neg ctm(A,0)$ is subsumed by $cti(A,0)$, and thus, forms two (logically equivalent) implication conjectures which state that cars can travel only in one direction:

$$\textbf{conj}_\textbf{1}\textbf{:}\ \forall A.(state(A) \wedge \neg cti(A,0)) \Rightarrow ctm(A,0)$$
$$\textbf{conj}_\textbf{2}\textbf{:}\ \forall A.(state(A) \wedge \neg ctm(A,0)) \Rightarrow cti(A,0)$$

3.2 Further examples

A second example is that of a vending machine. As in the previous example, the objects of interest are states, which take various values, such as the amount of stock or money currently in the machine (these are called stock and machineMoney respectively). States were labelled good or bad and given to HR so that it may develop a theory. In particular, we labelled any states with a negative amount of stock as bad states. After a run of 1,000 theory formation steps, HR found the invariants that the amount of money in the machine is always either zero or positive, and all good states always have a positive or zero amount of stock:

1. For all states S, $machineMoney(S,M) \rightarrow M \geq 0$, and
2. For all states S, $good(S) \wedge stock(S,A) \rightarrow A \geq 0$.

In a separate run, we represented products of the vending machine in terms of sets labelled *available*, *limited* and *soldout*. HR found that a product cannot be in two sets at the same time: that is, $available \cap limited = \emptyset$, $available \cap soldout = \emptyset$ and $limited \cap soldout = \emptyset$. It also found the invariant that the initial amount of stock multiplied by the value of an accepted coin is equal to the amount of money in the machine plus stock multiplied by the value of an accepted coin.

4 Future Work and Conclusions

This is the first time that ATF has been used to reason over Event-B models. While these results are preliminary and based on small examples, we believe that they will scale up over larger models and provide a useful way of automatically reasoning about the models. We are now developing this work in various directions; including scaling up, further automation between stages, and exploring how to generate gluing invariants. Furthermore we expect to introduce a flexibility based on human reasoning, by applying Lakatos's philosophical theory of reasoning [3] to the verification of user-given invariants. Lakatos's theory is based on how mathematicians use counterexamples to trigger concept and conjecture development: in [5] we detail ways in which we expect to use these techniques to repair faulty invariants as well as providing a way of analysing failed proof obligations.

References

[1] J.-R. Abrial. *Modelling in Event-B: System and Software Engineering*. Cambridge University Press, 2010.
[2] S. Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.
[3] I. Lakatos. *Proofs and Refutations*. CUP, Cambridge, UK, 1976.
[4] M. Leuschel and M. Butler. ProB: an Automated Analysis Toolset for the B Method. *Journal Software Tools for Technology Transfer*, 10(2):185–203, 2008.
[5] A. Pease, A. Smaill, S. Colton, A. Ireland, M. Llano, R. Ramezani, G. Grov, and M. Guhe. Applying Lakatos-style reasoning to AI problems. In J. Vallverdú, editor, *Thinking Machines and the philosophy of computer science: Concepts and principles*, pages 149–174. IGI Global, PA, USA, 2010.