Soundness of a Reasoning System for Asynchronous Communication with Futures

Crystal Chang Din and Olaf Owe

Department of Informatics, University of Oslo, Norway
{crystald, olaf}@ifi.uio.no

Distributed systems play an essential role in society today. For example, distributed systems form the basis for critical infrastructure in different domains such as finance, medicine, aeronautics, telephony, and Internet services. It is of great importance that such systems work properly. However, quality assurance of distributed systems is non-trivial since they depend on unpredictable factors, such as different processing speeds of independent components. It is highly challenging to test such distributed systems after deployment under different relevant conditions. These challenges motivate frameworks combining precise modeling and analysis with suitable tool support. In particular, compositional verification systems allow the different components to be analyzed independently from their surrounding components. Thereby, it is possible to deal with systems consisting of many components.

Object orientation is the leading framework for concurrent and distributed systems, recommended by the RM-ODP [12]. However, method-based communication between concurrent units may cause busy-waiting, as in the case of remote and synchronous method invocation, e.g., Java RMI [2]. Concurrent objects communicating by asynchronous method calls, which allows the caller to continue with its own activity without blocking while waiting for the reply, combine object-orientation and distribution in a natural manner, and therefore appears as a promising paradigm for distributed systems. Moreover, the notion of futures [4, 15, 10, 16] improves this setting by providing a decoupling of the process invoking a method and the process reading the returned value. By sharing future identities, the caller enables other objects to wait for method results.

ABS is a high-level imperative object-oriented modeling language, based on the concurrency and synchronization model of Creol [14]. It supports futures and concurrent objects with an asynchronous communication model suitable for loosely coupled objects in a distributed setting. In ABS, each concurrent object encapsulates its own state and processor, and internal interference is avoided as at most one process is executing. The concurrent object model of ABS without futures supports compositionality because there is no direct access to the internal state variables of other objects, and a method call leads to a new process on the called object. With futures, compositionality is more challenging.

In this paper, we focus on the communication model of ABS with futures but ignore the aspects of inheritance and object creation. A compositional reasoning system for ABS with futures has been presented in [9] based on local communication histories. We here show that this system is sound with respect to a revised version of the operational semantics of ABS, which incorporates a notion of global communication history.

The execution of a distributed system can be represented by its communication history or trace; i.e., the sequence of observable communication events between system components [9]. At any point in time the communication history abstractly captures the system state [7]. In fact, traces are used in the semantics for full abstraction results (e.g., [13]). The local history of an object reflects the communication visible to that object, i.e., between the object and its surroundings. A system may be specified by the finite initial segments of its communication histories, and a history invariant is a predicate which holds for all finite sequences in the set of
possible histories, expressing safety properties \cite{3}.

In our reasoning system, we formalize object communication by an operational semantics based on four kinds of communication events \cite{8}, capturing shared futures, where each event is visible to only one object as shown in Fig. 1. Consequently, the local histories of two different objects share no common events. For each object, a history invariant can be derived from the class invariant by hiding the local state of the object. Modularity is achieved since history invariants can be established independently for each object, without interference, and composed at need. This results in behavioral specifications of dynamic system in an open environment. Such specifications allow objects to be specified independently of their internal implementation details, such as the internal state variables. In order to derive a global specification of a system composed of several components, one may compose the specification of different components. Global specifications can then be provided by describing the observable communication history between each component and its environment.

![Figure 1: A method call cycle: object $o$ calls a method $m$ on object $o'$ with future $u$. The four kinds of events are indicated by $\rightarrow$, $\rightarrow^*$, $\leftarrow$, and $\leftarrow^*$, indicating method invocation, method initiation, writing to future and reading from future, respectively. The events on the left-hand side are visible to $o$, those in the middle are visible to $o'$, and the ones on the right-hand side are visible to $o''$. There is an arbitrary delay between message receiving and reaction.](image)

The main contribution of this paper is the proof of soundness with respect to the revised operational semantics including a global communication history. The operational semantics is implemented in Maude by rewriting rules and can be exploited as an executable interpreter for the language, such that execution traces can be automatically generated while simulating programs. In earlier work \cite{8, 9}, a similar proof system is derived from a standard sequential language by means of a syntactic encoding. However, soundness with respect to the operations semantics was not considered. A challenge of the current work is that the presence of shared futures complicate compositional reasoning and also the soundness proof. We therefore focus the current work on the ABS communication model with futures, and ignore other aspects such as object creation.

An ABS reasoning system is currently being implemented within the KeY framework at Technical University Darmstadt. With the tool support from KeY for (semi-)automatic verification, it will be valuable to verify larger ABS programs. An elevator system has been implemented, simulated and tested according to the generated histories. The elevator system will be used here to illustrate the language and the reasoning system.
References


