Multi-Level Semantics with Vertical Integrity Constraints

Alison R. Panisson¹ and Rafael H. Bordini¹ and Antônio Carlos da Rocha Costa²

1 Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil — Email: alison.panisson@acad.pucrs.br, rafael.bordini@pucrs.br
2 Graduate Program in Computers in Education – UFRGS, Porto Alegre, Brazil and Graduate Program in Computer Science – FURG, Rio Grande, Brazil — Email: ac.rocha.costa@gmail.com

Abstract. Operational semantics is a fundamental approach to the formalisation of programming languages and almost a standard when it comes to agent-oriented programming languages. It helps ensure the correctness of interpreters, facilitates their implementation, and supports proofs of important properties. Multi-agent oriented systems are a particular kind of distributed systems and through the semantics of agent languages, operational semantics ended up playing an important role towards ensuring their desired behaviour, even though the operational semantics becomes more involved than originally intended. This work presents a new style for the operational semantics of systems with multiple levels of abstractions (such as multi-agent systems), by providing multi-level transitions (i.e., multiple hierarchical transition systems) with vertical (i.e., inter-level) integrity constraints to ensure consistency of interrelated transitions.

1 Introduction

In multi-agent systems (MAS) there are multiple different levels of specification, each one corresponding to a different conceptual level in the system, and playing important roles in a more general framework for programming MAS (e.g., JaCaMo Framework [1], PopOrg (Populational-Organisational) Model [2], Electronic Institutions [3], etc.). In order to ensure the desired behaviour of MAS, it has been common in the Agents literature to give formal semantics to such systems as transition systems [8, 5, 4, 6] (using operational semantics [7]), expressing the possible states of the system and the necessary conditions for the system to move from one such state to another.

Given the complexity of multi-agent system abstractions, we propose a multi-level operational semantics with vertical (i.e., inter-level) integrity constraints in order to specify those systems. Such vertical integrity constraints aim to ensure that the transition systems give separate semantics to the individual levels of abstraction are combined in a way that preserves the required interrelations of those levels. Furthermore, they allow some of the required semantic rules to be automatically generated from compact representations of such integrity constraints. In this work, we focus on the functioning of multi-agent organisations, modelling them as multiple, independent but interrelated, transition systems. In particular, we express the specification in terms of count-as relations between levels, i.e., relations that express certain combinations of actions — i.e., processes — executed at a given abstraction level count as actions being executed at an upper level.

2 Vertical Integrity Constraints

In order to exemplify our work, here we focus on processes (a ordering of actions/events) which occurs at a level of abstraction and count-as actions/events on superior levels. In particular, here we consider three levels in the multi-agent system, the agents, the organisations, and the social sub-system, so we introduce two vertical integrity constraints:

$$\#VIC_{ag} \uparrow [AS_{org}, A_{org}, AS_{ag}, Act_{ag}] = S$$ (1)

being $AS_{org}$ the history of the organisational actions already executed, $A_{org}$ the set of organisational actions yet to be executed (i.e., that the organisation aims to see executed), $AS_{ag}$ the history of actions executed by the agents in that organisation, $Act_{ag}$ the set of agent actions being executed at that moment, and $S$ the set of organisational actions that can be considered as executed given that history of agents actions ($AS_{ag}$ and $Act_{ag}$) and the count-as relations.

$$\#VIC_{ss} \uparrow [AS_{ss}, A_{ss}, AS_{org}, Act_{org}] = S$$ (2)

being $AS_{ss}$ the history of the social sub-system actions, $A_{ss}$ the set of social sub-system actions that were not executed, $AS_{org}$ the history of actions executed by the organisations in that social sub-system, $Act_{org}$ the set of organisational action being executed at that moment, and $S$ the set of social sub-system actions that can be considered as executed given that history of organisational actions ($AS_{org}$ and $Act_{org}$) and the count-as relations.

