Towards a formal semantics of the TESL specification language[⋆]

Hai Nguyen Van, Thibaut Balabonski, Frédéric Boulanger, Safouan Taha, Benoît Valiron, Burkhart Wolff, and Lina Ye

LRI, Univ. Paris-Sud, CentraleSupélec, CNRS, Université Paris-Saclay, France
`Firstname.Lastname@lri.fr`

Abstract. Most relevant industrial modeling problems depict heterogeneity issues when combining different paradigms. Designing such systems with discrete and continuous parts necessarily raises formal verification problems. We focus on a synchronous heterogeneous specification language, called TESL. In particular, it allows the expression of interrelations of clocks and — unlike other existing languages — is able to express *time-triggered behaviors* above event-triggered ones. We are interested in formalizing such a language to derive a verification framework, including model-checking, and testing. The long-term purpose of this work is to develop in the Isabelle/HOL proof assistant, a formal semantics for such a language. In the current paper, we present and compare three complementary semantic approaches for our problem.

Keywords: Heterogeneity, Synchronicity, Time Model, Logic, Formal Methods, Theorem Prover

1 Introduction

Modeling complex systems often requires several different paradigms, each with its own representation of time, whether *discrete* or *continuous*. Such a system is said to be *heterogeneous*. As an example, a car power window might be modeled using several separate sub-systems [4]: the controller is represented by timed finite state machines, while the mechanical part of the window is handled with synchronous data flows, and yet discrete events are used to model the communications on the car's bus between the components.

In this work, we consider the Tagged Events Specification Language (TESL) [5], developed to express *time-triggered behaviors*, as well as the more usual event-triggered behaviors. The language has been jointly designed with a simulator[1] that takes a specification and produces a *run* (*i.e.* an execution trace of clocks satisfying the constraints).

This paper presents our approach to the design of a formal semantics for TESL. The first goal is to develop a mathematical representation of the type of

[⋆] This work has been partially funded by the Euro-MILS project funded by the European Union's Programme FP7 under grant agreement ICT-318353.

[1] URL: `http://wwwdi.supelec.fr/software/TESL`

model that the language can describe. In particular, we wish to handle entities like clocks or run traces as well as all the constraints provided by the language.

Using this formalization, we want to investigate the existing algorithm that is used in the simulator. Our main concerns are: Does the algorithm produce a "correct" run trace? What properties may be derived? Is there any "canonical" solution?

We develop the formalization using the Isabelle/HOL proof assistant. This tool combines a programming language and a *proof* assistant. It can be used both to test intuitions and to build formal definitions and proofs. It also has code generation capabilities that allow us to produce an executable semantics, which could directly implement a simulator.

Our final purpose is to allow for automatic generation of test cases, and thus provide a good exploratory tool for developing specifications of heterogeneous timed systems.

Contribution. In this paper, we introduce and compare three possible semantics for TESL, which we respectively call "axiomatic", "trace compositional", and "algebraic". These different semantics are meant to be equivalent, but they have different strengths and weaknesses: one is close to the application, one naturally enables test cases generation, and one relies on generic mathematical structures. We discuss how these different presentations relate and compare them to the existing informal semantics used in the simulator.

2 Related work

The TESL specification language takes ideas from several sources. On one hand, it combines the (purely synchronous) clock operators of the CCSL specification language [9], adding support for time tags and time scales relations found in the Tagged Signal Model [8], which allows specifying that an event should occur after another event, with a chronometric delay Δt (on a given time scale). On the other hand, the solving algorithm relies on principles from the constructive semantics [3] of the Esterel synchronous language.

Benveniste *et al.* [2] proposed an algebra of tag structures to formally define heterogeneous parallel composition. The relationships between heterogeneous models at different levels is represented by morphisms between tag structures. Such a formalism captures logical time, physical time, causality, scheduling constraints, and so on. However, even though their theory can be used as a unifying mathematical framework to relate heterogeneous models of computations, there are no explicit relations between the time scales of different models. Furthermore, their theory is not proposed in a machine-checkable formalization, which is exactly what we want to achieve in our work.

3 Brief presentation of TESL

TESL is used to describe the behavior of *synchronous clocks*. Each has its own *time scale* with attached events, represented by *ticks* timestamped with *tags*.

They are specified by *positive constraints*, *i.e.* a set of minimal required pre-existing tags. Whereas, clocks timescales might be very diverse and can possibly be specified with *negative constraints*, which forbid some arrangements of tags. Tag domains might be continuous, discrete, or even the singleton Unit where time does not progress, as long as they remain totally ordered.

Compared with CCSL, the special nature of TESL is its ability to specify relations between the time scales of clocks and to characterize time-triggered events under three main classes of primitives.

Event-triggered implications. Such an implication is typically of the form: "For all ticks of clock `a` with property X, clock `b` also has a tick".

