The Use of Automated Theory Formation in Support of Hazard Analysis

The Use of Automated Theory Formation in Support of Hazard Analysis*

Andrew Ireland[1], Maria Teresa Llano[2], and Simon Colton[2,3]

[1] School of Mathematical and Computer Sciences,
Heriot-Watt University, Edinburgh, UK
a.ireland@hw.ac.uk
[2] Department of Computing, Goldsmiths, University London, London, UK
m.llano@gold.ac.uk,s.colton@gold.ac.uk
[3] Games Academy, Falmouth University, Cornwall, UK

Abstract. Model checking and simulation are powerful techniques for developing and verifying the design of reactive systems. Here we propose the use of a complementary technique – automated theory formation. In particular, we report on an experiment in which we used a general purpose automated theory formation tool, HR, to explore properties of a model written in Promela. Our use of HR is constrained by meta-knowledge about the model that is relevant to hazard analysis. Moreover, we argue that such meta-knowledge will enable us to explore how safety properties could be violated.

Keywords: Formal methods, verification, hazard analysis

1 Introduction

Typically we have in mind a set of desired properties when we begin to develop a formal design model. Once constructed we verify our design model against the given properties. Here we propose a complementary approach to how the properties of a formal design model are obtained and used post-verification. Our starting point is SPIN and the Promela modelling language [6]. Firstly, we propose the use of a general purpose automated theory formation tool, HR[4] [2], to search for properties within Promela simulation traces. While such an approach will find properties that we expect of our models, it may also find properties of interest that we did not anticipate. Secondly, we propose an approach for discovering how a formally verified design could fail. That is, a formal counter-part to step 2 of Leveson's STPA hazard analysis technique [8] where one considers how dysfunctional behaviour could emerge. Both aspects of our proposal rely on meta-knowledge to constrain the search for properties as well as dysfunctional

* The work reported here is funded by EPSRC Platform Grant EP/N014758/1. We thank the three anonymous NFM 2018 reviewers for their constructive feedback.

[4] HR is derived from the initials of the mathematicians Godfrey Harold Hardy and Srinivasa Aiyangar Ramanujan.

behaviours. While meta-knowledge places an additional burden on the designer, we believe that it will deliver benefits during hazard analysis.

2 Background

HR was originally implemented as a system for *automated theory formation* (ATF) in domains of pure mathematics [2, 4], e.g. the invention of integer sequences [3] and large-scale algebraic classification [11]. HR has also been used within the context of formal methods, specifically in discovering invariants from Event-B [1] animation traces [9]. HR forms theories about a domain through an iterative application of general purpose *production rules* (PRs) for concept invention. Each PR works by performing operations on the content of one or two input data tables – where a data table represents a concept by means of a set of examples. An application of a PR produces a new table, i.e. a new concept. HR then searches for relationships between the new concept and the concepts already in the theory. Specifically, it is looking to see if the new concept is: i) equivalent to an existing concept; ii) subsumed by or subsumes an existing concept; or iii) empty. These relationships take the form of equivalence, implication, or non-existence conjectures, respectively.

To illustrate, we show in Figure 1 the data tables used by HR to produce the concept of prime numbers. Thousands of PR applications occur during the ATF process, here we focus only on the specific PR applications that lead to the concept of prime numbers. Firstly, HR is given the concept of a divisor, as shown partially in Figure 1 for integers from 1 to 10 ($b|a$ where b is a divisor of a). Secondly HR would apply the *size* PR with the parameterisation $< 1 >$ to count the number of tuples of each entry in column 1 (the Tau function table). HR then takes in this new concept and applies the *split* PR with the parameterisation $< 2{=}2 >$ to extract the entries in the previous data table whose value in the second column is 2. The resulting table defines the concept of a prime number.

