An Interface Theory for Timed Systems

Alexandre David[1], Kim. G. Larsen[1], Axel Legay[2],
Ulrik Nyman[1], Andrzej Wąsowski[3]

[1] Computer Science, Aalborg University, Denmark
[2] INRIA/IRISA, Rennes Cedex, France
[3] IT University, Copenhagen, Denmark

1 The subject

Contemporary IT systems are assembled out of multiple independently developed components. Component providers operate under a contract on what the interface of each component is. Interfaces are typically described using textual documents or models in languages such as UML or WSDL. Unfortunately, such specifications are subject to interpretation. To avoid the risk of ambiguity, we recommend mathematically sound formalisms, such as interface theories, whenever possible. A good interface theory supports *refinement checking* (whether an interface can be replaced by another one), *satisfaction checking* (whether an implementation satisfies the requirements expressed with the interface), *consistency checking* (whether the interface can be implemented), a *composition operator* (structurally combining interfaces), a *conjunction operator* (computing a specification whose implementations are satisfying both operands), and a *quotient operation* that is the adjoint for composition. It should also guarantee important properties such as independent implementability [9].

Recently, building good interface theories has been the subject of intensive studies (see e.g., [13, 8, 2, 12, 7, 10]). It has been argued [6, 9] that *games* constitute a natural model for interface theories: each component is represented by an automaton whose transitions are typed with *input* and *output* modalities. The semantics of such an automaton is given by a two-player game: the *input* player represents the environment, and the *output* player represents the component. Contrary to the input/output model proposed by Lynch [14], this semantic offers (among many other advantages) an optimistic treatment of composition (two interfaces can be composed if there exists at least one environment in which they can interact together in a safe way) and refinement (the refined system should accepts at least the same inputs and not produce more outputs). Game-based interfaces were first developed for *untimed systems* [9, 7] and the composition and refinement operations were implemented in tools such as TICC [1] or CHIC [4].

Existing results on game-based interfaces are for untimed systems. However, time can be a crucial parameter in practice, *e.g.* in embedded–system applications. We have recently proposed the first complete *timed* interface theory based on timed games [5]. The idea is similar to the untimed case: components are modeled using timed input/output automata (TIOAs) with a timed game

semantics [3]. Our theory is rich in the sense that it captures all the good operations for a compositional design theory. Reporting on timed interfaces is the main subject of this presentation. In our talk, we will discuss:

- the concept of interface theories, the mathematical requirements, and the game-based approach;
- the theory introduced in [5];
- our new tool ECDAR [11], which implements the theory of [5].ECDAR is the the first complete game-based tool for timed interfaces in the dense time setting. ECDAR implements checkers such as satisfaction/consistency, refinement, and satisfaction of TCTL formulas. The tool also supports the classical compositional reasoning operations of conjunction and composition. To the best of our knowledge, ECDAR is the first tool to propose an implementation of quotient (that we will discuss intensively). In addition, it comes with a user-friendly interface, where errors are reported in an intelligible way.
- Some applications to illustrate the use ECDAR and the advantages of compositional design.

References

1. B. T. Adler, L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, V. Raman, and P. Roy. Ticc: A tool for interface compatibility and composition. In *CAV*, volume 4144 of *LNCS*, pages 59–62. Springer, 2006.
2. S. Bliudze and J. Sifakis. A notion of glue expressiveness for component-based systems. In *CONCUR'08*, volume 5201 of *Lecture Notes in Computer Science*, pages 508–522. Springer, 2008.
3. F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR*, 2005.
4. Chic, 2003. `http://www-cad.eecs.berkeley.edu/~tah/chic/`.
5. A. David, K. Larsen, A. Legay, U. Nyman, and A. Wąsowski. Timed I/O automata: a complete specification theory for real-time systems. In *HSCC*, 2010. Accepted.
6. L. de Alfaro. Game models for open systems. In *Verif (Theory in Practice)*, volume 2772 of *LNCS*. Springer, 2003.
7. L. de Alfaro, L. D. da Silva, M. Faella, A. Legay, P. Roy, and M. Sorea. Sociable interfaces. In *FroCos*, volume 3717 of *lncs*, pages 81–105. Springer, 2005.
8. L. de Alfaro and T. A. Henzinger. Interface automata. In *FSE'01*, pages 109–120. ACM Press, 2001.
9. L. de Alfaro and T. A. Henzinger. Interface-based design. In *Marktoberdorf Summer School*. Kluwer Academic Publishers, 2004.
10. L. de Alfaro, T. A. Henzinger, and M. I. A. Stoelinga. Timed interfaces. In *EMSOFT*, volume 2491 of *LNCS*, pages 108–122. Springer, 2002.
11. `http://www.cs.aau.dk/~adavid/ecdar/`.
12. J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity - the ptolemy approach. *Proc. of the IEEE*, 91(1):127–144, 2003.
13. T. A. Henzinger and J. Sifakis. The embedded systems design challenge. In *FM*, volume 4085 of *LNCS*, pages 1–15. Springer, 2006.
14. N. A. Lynch and M. R. Tuttle. An introduction to input/output automata. Technical Report MIT/LCS/TM-373, The MIT Press, Nov. 1988.