FORMAL VERIFICATION SUPPORT FOR ELLA

IED 4/1/1357

Verification Strategies Report*

Howard Barringer Graham Gough Brian Monahan

Department of Computer Science
University of Manchester†
Oxford Road, Manchester, UK.
email: howard@cs.man.ac.uk

March 1991

†The University of Manchester acknowledges support from the SERC via research grant GR/F 38174.

1 Verification Strategy

The verification strategy adopted in the project has, from the outset, relied on two complementary approaches, First, the use of *state based* verification, based on model checking and related approaches, and send the use of syntactically driven *transformations* to establish the semantic equivalence or containment of one ELLA design in another. This document will concentrate on the former approach, but we believe that the semantic basis necessary for the development of of state based verification will also be used in the development of the transformational approach later in the project.

State based verification relies on the construction of an automaton from a piece of program text (in this case an ELLA description) which represents the runs or behaviours of the program or circuit. This automaton can then be compared with automata generated from other descriptions, or tested to determine whether the circuit satisfies properties described in some specification language, such as temporal logic. Thus the automaton construction process is central to this style of verification.

In an earlier deliverable [BHM90, BGM91], an operational semantics in terms of transition rules was given for a language, *LD*, which is an abstraction of a subset of ELLA, with a view to developing this into a full ELLA semantics during the course of the project. It was thought that, as well as providing a much needed formal semantics for ELLA, this semantics would also form the basis of the automaton generation process. Techniques for generating automata directly from rule based semantics have been illustrated in earlier work.

The attached paper[BGL$^+$91], however, adopts a different approach, which will provide the technical base for the next stage of the work on verification. Rather than constructing the automation from a transition system generated from the semantic rules, we have taken a more direct approach. Each construct of Boolean Kernel ELLA is given a semantic denotation which is itself an automaton; these are composed using direct automata operations to give the denotation of a complete description. This direct approach has several advantages; the automata constructed are more clearly related to the original program text than those obtained using the indirect approach, and they are appreciably smaller. The indirect approach would involve building a larger automaton, which would then be minimised. However, we do not feel that the earlier, rule based work should be abandoned, since it may well be a more appropriate basis for the later transformation based approach. It will, of course, be necessary to show the equivalence of the semantics given by the two approaches.

While the attached paper provides the basis for our approach to state based verification, there are of course many issues still to be addressed. The remainder of this document will outline some of these, and indicate the approach which we intend to follow.

2 Problems and Future Directions

2.1 Extensions to support different specification formalisms

Given that several different specification formalisms will be supported, there is a clear need to ensure they share a common semantic underpinning, such as IO automata. This common semantic model is essential to permit different objects to be combined together, no matter what formalism was used originally to define them. Furthermore, this will facilitate the establishment of a clean interface between different verification components within the environment. To this end, we envisage that an Automaton Interface Language will be developed as part of establishing this common tools interface.

2.2 Towards taming the complexity of verification

The main thrust of the verification work is to implement automaton-based verification and transformation techniques that are applicable to a broad class of industrial-scale, hardware-oriented problems. This objective requires that the implementation of these techniques face head on a number of difficult and subtle complexity questions.

The main problem concerns the inherent space complexity of the verification process. Semantic analysis of automaton structures proceeds by constructing a graph based model structure, to which various graph theoretical analysis techniques are then applied. The main idea is to avoid any process that involves the exhaustive enumeration of ‘runs’ or executions, as this is an infinite process in general. This relies on generating appropriate semantic structures so that this potentially infinite search becomes a finite, bounded analysis.

Clearly, one may envisage techniques that only generate fragments of this semantic graph, as and when required during analysis. However, it is certainly not the case that this graph can always be analysed completely in this localised, piecemeal fashion. For example, liveness properties depend in general upon the global topology of the entire semantic graph. Even so, it is certainly of interest to find low-cost techniques which can reduce the amount of the semantic graph structure needed for the analysis at any one time. From the environment point of view, there are serious and interesting technology issues about how these partly static, partly dynamic objects are to be efficiently maintained. Such issues have been touched on briefly in [Arc91, BEG$^+$89].

There are a number of interesting techniques and promising avenues which are either presently under consideration or shortly will be:

IO Automata: A Mealy style automaton model in which information is carried by the transition arcs, rather than being explicitly embedded within states. Each transition arc is a directed link from one state to another and has a label structure consisting of a pair containing the input and output information. Even by itself, this generally reduces the number of states required to represent systems when compared to Moore machine models, whilst potentially increasing the number of transitions.

