Hierarchical Timed Symbolic Abstract State Machines for precise WCET estimation

Vladimir-Alexandru Paun, Bruno Monsuez, Philippe Baufreton

Hierarchical Timed Symbolic Abstract State Machines for precise WCET estimation

Vladimir-Alexandru Paun
UIIS ENSTA ParisTech
828, Boulevard des Maréchaux,
91762 Palaiseau Cedex
Email: paun@ensta-paristech.fr

Bruno Monsuez
UIIS ENSTA ParisTech
828, Boulevard des Maréchaux,
91762 Palaiseau Cedex
Email: monsuez@ensta-paristech.fr

Philippe Baufreton
Sagem - SAFRAN Electronics
Etablissement F. Hussenot - R&T
100 avenue de Paris, 91344 MASSY Cedex France
Email: philippe.baufreton@sagem.com

***Abstract*—The Abstract State Machines have been around for a while, earning their place in the embedded system world. Their formal background makes them suited for proofs, their refinement design method eases the system engineering and their apparent simplicity steepens the end user's learning curve. The numerous extensions that followed have adapted the ASMs to most of the real-time system needs. Our aim is to provide a safe, precise and adaptable worst-case execution time (WCET) estimation for processors featuring modern components. The safety property implies that among all the possible processor states, generated by the binary for all possible inputs, the ones that cause the maximal execution time must be considered. However, complex architectural components, designed to speedup the average case, make it impossible to infer local timing decisions to the global systems as the monotony is broken by the timing anomalies. Therefore, a large number of states must be analysed, generating a combinatorial explosion. Our approach starts with a value analysis of the binary code performed by abstract interpretation. The inherent imprecision of its results is taken into account by the initially concrete abstract state machine processor model. Through the use of an internal oracle, it can dynamically adapt to the lack or imprecision of information by choosing a different hierarchy level for all the impacted components. This kind of execution optimises the granularity of the run based on several strategies. State merging is also used, in order to further counter the state space explosion, in the detriment of precision. The merging is based on identified similar states through the use of equivalence classes. This also offers a leverage on the tradeoff between the precision and scalability of the analysis. Time, different abstraction levels of the processor during the same run and symbolic execution are directly added in the ASM model, as opposed to other approaches. This provides us with the needed architectural adaptability and full control over the main target of the analysis: precise WCET estimation. Taking into account a new processor becomes an engineering task as a new model for the processor is given in our extension of the ASM, with little syntactical differences. This in made possible by the seamless integration of the WCET estimation alongside the ASM semantics, which must not be changed whenever a new platform is considered.**

I. Introduction

In order to safely and precisely estimate the WCET of a processor we need a versatile model that can take into account all the possible component interaction and offer the means to confine and control the inherent state space explosion of exploring all the execution scenarios. Abstract State Machines (ASMs) have been used with success in processor modelling and verification [1], and proved in our case to ne the best candidate to describe the underlying architecture for worst-case execution time estimation. Despite the formal background which makes it suited for proofs, the ASM model can be seen as a simple language and used accordingly with a minimum time to take in hand.

II. Abstract State Machines

A. Timed ASM

The addition of time to the ASMs is not new and is present in different shapes. Timed ASM with instantaneous actions were first introduced in [2]. Both paradigms are further developed in [3] and [4] with semantics oriented on verification. The approach presented in this work incorporates concepts from previous approaches from the ASM community however it represents the time in the simplest useful manner, close to the basic ASM but on the other hand adapted to generate runs that can reduce the number of processed information. A complete formal definition of the new model is available, therefore the verification capabilities of the ASMs are preserved. One of the first step to counter the state space explosion is to enable a time-accurate model for the processor, by allowing the step of the run to change in order to make a transition directly to a different new state. Timed ASM were also designed in order to conform with the set of axioms defined in [5] like global tine, srict time progress along causal chains, absence of Zeno computations, global urgency, etc.

B. Hierarchical ASM

Precise WCET estimation is dependent on the precision of the hardware model. However generating all the exact states of the processor at every step of the run triggers a state-space explosion. We introduce the concept of dynamic choice of the refinement granularity in what we call Hierarchical ASMs. Therefore, the model itself, is able to chose on the fly the appropriate abstraction level for a given rule among several, user-defined, definitions. This can be particularly useful in the case of a processor design as the precision on the values of manipulated data, given by a value analyser for example, is not always on pair with the level of detail of the processor model. The main idea is to adapt the level of abstraction of the processor description to the precision given by the value analysis in order to master or reduce the state-space explosion inherent to the WCET analysis. The definition of the ASMs is modified in order to allow not only a single rule definition for a

rule name, but multiple, chosen according to the oracle among a list associated to each rule. A rule name assignment for an ASM state is a function that assigns to each rule name, a rule definition taking into account the choices of the Oracle and the critical locations. Choosing a certain definition for a rule name may imply also the automatic choice for other rules, explained by the notion of critical locations that we introduced for rules that use other rules locations. When conceiving the HTASM execution, we must choose the way we deal with the update sets associated with the execution trace. We need to be able to *come back* from an abstract state to a more concrete one. This means that the abstract run will potentially cover more concrete executions. In order to be able to resume to a concrete state we must be able to reconstruct the exact definition of the *store* equivalent to a particular concrete execution. In order to achieve this we can associate update sets to a certain rule name definition and collect them into an *update bag* that will not modify the general superuniverse but only allow us to recreate the exact state corresponding to the concrete state. In this way, ASMs can naturally deal with multiple parallel executions.