Assuming the concept of non-square numbers has been formed previously by HR, with a data table formed by the examples {2, 3, 5, 6, 7, 8, 10} and the logical construction $[a] : \neg(\exists\ b.(b|a\ \&\ b*b=a))$. HR would then detect that the data table for the concept of non-square numbers subsumes the data table for the concept of prime numbers. That is, it sees that all of its prime numbers are also non-squares, and so conjectures that this is true for all prime numbers as follows:

$$\underbrace{2\ =\ |\{b\ :\ b|a\}|}_{\text{prime number}} \rightarrow \underbrace{\neg(\exists\ b.(b|a\ \&\ b*b=a))}_{\text{non-square number}}$$

Because of the empirical nature of this process, false conjectures may be generated. Third party formal reasoning tools are employed to identify such false conjectures. Conversely, sometimes when developing a theory, conjectures which do not fully satisfy all the given examples may still be of interest. HR has a feature that allows such *near conjectures* to be identified, i.e. a user supplies a

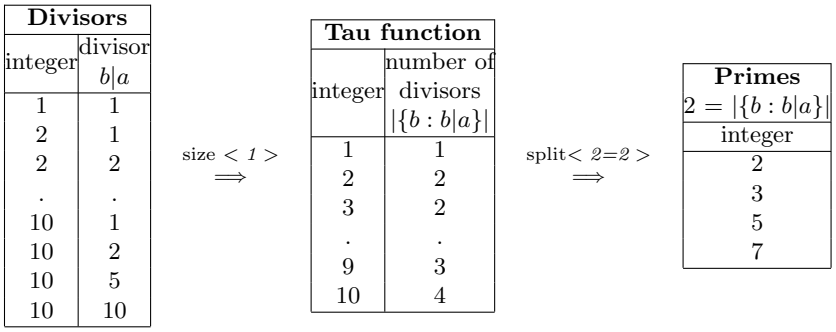

Fig. 1. Steps applied by HR to produce the concept of prime numbers.

lower bound (percentage) on the number of examples that must be satisfied for a conjecture to be of interest.

3 Experiments with a Simple Design Model

For the purposes of our experiment we use a model of a laser control system. The laser is housed within a protective container which we refer to as the `Laser` unit. Access to the laser is via a door which is directly controlled by an `Operator`. In order to switch on the laser the `Operator` uses a `Control` unit. As well as controlling the power supply to the laser, `Control` also illuminates a warning light when the power is on. The model is both deliberately simple and poorly designed from the perspective of safety. Our Promela model of the system is shown in Figure 2. There are two use cases an `Operator` can perform:

PowerOn: The `Operator` closes the `Laser` unit door then selects the power to be switched on. `Control` reacts by illuminating the power-on-light and then requests for power to be supplied to the `Laser`. The `Laser` unit then switches on the power and sends confirmation to `Control`.

PowerOff: The `Operator` selects the power to be switched off. `Control` reacts by requesting the `Laser` unit to stop supplying power to the laser. Once the `Laser` unit confirms the power is off, `Control` then switches off the power-on-light and the `Operator` opens the `Laser` unit door.

3.1 Applying HR to the Laser Control System

In our experiment, the state variables that occur within the Promela model provide the basic concepts that are given to HR. As highlighted above, the simulation traces produced by SPIN represent the examples that HR requires in order to form conjectures. False conjectures can be identified using the SPIN model checker. One of the contributions of this work is an extension to HR that enabled it to discover temporal properties. Technically, *response properties*

```
bool opr_select_power_on   = false;
bool opr_door_closed       = false;
bool ctr_request_power_on  = false;
bool ctr_power_on_light    = false;
bool lsr_power_on          = false;
bool lsr_confirm_power_off = true;

active proctype Operator(){
do
:: !opr_door_closed ->
       opr_door_closed = true; opr_select_power_on = true; ctr_power_on_light;
::  opr_select_power_on ->
       opr_select_power_on = false; !ctr_power_on_light; opr_door_closed = false;
od;}

active proctype Control(){
do
:: opr_select_power_on ->
      ctr_power_on_light = true; ctr_request_power_on = true;
      !lsr_confirm_power_off; !opr_select_power_on; ctr_request_power_on = false;
      lsr_confirm_power_off; ctr_power_on_light = false;
od;}

active proctype Laser(){
do
:: ctr_request_power_on ->
      lsr_confirm_power_off = false; lsr_power_on = true; !ctr_request_power_on;
      lsr_power_on = false; lsr_confirm_power_off = true;
od;}
```

Note that state variables associated with the `Operator` are prefixed by `opr`. Similarly, the prefixes `ctr` and `lsr` are associated with the `Control` and `Laser` units respectively.

