Theoretical Computer Science 549 (2014) 146–174

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

An algebraic theory of interface automata

Chris Chilton [a], Bengt Jonsson [b], Marta Kwiatkowska [a,*]

[a] Department of Computer Science, University of Oxford, UK
[b] Department of Information Technology, Uppsala University, Sweden

ARTICLE INFO

Article history:
Received 27 February 2013
Received in revised form 20 April 2014
Accepted 18 July 2014
Available online 28 July 2014
Communicated by V. Sassone

Keywords:
Component-based design
Interfaces
Specification theory
Compositionality
Refinement
Substitutivity
Synthesis

ABSTRACT

We formulate a compositional specification theory for interface automata, where a component model specifies the allowed sequences of input and output interactions with the environment. A trace-based linear-time refinement is provided, which is the weakest preorder preserving substitutivity of components, and is weaker than the classical alternating simulation defined on interface automata. Since our refinement allows a component to be refined by refusing to produce any output, we also define a refinement relation that guarantees safety and progress. The theory includes the operations of parallel composition to support the structural composition of components, logical conjunction and disjunction for independent development, hiding to support abstraction of interfaces, and quotient for incremental synthesis of components. Our component formulation highlights the algebraic properties of the specification theory for both refinement preorders, and is shown to be fully abstract with respect to observation of communication mismatches. Examples of independent and incremental component development are provided.

1. Introduction

The interface automata of de Alfaro and Henzinger [11] are an influential formalism for modelling the interactions between components and their environment. Components are assumed to communicate by synchronisation of input and output (I/O) actions, with the understanding that outputs are non-blocking. If an output is issued when a component is unwilling to receive it, a communication mismatch is said to occur. This allows one to reason about the allowed behaviours of the environment, which is crucial in assume-guarantee reasoning, for example.

An important paradigm for developing complex reactive systems is component-based design, which can be underpinned by a specification theory. A specification captures the requirements for a component to function in an intended system context, while operators and refinement relations allow for the composing and comparing of specifications in analogy with how components are composed and refined towards an overall system design. Substitutive refinement is essential for dynamic systems of components in this respect, as it allows for the replacement of components without introducing errors to the system.

The original theory of interface automata defines a substitutive refinement in terms of alternating simulation [2], along with a parallel composition operator for observing component interaction. In subsequent papers, variants of the framework have been extended with additional operators, including conjunction (defined by Doyen et al. for synchronous automata [15]) and quotient for supporting incremental development (defined by Bhaduri and Ramesh for deterministic automata [4]).

* Corresponding author.

http://dx.doi.org/10.1016/j.tcs.2014.07.018
0304-3975/

In this article, we formulate a theory for components that is conceptually similar to interface automata, but is based on a linear-time notion of substitutive refinement involving trace containment. We define a specification theory for component behaviours, which includes the operations of: *parallel composition* for structural composition of components; *conjunction* for supporting independent development, by constructing a component that will work in any environment compatible with at least one of its arguments; *disjunction* for constructing a component that has an environment compatible for both of its operands; *hiding* to support abstraction in hierarchical development; and *quotient* for incrementally synthesising new components to satisfy partial requirements. We prove compositionality for all the operations and show that the specification theory enjoys strong algebraic properties.

Our formalism addresses the following shortcomings of the interface automata theory as formulated by de Alfaro and Henzinger in [11]:

- Alternating simulation is conceptually more complex than refinement based on trace containment, which is standard in widely used theories such as CSP [5] and I/O automata [27,20]. Further, alternating simulation is overly strong in comparison to our refinement based on traces, which is the weakest preorder preserving compatibility with the environment.
- It is not clear how to extend a refinement relation based on alternating simulation so that it also preserves liveness properties. This should be contrasted with the conceptually simple handling of liveness properties in formalisms such as I/O automata, which use trace inclusion. In the case of our refinement, we are able to extend it with the notion of quiescence to guarantee observational progress, in addition to substitutivity. We prove that compositionality results for all the operations continue to hold for this enhanced refinement.

The contribution of this article is therefore a compositional, linear-time specification theory for interface automata based on fully abstract substitutive and progress-preserving refinement. Our framework includes all desirable operations on components known from the literature, and satisfies strong algebraic properties, including the characterisation of conjunction and disjunction, respectively, as the meet and join of the refinement preorder. The theory naturally supports a component-based design process that starts with initial design considerations, from which the operations of the theory are applied compositionally in a stepwise fashion, relying on substitutivity to guarantee that no errors will be introduced, even if components are refined at runtime.

A preliminary version of this article appeared as [6], where we introduced the operations of parallel, conjunction and quotient, but did not consider an extension with quiescence. To demonstrate the applicability of the theory to component-based design, the quotient operation was used to synthesise mediator components in [18] and [3]. Furthermore, the flexibility and expressiveness of the theory has been shown through a compositional assume-guarantee reasoning framework [8,9] and a real-time extension [10,7].

1.1. Related work

Interface automata The models in this article are conceptually similar to interface automata [11], which are essentially finite state automata with I/O distinction on actions. A key difference is that our refinement preorder is a linear-time alternative to the alternating simulation of Alur et al. [2] defined on interface automata. Both refinements are substitutive, but alternating simulation is overly strong due to the conflict between non-determinism in the automaton and the selection of a matching transition to complete the simulation. Effectively, our notion of parallel composition is the same as for interface automata, except that we encode inconsistency due to communication mismatches explicitly in the model. To the best of our knowledge, conjunction and disjunction have not been defined on interface automata, although Doyen et al. [15] define conjunction (called shared refinement) on a synchronous component model. A definition of quotient has been provided for deterministic interface automata by Bhaduri and Ramesh [4], which mirrors the method developed by Verhoeff [40].

I/O automata Due to Lynch and Tuttle [27], and Jonsson [20], I/O automata are highly similar to interface automata, except that each state is required to be input-enabled. This input receptiveness means that communication mismatches cannot arise between a component and its environment. Consequently, substitutive refinement can be cast in terms of trace containment [20]. The operation of parallel composition is defined in the same way as for interface automata, except that consideration need not be given to inconsistencies. Conjunction can be defined as a synchronous product, meaning that its set of traces is the intersection of its operands' traces. Disjunction can be defined similarly. Hiding is already defined on outputs by Jonsson [20], and quotient can be defined in a straightforward manner as demonstrated by Drissi and von Bochmann [16].

We mention a process-algebraic characterisation of I/O automata due to de Nicola and Segala [13], which is also applicable to interface automata, since a process exhibits chaotic behaviour on receiving a non-enabled input. Refinement is defined by trace inclusion, but this does not extend to inconsistent trace containment. Consequently, the theory is not able to distinguish a non-enabled input from one that is enabled and can subsequently behave chaotically. Furthermore, high-level operations such as conjunction and quotient are not defined. Note that the CCS of Milner [28] merely has a syntactic distinction of inputs from outputs, so we give it no further attention.