Operational Semantics for Agents: the Grey-Box Modeling Approach

Federico Bergenti
AOT Lab, Università di Parma
43100 Parma, Italy
bergenti@ce.unipr.it

Giovanni Rimassa
AOT Lab, Università di Parma
43100 Parma, Italy
rimassa@ce.unipr.it

Mirko Viroli
DEIS, Università di Bologna
47023 Cesena (FC), Italy
mviroli@deis.unibo.it

Categories and Subject Descriptors
D.2.10 [Software Engineering]: Design; I.2.11 [Distributed Artificial Intelligence]: Multi-agent Systems

General Terms
Languages, Design

Keywords
Agent Design, Agent Communication Languages, Transition Systems

1. INTRODUCTION

We study the issue of sound multi-agent design trying to keep a system engineering perspective; that is, both when analyzing a problem and when synthesizing a solution the focus is kept on the whole, large-scale structure of the software artifact that is to be realized.

The formal tool we evaluate to provide effective assistance to the sound design of multi-agent systems (MASs) applies (labeled) transition systems to the description of software interactive behavior [3], as elaborated and promoted in the field of concurrency theory. When specifying a transition system semantics for an agent, a number of rules are given that describe in a quite declarative way the dynamics of its inner machinery, also providing insights on its internal architecture – expressed at a given level of abstraction. We believe that this formalism is fit for design because of its calculus-like nature, which combines the description of a system with the prescription of its possible evolutions; this property buys designers some generativity, while remaining in a well grounded mathematical landscape. Moreover, transition systems allow us to address both compositability and extensibility. An agent model is divided into several LTS specifications that we call tiers, capturing different aspects of an agent design. A partial specification of an agent can be further detailed by adding new tiers, with a technique known as grey-boxing [4, 5].

As a use case for our approach we consider ParADE [2], a development environment exploiting the formal semantics of a FIPA-like Agent Communication Language (ACL) to gain agents semantic interoperability, and providing interaction laws as a design mechanism to specify the agent social role.

2. TRANSITION SYSTEMS

Whereas LTS originated in the field of concurrency theory to provide an operational semantics for process algebras, they are here used in a slightly different, yet not uncommon fashion: instead of focusing on the semantics of a language or algebra, we apply them to the description of the interactive behavior of a system. As usual, a LTS is a triple \((X, \rightarrow, \text{Act})\), where \(X\) is the set of states of the system of interest, \(\text{Act}\) is the set of actions, and \(\rightarrow \subseteq X \times \text{Act} \times X\) is the transition relation, describing how the state evolves by way of actions.

Several remarkable concepts are promoted by LTS, such as the notions of system observation and refinement of specifications. In particular, a whole system can be characterized in terms of all its admissible evolutions, e.g. expressed as traces of actions. The notion of refinement then naturally comes in: a system description is considered an “implementation” of another – i.e., it is more directly executable – if it is more deterministic, that is, if it allows for strictly fewer traces [3]. This notion is particularly crucial in the context of software engineering, since it guarantees that by substituting a component with a refinement of it, the whole system dynamics keeps satisfying the same general rules, namely, is a refinement of the initial system.

3. GREY-BOX AND REFINEMENT

In [5], the rationale behind the grey-box modeling approach for agents is introduced. Its key idea is to represent an agent behavior by focusing on its part dealing with the interactions with the environment (agent core), while abstracting away the complex inner details (agent internal machinery).

Formally, an agent core can be defined by a mathematical structure \(\mathcal{A} = (I, O, P, X, E, U, \rightarrow, \text{Act})\), where \(I\) and \(O\) are input and output acts (either physical or communicative), \(P\) is the set of states that can be observed by the agent internal machinery, \(X\) is the set of states of the remaining part of the agent core, \(E\) is the set of events occurring within the agent core and possibly affecting the internal machinery, and \(U\) is the set of updates notified by the internal machinery to the agent core, modeling internal happenings of interest. A transition relation \(\rightarrow\) can be defined that describe how the agent core state evolves as actions with the syntax \(\text{Act}_A := \tau \mid ?i \mid !o \mid \triangleright e \mid \triangleleft u\), where \(\tau\) is the internal computation, the agent listening act \(i\), the agent executing act \(o\), the event \(e\) occurring, and the update \(u\) being notified by the internal machinery. The transition system \((A, \rightarrow, \text{Act}_A)\) is indeed a grey-box model of...
an agent behavior, focusing on its description according to the abstraction level of interest.