Fig. 2. A Simple Laser Control System.

provided the greatest challenge and involved HR forming conjectures by relating two concepts as follows: if there is a state S_i for which the first concept has an example in the trace, then there is a state S_j for which the second concept has an example in the trace and $j \geq i$; i.e. whenever the first concept happens, the second concept happens eventually. These conjectures are written in HR as:

$$\forall s_i \ . \ \mathit{state}(s_i) \wedge \mathit{concept1}(s_i, ...) \rightarrow \exists s_j \ . \ \mathit{state}(s_j) \wedge \mathit{concept2}(s_j, ...) \wedge j \geq i$$

where, $\mathit{state}(s_x)$ refers to a step in the simulation trace occurring in time x, and $\mathit{concept}(s_y, ...)$ represent a tuple with the value of a concept in state s_y. Given that a simulation trace provides only a partial exploration of the state space, we have also extended HR to generate *near-response conjectures* – a natural generalization of the near-conjecture notion outlined in §2. In this way, we reduce

the chances of missing properties that are rejected because a response property is violated by virtue of the finiteness of the traces given to HR.

3.2 Discovery Properties in support of Hazard Analysis

HR will generate thousands of conjectures for a given simulation trace. As noted above we rely upon the designer to provide meta-knowledge about their models so that HR can constrain its search for interesting properties. Here we consider three kinds of meta-knowledge: Promela statements that represent: *hazards*, *defences* and *biddable* actions:

$$\langle \text{hazard} \rangle = \{\texttt{lsr\_power\_on}\}$$
$$\langle \text{defence} \rangle = \{\texttt{opr\_door\_closed}\}$$
$$\langle \text{biddable} \rangle = \{\texttt{opr\_door\_closed, opr\_select\_power\_on}\}$$

Note that we adopt Reason's [10] use of "defence" to denote mechanisms which prevent hazardous situations arising while "biddable", inspired by Jackson's work on Problem Frames [7], denotes actions performed by a human. In general we envisage an extensible meta-language for annotating model elements with respect to the role they play within a design and its context. Armed with this meta-knowledge we focus the search on three generic temporal properties.

Firstly, we search for safety conjectures of the form:

$$\Box(\langle hazard \rangle \rightarrow \langle defence \rangle) \tag{1}$$

That is, wherever a hazardous state is identified, as defined by the designer, then it must follow that a defence also holds. For our laser example HR generates 1565 implication conjectures. Using (1), and the meta-knowledge associated with the model, these are reduced to one conjecture:

$$\Box(lsr\_power\_on \rightarrow opr\_door\_closed) \tag{2}$$

This suggests that while power is being supplied to the laser there is only one defence in-place. Moreover, the defence *opr_door_closed* is also a member of the biddable set, i.e. the sole defence mechanism relies upon a human operator. Biddable operations are typically more vulnerable than say electrical or mechanical mechanisms involving redundancy. We will return to this point in §3.3. While (2) could have been anticipated by a designer, we would argue that the following properties are less obvious:

$$\Box\,(lsr\_power\_on \rightarrow ctr\_power\_on\_light) \tag{3}$$
$$\Box\,(lsr\_power\_on \rightarrow !lsr\_confirm\_power\_off) \tag{4}$$

Note the violation of (3) would not directly effect the safety of the system. However, if the power-on-light is not illuminated when the power is on then potentially the `Operator` could believe that it is safe to open the door, i.e. the violation of (3) could lead to the violation of (2). Property (4) ensures that `Control` is

correctly informed of the laser's status while power is being supplied. A violation of (4) would lead to `Control` incorrectly switching off the power-on-light with the potential violation of the high-level safety property described above. It is worth noting that HR may find non-existence conjectures corresponding to (1) for a given model. The presence of such non-existence conjectures would provide a warning to the designer about the lack of defences within their model.

A second form of safety conjecture takes the form:

$$\Box\,!\,(\langle \mathit{hazard} \rangle \wedge\, !\langle \mathit{defence} \rangle) \tag{5}$$

Here we constrain HR to find conjectures where a hazard and the negation of a defence hold. Corresponding non-existence conjectures will strongly suggest safety properties. Here HR generates 20 non-existence conjectures. We follow a similar heuristic process as performed with the previous safety constraint. Firstly, we instruct HR to identify the non-existence conjectures that involve hazardous states. This narrows down the search to 10 conjectures. Next, we direct HR to further focus on conjectures that involve the negation of a defence, resulting in 5 conjectures. We end by removing non-existence conjectures that involve states that do not belong to the hazard or defence sets. The following instantiation of (5) is identified:

$$\Box\,!\,(\mathit{lsr\_power\_on} \wedge\, !\mathit{opr\_door\_closed})$$

Thirdly we search for response conjectures of the form:

$$\Box(\langle \mathit{biddable} \rangle \rightarrow \Diamond \langle \mathit{hazard} \rangle) \tag{6}$$