Time-triggered implications. These properties are what makes TESL special. They are of the form: "For all ticks of clock `a`, clock `b` has a tick after a delay Δt measured on the time scale of clock `c`". The clock `c` does not need to have ticks: it only serves as a time-measuring reference.

Tag relations. For time-triggered implications to make sense, we need relations between time scales of different clocks: this is the purpose of tag relations (e.g., a typical use will be: "Time on clock `a` runs 2 times faster than on clock `b`").

A small example of time and causalities in TESL is given below:

```
1 rational-clock m1 sporadic 1, 2
2 rational-clock m2
3 tag relation m1 = 2 * m2 + 0
4 m1 delayed by 1 on m1 implies m2
5 m1 time delayed by 1.5 on m2 implies m2
```

Lines 1 and 2 define two clocks `m1` and `m2` with tags valued in rationals. The constraint `sporadic` imposes *at least* two ticks on `m1` whose tags are 1 and 2. Line 3 specifies that time on `m1` goes twice as fast as on `m2`. Finally, Line 4 specifies to count 1 tick on `m1` before requiring one on `m2`. Whereas on Line 5, for each tick on `m1`, there is a tick on `m2` 1.5 units of time (measured on `m2`) later.

The specification does not impose any *maximal* number of ticks: they are only required to satisfy the set of constraints described in the specification. Two satisfying runs are given in Figure 1 where ticks (solid rectangles) necessarily have tags (small numbers above right) and are partitioned by a sequence of coincidence instants $(I_n)_{n\geq 0}$ (dashed rectangles). The run on the right side has more ticks than required, but keeps satisfying the specification.

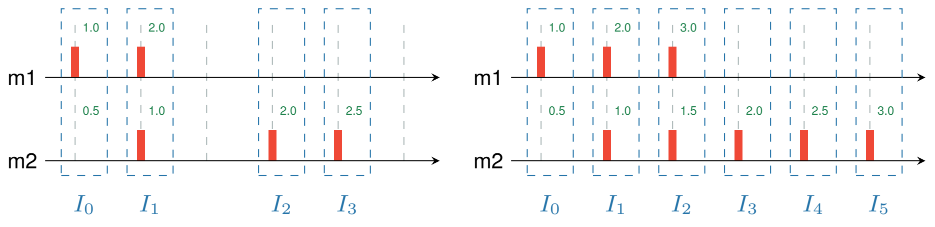

Fig. 1. Two satisfying runs

4 Formal approach

Our main purpose is the understanding of the meaning of a TESL specification made of entities such as clocks, ticks, tags and relations between these tags. The desired semantics consists in a set of all possible execution traces (runs) of the specified clocks and whose tags match the constraints. Currently, the TESL simulator takes as input a specification, and outputs only one run, empirically designed, that is suitable for the execution of heterogeneous models where events occur as *soon* as possible.

The goal of the current work is threefold :

1. Give a machine-checkable formalization of TESL semantics,
2. Show the correctness of the existing simulator algorithm w.r.t. the semantics,
3. Derive properties over the simulator runs: time complexity, compactness, . . .

The formalization relies on the Isabelle/HOL proof assistant with several expected benefits : (1) formally characterize what a correct run is, (2) prove properties on runs, (3) derive the simulator from the formal semantics, and (4) generate tests, especially with the HOL-TestGen framework [6].

The difficulty steams from several aspects : (1) the lack of formal definitions of heterogeneous time and causality models, (2) the non-trivial nature of such entities, and (3) the need to prove correctness of the TESL informal simulation algorithm with a machine-checkable formalization.

In order to make sure to capture most of the aspects of the problem, we chose to work on several parallel, independent representations of the semantics. Each has its strengths and weaknesses. Formalizing them and then exploring their relations will help us to understand real issues.

4.1 Axiomatic semantics

Our first approach to give a formal semantics for TESL runs starts with an axiomatic semantics based on first-order logic. We give a logical formula for each atomic TESL specification operator standing for the expected correct behavior. In particular, we observe that such a framework does not allow constructiveness and cannot derive runs for the satisfiability problem. We notice that the unbounded μ-operator[2] becomes largely useful for non-trivial TESL specifications.

The axiomatic semantics is given by a correctness assessment rule for each atomic specification and a run is said to *satisfy* the specification when all atomic rules hold. We shall denote $\Uparrow_r^c$ as a predicate satisfied when clock c *is ticking* at instant index r. For instance, in the case of the `delayed` implication on Line 4 in the listing page 3, both runs in Fig. 1 do satisfy the following rule. We check that each time `m1` ticks then the next tick on `m1` will trigger a tick on `m2`:

$$\forall \text{ instant index } r \quad \text{s.t.} \quad \Uparrow_r^{\texttt{m1}}, \quad \forall r' > r \quad \text{s.t.} \quad r' = \mu x_{r<x}.\left[\Uparrow_x^{\texttt{m1}}\right], \quad \text{we have } \Uparrow_{r'}^{\texttt{m2}}$$

A more complete version is available online as a research report[3].