However, it is common practice of system analysis and design to start considering a system at a high abstraction level, which is later lowered in order to consider much more details. This is realized by composing the specification $A$ with a new specification $B$ called completion, which represents a new agent tier, so that the resulting system $C = A \otimes B$ has still the structure of an agent core specification likewise $A$—hence it still a grey-box model—, but it also provides a number of additional details about the agent behavior. Formally, $B$ is a structure of the kind $(Y, Z, F, V, \rightarrow_B)$, where $W = Y \times Z$ is the set of local states added by $B$ to the core state $Y$ so that the observable part, $F \subseteq E$ and $V \subseteq U$ are respectively the new events and updates by which $C$ interacts with its internal machinery. A transition relation $\rightarrow_B$ analogous to $\rightarrow_A$ can be given describing the behavior of the completion.

Then, given models $A$ and $B$, $C = A \otimes B$ is defined as the agent core specification $(I, O, P \times Y, X \times Z, F, V, \rightarrow_C)$. The relation $\rightarrow_C$ is obtained by just composing the specifications of $\rightarrow_A$ and $\rightarrow_B$, in a similar way as done in [4], which is here avoided for brevity. A pictorial representation of our composition technique is shown in Figure 1. Following again the general approach of [4], it is possible to prove that agent core $C$ has a more refined behavior than agent core $A$ in the sense specified by trace semantics [3]. Hence, our refinement technique can indeed be considered as a way of deepening a specification towards implementation issues.

4. A SPECIFICATION FOR ParADE

ParADE, the Parma Agent Development Environment [2], is a development framework that provides the agent developer with high-level abstractions like beliefs and goals. It implements a hybrid architecture capable of supporting autonomy and intelligent behaviors by exploiting an ACL—resembling the FIPA ACL but with a much simpler semantics—, so that semantics is attached to the agent utterances. Still, an agent is not considered within an environment that is not limited to the social milieu; there are actions to perform and events to perceive that are not linguistic, but which belong to the $domain$ of the discourse, i.e. the external environment itself, and are typically described within a $domain$ model or ontology. Then, in order to support social fruitful communication, the ParADE agent model provides interaction laws, which an agent decides to adopt to govern its interactions with other agents. The ACL semantics, the domain model, and the interaction laws work together to expose an observable model of a ParADE agent that enables semantically meaningful communication with others [1].

So, an agent adopting the ParADE model is suitable for a characterization in our framework by considering different tiers, respectively: (i) the interaction tier, managing interactions with the environment, (ii) the linguistic tier, dealing with aspects related to the ACL semantics, (iii) the domain tier, concerning the specific ontology of the application, (iv) the social tier, where interaction laws define the role and public peculiarities of the agent, and finally (v) the internal tier, dealing with other aspects such as private laws, planning, and proactiveness. The order of such tiers reflects their impact on the external, observable behavior of the agent, from tackling interaction issues to considering internal reasoning.

The starting agent core considered here is that forming the interaction tier, keeping track of pending input and output acts and of the agent mental states (beliefs, desires, intentions, and so on). There, events represent listening of input acts to be notified to the internal machinery, and updates are either requests for performing output acts or for changing the mental state. This basic specification can be refined by taking into account the linguistic tier, dealing with all the aspects related to semantics of communications. In order to grasp the flavor of the specification style of our methodology, consider the following rules, which describe some aspect of the linguistic tier:

\begin{align*}
\downarrow_{\Phi_A} & \rightarrow_B \text{ put}(\text{ac}^F(\text{ic})) \quad \text{if } \text{ic} \in I_{ACL} \\
\downarrow_{\Phi_A} & \rightarrow_B \text{ put}(\text{ac}^F(\text{ic})) \quad \text{otherwise} \\
\downarrow_{\Phi_A} & \rightarrow_B \text{ put}(\text{ac}^F(\text{ic})) \quad \text{if } \text{oc} \in O_{ACL} \\
\downarrow_{\Phi_A} & \rightarrow_B \text{ put}(\text{ac}^F(\text{ic})) \quad \text{otherwise}
\end{align*}

Orderly: when a message is received either (i) the rational effects prescribed by the ACL are applied or (ii) a failure message is scheduled which is later sent out (iii); requests for sending messages are either (iv) processed if ACL preconditions hold, or (v) ignored otherwise.

A complete specification of all the tiers for the ParADE agent model is given in the full version of the paper, which can be obtained from the URL: http://www.ingce.unibo.it/~mviroli.

5. REFERENCES