It is presently unclear by how much the number of transitions might increase in specific cases when going from a Moore-style automaton to an IO automaton. The examples we have looked at so far have shown little increase and sometimes a decrease. However, note that the maximum number of transitions is bounded above by the product of number of states with the total number of possible input and output mappings. Hence, if the size of state space decreases then this upper bound decreases also.

Partial mappings: We are able to use partial mappings in the transitions to record when the inputs or outputs do not care how particular labels are to be interpreted. Therefore, this provides one natural and compact way to bundle together transitions. Since the input and output signatures and the value sets are known for any IO automaton, any such partial mapping represents the set of all total mappings on the appropriate signature that are functionally consistent with the given partial mapping.

Modularity and compositionality: By this we mean the natural idea of conducting verification in a modular fashion, to promote not only the re-use of component subsystems but also the reuse of any properties that they have already been shown to possess[Kur87, GS90]. Such a verification methodology would have very considerable productivity benefits, such as the construction of systems from libraries of trusted components. Although the effort in building up such a library is considerable, the payoff in engineering terms would be repaid many times over.

Clearly, this has considerable implications for the environment in that support is essential for 'chunking', the grouping together of components with properties in common (see [Arc91, BEG$^+$89]). It is clear that, from a verification point of view, the linkage between objects that are chunked together will often be of a semantic nature in addition to other, more syntactical structural relationships.

Intensional representations: This covers *symbolic* techniques of representing information which would be otherwise highly inefficient to represent in an extensional way i.e. by means of an explicit table or enumeration.

Bundling up transitions: The use of transitions to carry information rather than in the states permits the following observation. It enables transitions with the same start and finish states to be bundled together and then treated as functions, at least in the deterministic case. This introduces the potential for introducing symbolic representations for bundled transitions.

Using predicates for sets: There are various places where sets are required in the definition of IO automata, such as in the initial states, the terminal states and the accepting states. It would then be possible to represent such sets by means of a logical predicate, rather than by a tedious and space consuming enumeration of the elements.

However, although these intensional descriptions are often compact, their main advantage is also a serious disadvantage: they are essentially *syntactic programs* which need further processing and interpretation to yield any semantic information. If they are not constrained in some way, they simply do not advance the analysis process, but merely *defer* it to the point where expansion to the corresponding extension becomes inevitable. In brief, the problem with intensional representation is that their expressiveness is too easily increased to the point where they become very hard or impossible to analyse *decidably*.

Hence, the main issue here is to carefully consider the class of symbolic expressions used for bundling transitions and representing predicates to ensure that the appropriate symbolic analysis remains *tractable* and so decidable. It is also clear that this work on symbolic manipulation has a very natural tie-up with forthcoming work in ELLA transformation later on the project.

Trace Containment, Equivalence, Implementation and Refinement: These are all examples of important relationships for which users expect to have explicit support. As these relations form the basis for most queries that a user is likely to want to know, their implementation should be as comprehensive, reliable and efficient as possible. In this regard, we are presently pursuing ideas for algorithms that perform semantic comparison of automata structures, but in a direct and non-expansive fashion.

References

[Arc91] M. Arcus. The ELLA environment. Technical Report (internal), Formal Verification Support for Ella, IED project 4/1/1357, 7th January 1991.

[BEG+89] H. Barringer, D. Edwards, G. Gough, J. Marks, B. Monahan, M. Peim, and A. Smith. Design document arising from a pre-project discussion meeting. Meeting minutes, Broomcroft Hall, November, 1989.

[BGL+91] H. Barringer, G. Gough, T. Longshaw, B. Monahan, M. Peim, and A.J. Williams. Semantics and verification of Boolean Kernel ELLA using IO Automata. Draft Technical Report, University of Manchester Department of Computer Science, 1991.

[BGM91] H. Barringer, G. Gough, and B. Monahan. Operational Semantics of Hardware Design Languages. Technical Report UMCS-91-2-2, University of Manchester Computer Science Department, February 1991.

[BHM90] H. Barringer, M. Hill, and B. Monahan. Towards an Operational Semantics of Ella. Technical Report 1.1a, Formal Verification Support for Ella, IED project 4/1/1357, October 1990.

[GS90] S. Graf and B. Steffen. Compositional minimization of finite state processes. In *Workshop on Computer-Aided Verification*. DIMACS Center, Rutgers University, Springer Verlag, June 1990.

[Kur87] R.P. Kurshan. Reducibility in analysis of coordination. In *Discrete Event Systems : Models and Applications*, pages 19–39. Lecture Notes in Control and Information Sciences 103, Springer Verlag, 1987.