3 Abstract Semantic Rule

The definition of multi-level semantics with vertical integrity constraints between the levels allows just one abstract semantic rule interpreting all count-as relations in the multi-agent system, where the corresponding actions can be instantiated to particular cases. The abstract semantic rule is showed below³:

$$execute(act) \quad act \in A$$

$$\#VIC_{ag} \uparrow [AS_{org}, A_{org}, AS_{ag}, \{act\}] = S_{org}$$

$$\#VIC_{ss} \uparrow [AS_{ss}, A_{ss}, AS_{org}, S_{org}] = s_{org}$$

where:

(a) $AS_{org} = AS_{org} \cup S_{org}$

(b) $A_{org} = a_{org} \cup A_{org}$

(c) $AS_{ag} = AS_{ag} \cup \{act\}$

where the tuple $(s_{id}, \mathcal{O}, A_{ss})$ represents the social sub-system, with $s_{id}$ the social sub-system identifier, $\mathcal{O}$ a set of organisation identifiers, each one representing the organisation populating the social

³ We use the notation “$(a_{g_{1}}, A_{ag_{1}}) \ldots AS_{ag}$” as a simplified representation for “$(a_{g_{1}}, A_{ag_{1}}), \ldots (a_{g_{n}}, A_{ag_{n}}), AS_{ag}$.”
The tuple \((org_{sd}, Ag, A_{org})\) represents an organisation, with \(org_{sd}\) the organisation identifier, \(Ag\) a set of agent identifiers, which are populating that organisation, and \(A_{org}\) the set of organisational actions. The tuple \((ag_{sd}, A)\) represents an agent in the multi-agent system, with \(ag_{sd}\) the agent identifier, and \(A\) the set of actions that the agent is able to execute. At each level of the system (i.e., agents, organisations and social sub-systems), we maintain a history (i.e., trace) of actions already executed at that level, \(AS_{ag}\), \(AS_{org}\), and \(AS_{ss}\), respectively. When an action is executed by an agent, the VICs check if the particular action satisfies some process which count-as an action in the superior level, and update the set of organisational and social sub-system in accordance. Further, we use a continuation function \(c(A, AS, a)\), in order to specify when the action being executed (i.e., \(a\)) is removed or not from the set of actions \(A\), and if other actions are included in the set of actions given the ones that were executed. This function covers both achievement and maintenance tasks in MAS.

4 Example

Imagine the process of exchanging an employee by two organisations \(org_1\) and \(org_2\). On the top of our multi-agent system, we have the social sub-system \(sys\) regulating such exchange through a process for a norm-conforming exchange \(norm\)-conf-exchange, which is achieved by the sequence of the organisational actions \{dismiss
\_org, appoint\_org\}. The appoint\_org action is achieved by the sequential process \{interview, admission, request, admission, processing, appoint\}, and dismiss\_org is achieved by the sequential process \{give notice, dismissal request, dismissal, processing, dismiss\}, where all those are actions executed by agent populating those organisation, and that count-as organisational action when the process is followed.

In this section, we present three instances from the abstract semantic rule, in accordance with the scenario described. The first, ExInterviewAction, shows an agent called interviewer executing the first action of the appoint\_org process. In this semantic rule, as the VICs are not satisfied (none process is satisfied) only the agent level (c) is updated in accordance.

In contrast, when an agent (exemplified by the agent named manager) executes the last action for the process dismiss\_org, i.e., the action dismiss, such organisational action is achieved. This is exemplified by the semantic rule ExDismissAction, where we assume that all other action for such dismiss\_org process have been executed, and therefore the \(\#VIC_{dismiss}^org \uparrow\) is satisfied, i.e., the agents’ actions satisfies a process for dismiss\_org. In this case both, the agent and organisational levels change in accordance.

The last semantic rule ExAssignAction shows when the last action of the norm-conf-exchange process is achieved, i.e., appoint\_org is achieved by the organisation. Further, the appoint\_org organisational action is achieved when the agent named manager executes the last action of such process, i.e., the action appoint. Observe that both VICs are satisfied and all levels in the systems are updated.

5 Conclusion

In this work, we introduced an approach to the formalisation of MAS based on operational semantics; we call it multi-level semantics with vertical integrity constraints. The approach allows the representation of the interactions between components of different system-levels. Given the complexity and ubiquity of such multiple levels in MAS, the approach seems to allow for a clearer understanding of such complex semantics. Furthermore, we demonstrate, using count-as relations applied to processes, how the proposed style for multi-level operational semantics can be strengthened through the definition of vertical integrity constraints. Such multi-level semantics with vertical integrity constraints allows the independent specification of the various levels typical of the MAS, so that each level is formalised through its own transition system, and the vertical integrity constraints between these transition systems help guarantee that the overall system operates coherently in all possible executions.

REFERENCES