C. The Oracle

Whenever multiple definitions for a rule name are available, the Oracle is in charge with choosing the most adapted, according to different strategies. For example, if a rule definition depends on locations that are unknown, a more abstract version of that component is selected, that either no longer depends on those locations or allies to a whole set of values for those locations. The Oracle can also be trained, based on dynamic predictions for example it will try to minimise the state numbers based on previous success correlated to active locations and their values.

D. Symbolic ASM

Symbolic execution (SE) has already been used successfully in WCET estimation. Our analysis generates all the reachable states of the processor under all the reachable states of the program by doing a conjoint symbolic execution of the processor model and the binary. We integrated the SE directly into the ASMs, modifying the run in order to take into account symbolic values (completely unknown or baring information from the value analysis), from an additional universe, and adding the interpretation of symbolic terms.

E. State Merging

An instruction needs and generates a certain context in order to execute on a processor. The conjunction of this contexts generate similar processor states for different execution histories, [6]. Therefore the processor will find itself in similar states. This allows us to create equivalence classes for the generated states, differentiated by the notion of a distance that takes advantage of the formal definition of the processor model. Therefore, the search for merging state candidates is greatly improved with every new defined class. For example we can have identical states, where all the location values and rule definitions are the same or similar states. The decision to merge identical states is take automatically. On the other hand, similar states must be further analysed in order to decide if they should be merged. At this moment the dynamic prediction module takes over and compares the results of the execution before and after the state merge in order to make the choice. This module has another purpose that is to backtrack on the path that led to identical states in order to identify other potentially similar states.

III. Use cases

The ability to estimate the impact of state merging on the resulting WCET is important to give an idea about its inherent over-approximation. Our analysis allows the control of the state merging threshold which enables the user to decide whether a precise but long and resource consuming estimation or a less precise but faster one should be performed. Having the choice can be particularly handy, as quick product cycles influence the development process, meaning code modifications take place more often with minor changes. However, in the general case, a safe WCET estimation implies a complete system WCET analysis, which may lead to delays for the new interruptive, agile development process. Therefore a faster and less precise analysis can be useful, especially that in early development stages, the system in conceived in order to have a higher temporal budget reserves in order to allow later evolutions.

IV. Conclusions

We have presented the structure of our tool that estimates the WCET with a custom precision. Thanks to the conjoint symbolic execution, all the reachable states of the processor, running the binary, are generated. The possibility to make delayed transition is presented as a support for abstracting the processor components in order to achieve a more compact simulation. We have also introduced a model that can dynamically refine the components of the processor, preparing a framework where run-time abstraction can be made in both ways between the concrete and the more abstract definition. In order to further reduce the state space explosion, merging is used through the dynamic prediction model. The tool is currently in the implementation phases, however prototypes of a previous version, using state merging and a classical ASM processor model had give good results regarding the precision of the estimation on benchmarking code examples.

References

[1] J. K. Huggins and D. V. Campenhout, "Specification and Verification of Pipelining in the ARM2 RISC Microprocessor," *ACM Transactions on Design Automation of Electronic Systems*, vol. 3, pp. 563–580, 1997.

[2] Y. Gurevich and J. Huggins, *The railroad crossing problem: An experiment with instantaneous actions and immediate reactions*, ser. Lecture Notes in Computer Science, H. Kleine Bning, Ed. Springer Berlin Heidelberg, 1996, vol. 1092. [Online]. Available: http://dx.doi.org/10.1007/3-540-61377-3_43

[3] D. Beauquier and A. Slissenko, "A First Order Logic for Specification of Timed Algorithms: Basic Properties and a Decidable Class," *Annals of Pure and Applied Logic*, vol. 113, no. 1–3, pp. 13–52, 2002.

[4] M. Ouimet and K. Lundqvist, "The Timed Abstract State Machine Language: Abstract State Machines for Real-Time System Engineering," *Journal of Universal Computer Science*, vol. 14, no. 12, pp. 2007–2033, Jun. 2008.

[5] S. Graf and A. Prinz, "Time in State Machines," *Fundamenta Informaticae*, vol. 77, no. 1-2, pp. 143–174, May 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1365972.1365978

[6] B. Benhamamouch and B. Monsuez, "Computing worst case execution time (WCET) by Symbolically Executing a time-accurate Hardware Model," in *International MultiConference of Engineers and Computer Scientists*, vol. II, 2009, pp. 3–8.