[2] $\mu x_{n<x}.\varphi(x)$ is the smallest integer – greater that n – such that $\varphi(x)$ is satisfied.
[3] URL: `http://wwwdi.supelec.fr/software/downloads/TESL/axiomatic_semantics_tesl.pdf`

4.2 Trace compositional semantics

In this second approach, we want to construct all possible runs for a specification, using executable semantic definitions. For now, we consider only finite traces.

A first solution consists in generating all possible interleavings of the clocks in the specification, and then to filter out those which do not satisfy the constraints of the specification. This solution has the advantage of giving a simple to understand specification of each TESL operator, however, it generates a huge amount of potential runs before keeping only the ones that satisfy the constraints.

A second solution consists in generating only the interleavings that satisfy the constraints. This is much more efficient, but the correctness of the algorithm has to be proved by showing that the generated runs are the same as those obtained with the first solution.

Although it is executable, this formal specification can only check that some interleavings of clocks satisfy a specification. It is not yet able to add ticks to the clocks so that all correct interleavings are generated. This formal specification is available online[4].

4.3 Algebraic semantics

The last approach we follow, aims to separate time and causality entities between mathematical n-cells. Distinguishing entities this way tends to give a complete generic semantics, in which only basic mathematical structures are used in a *symmetric* setting [1], as depicted below.

Hierarchy	Objects	Time entities
1-cells	posets	events, tags
2-cells	monotonic functions	clocks
3-cells	morphisms	clock relations

In fact, clocks are seen as *monotonic relations* over partially ordered sets (aka *posets*): where *order* stands for time flow. The intuition is that a clock in TESL is not just a (time) ordered set of events: each event is also timestamped with a tag (valued in an ordered set). Therefore, a clock is an (ordered) set of pairs – each consisting of the instant at which the event occurs together with its tag – *i.e.* a monotonic relation of $\mathsf{Events} \times \mathsf{Tags}$.

5 Comparison of the semantics

Following Cousot [7], we wish to construct and relate a hierarchy of semantics, in order to provide a general understanding of the TESL semantics.

First, the axiomatic semantics gives rules, that are close to the concrete system, but lack constructiveness. The occurrence of unbounded μ-operators and

[4] URL: `http://wwwdi.supelec.fr/software/TESL/Isabelle#TraceComp`

the non-restriction of first-order logic are one of the reasons why such semantics would not be executable.

The algebraic semantics is an attempt to go further by classifying entities in n-cells and focusing on posets. This semantics would still not be executable, but it would give a generic understanding of entities at a larger scale than TESL and help define a general *clock calculus*.

Finally, the trace compositional semantics can be seen as a concrete instance of the previous approach, which allows generating runs. The management of large execution traces is a challenge, as exponential time and space are required to store intermediate computations. We address this issue with a monadic approach. While the adequacy of this last semantics to the original system is not straightforward, we expect it can be related to the other, more direct approaches.

6 Conclusion

The TESL language allows specifying timed systems whose components obey complex synchronization patterns. It is relatively easy to give to such a language an abstract, constraint-based semantics. However, in order to be able to compute on the semantics and generate test-cases, we have to derive far more involved semantics artifacts. Relating these approaches from the closest to the original TESL to the most fitted for implementation, would give us usable and trustable tools to specify complex timed systems.

References

1. S. Abramsky and A. Jung. Domain theory. In *Handbook of Logic in Computer Science*, pages 1–168. Clarendon Press, 1994.
2. A. Benveniste, B. Caillaud, L. P. Carloni, P. Caspi, and A. L. Sangiovanni-Vincentelli. Composing heterogeneous reactive systems. *ACM Trans. Embed. Comput. Syst.*, 7(4):43:1–43:36, Aug. 2008.
3. G. Berry. The constructive semantics of pure esterel. 1996.
4. F. Boulanger, C. Jacquet, C. Hardebolle, and A. Dogui. Heterogeneous Model Composition in ModHel'X: the Power Window Case Study. In *Workshop on the Globalization of Modeling Languages at MODELS 2013*, Miami, USA, Sept. 2013.
5. F. Boulanger, C. Jacquet, C. Hardebolle, and I. Prodan. Tesl: A language for reconciling heterogeneous execution traces. In *Formal Methods and Models for Codesign (MEMOCODE), 2014 Twelfth ACM/IEEE Intl Conf on*, pages 114–123, Oct 2014.
6. A. Brucker and B. Wolff. On theorem prover-based testing. *Formal Aspects of Computing*, 25(5):683–721, 2013.
7. P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoretical Computer Science*, 277(1–2):47 – 103, 2002. Static Analysis.
8. E. Lee, A. Sangiovanni-Vincentelli, et al. A framework for comparing models of computation. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 17(12):1217–1229, 1998.
9. F. Mallet. Clock constraint specification language: specifying clock constraints with Uml/Marte. *Innovations in Systems and Software Engineering*, 4(3):309–314, 2008.