For our model of the laser control system, HR generates 22950 response conjectures and 18228 near-response conjectures (with lower threshold of 95%). Two near-response instantiations of (6) are given below with the percentage match:

$$\Box(\mathit{opr\_select\_power\_on} \rightarrow \Diamond \mathit{lsr\_power\_on}) \;\; 99.43\%$$
$$\Box(\mathit{opr\_door\_closed} \rightarrow \Diamond \mathit{lsr\_power\_on}) \;\; 97.93\%$$

3.3 Breaking Properties

We now go beyond conventional verification, and consider how safety properties could be violated. Specifically, we focus on the safety property (2). There are a number of scenarios which could lead to the violation of (2). We consider here the simplest of scenarios. Given that `opr_door_closed` is a member of both the $\langle \mathit{defence} \rangle$ and $\langle \mathit{biddable} \rangle$ sets, then this is strongly suggestive of exploring a dysfunctional variate of the model in which `opr_door_closed` does not hold when it is supposed to hold. This dysfunctional behaviour can be included within the model by the addition of the following case to the definition of `Operator`:

```
:: !opr_door_closed ->
        opr_select_power_on = true; ctr_power_on_light;
```

SPIN will show that (2) is violated by the modified model. It is then the task of the designer to refine their design so as to mitigate for such an `Operator` error. In practice, scenarios that lead to single points of failure are typically not so simple. Part of our ongoing work is to further develop the idea outlined here. Specifically dealing with models where there are multiple defences and the application of a defence involves a chain of events.

4 Future Work and Conclusion

The experiments reported here demonstrate the potential for using meta-knowledge to guide HR in searching for properties relevant to hazard analysis. The version of HR we used is HR2. Running on a Macbook Pro (OS X Mavericks, processor 2.6 GHz Intel Core i5), HR2 could only deal with traces of 300 steps. As part of future work we aim to move to HR3 [5], which is significantly faster and more memory-efficient. This opens up the possibility of working with much larger datasets and exploring real-time theory formation. We chose to use HR2 for our initial experiments because of its GUI and support for browsing the results of the theory formation process. Such features will be added to HR3. The experiments reported here have also highlighted the need for a mechanism that will allow users to more easily tailor the conjecture making phase. Longer-term we envisage a computer-based *design assistant*, built upon HR3, which would run alongside a formal verification tool such as SPIN. The aim of such an assistant would be to help the designer explore the design space, as well as play a useful role during hazard analysis. What we propose compliments current practice, with the potential for identifying concerns which may otherwise be overlooked.

References

1. J-R. Abrial. *Modeling in Event-B: system and software engineering.* Cambridge University Press, 2010.
2. S. Colton. *Automated Theory Formation in Pure Mathematics.* Springer, 2002.
3. S. Colton, A. Bundy, and T. Walsh. Automatic invention of integer sequences. In *Proceedings of AAAI*, 2000.
4. S. Colton and S. Muggleton. Mathematical applications of Inductive Logic Programming. *Machine Learning*, 64:25–64, 2006.
5. S. Colton, R. Ramezani, and T. Llano. The HR3 discovery system: Design decisions and implementation details. In *Proceedings of the AISB Symposium on Computational Scientific Discovery*, 2014.
6. G.J. Holzmann. *The SPIN Model Checker.* Pearson Education, 2003.
7. M. Jackson. *Problem frames: analysing and structuring software development problems.* Addison-Wesley, 2001.
8. N.G. Leveson. *Enineering a Safer World.* MIT, 2011.
9. T. Llano, A. Ireland, and A. Pease. Discovery of invariants through automated theory formation. *Formal Aspects of Computing*, 26, 2011.
10. J. Reason. *Organizational Accidents Revisited.* Ashgate, 2016.
11. V. Sorge, A. Meier, R. McCasland, and S. Colton. Automatic construction and verification of isotopy invariants. *Journal of Automated Reasoning*, 40(2-3), 2008.