Applying theorem proving to formal property languages

Mike Gordon Cambridge mjcg@cl.cam.ac.uk

It has been said that *assertion-based methodologies offer the same leap in productivity for verification that logic synthesis did for design entry*[1]. Assertions are written in a property language and there are currently three competing languages: PSL/Sugar, SystemVerilog Assertions (SVA), both from the Accellera organisation, and the proprietary *e* language from Verisity.

We have been looking at various ways of using theorem proving to debug and execute the formal semantics of PSL/Sugar[2]. In the talk I will review this work and then discuss progress on further applications of theorem proving that are being explored.

A problem that besets the current use of assertions is that different tools use different property languages, and it is unclear whether only one language will survive, or if there will be several, each supported by different tools. We are investigating formally verifying translation mappings between languages, and then creating automated translators that execute the mappings. This will enable properties to be 'ported' between languages with a formal guarantee that meaning is preserved.

Another problem is difficulty in being sure that requirements have been accurately captured by sets of properties. The standard property languages are low level and there is a large gap between natural language descriptions of behaviour (e.g. as found in bus architecture manuals) and property language syntax. We are investigating sequential constructs that are higher level and more natural to use than those in the current industrial languages. The hope is that more powerful semantic frameworks can provide more 'engineer friendly' specification methods that can be supported by theorem proving tools. Initially we are experimenting with Interval Temporal Logic (ITL, Moskowski[3]) and Event Sequence Language (Fisler[4]). ITL provides predicates on intervals that go beyond regular expressions and Event Sequence Languages formalise concepts derived from timing diagrams. So far we have made a very preliminary foray into replacing the regular expressions of PSL with ITL formulas, and the results suggest potential for power and elegance. We also plan to add richer data modelling capabilities, including user defined data-types as found in modern specification notations such as Z, HOL and PVS. Our goal is to use theorem proving to automate the translation of properties captured using higher level constructs into standard formats like PSL.

We are planning to adapt methods from program verification (e.g. verification conditions) for statically checking, by theorem proving, that high level properties hold of models. Current property languages only contain boolean variables, and consequently support decision algorithms. Our extensions immediately take us into undecidable territory, but as we plan to use theorem proving, we hope that significant bespoke automation can be developed for common idioms.

We envisage our theorem proving 'point tools' being combined into an integrated environment, or *Property Studio*, to aid specifiers create properties and to provide facilities for converting them into the forms and notations they need (e.g. factoring complex properties into equivalents sets of simpler ones, then expressing them in a specific property language). A methodological question our research will address is whether a theorem prover can be an effective platform for implementing such an environment.

Summary: brief review of achievements in applying theorem proving to PSL/Sugar, followed by a report on ongoing research (much of it not started when this abstract was written).

---

[1] `http://www.eetimes.com/story/OEG20021112S0031`
[2] `http://www.cl.cam.ac.uk/users/mjcg/Sugar/`
[3] `http://www.cse.dmu.ac.uk/~cau/itlhomepage/`
[4] `ftp://ftp.cs.wpi.edu/pub/techreports/pdf/03-24.pdf`