ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1708

Multi-Level Semantics with Vertical Integrity Constraints

Alison R. Panisson[1] and **Rafael H. Bordini**[1] and **Antônio Carlos da Rocha Costa**[2]

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 giving 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.

[1] Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil — Emails: 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

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:

$$\#\,VIC^{org}_{ag} \Uparrow [\mathcal{AS}_{org}, \mathcal{A}_{org}, \mathcal{AS}_{ag}, Act_{ag}] = S \qquad (1)$$

being $\mathcal{AS}_{org}$ the history of the organisational actions already executed, $\mathcal{A}_{org}$ the set of organisational actions yet to be executed (i.e., that the organisation aims to see executed), $\mathcal{AS}_{ag}$ 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 ($\mathcal{AS}_{ag}$ and Act_{ag}) and the count-as relations.

$$\#\,VIC^{ss}_{org} \Uparrow [\mathcal{AS}_{ss}, \mathcal{A}_{ss}, \mathcal{AS}_{org}, Act_{org}] = S \qquad (2)$$

being $\mathcal{AS}_{ss}$ the history of the social sub-system actions, $\mathcal{A}_{ss}$ the set of social sub-system actions that were not executed, $\mathcal{AS}_{org}$ 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 ($\mathcal{AS}_{org}$ 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 below[3]:

$$\frac{\begin{array}{c} execute(\texttt{act}) \qquad \texttt{act} \in \mathcal{A} \\ \#\,VIC^{org}_{ag} \Uparrow [\mathcal{AS}_{org}, \mathcal{A}_{org}, \mathcal{AS}_{ag}, \{\texttt{act}\}] = \mathcal{S}_{org} \\ \#\,VIC^{ss}_{org} \Uparrow [\mathcal{AS}_{ss}, \mathcal{A}_{ss}, \mathcal{AS}_{org}, \mathcal{S}_{org}] = \mathcal{S}_{ss} \end{array}}{\begin{array}{llll} (a) & \langle ss_l, \mathcal{O}, \mathcal{A}_{ss}\rangle \ldots \mathcal{AS}_{ss} & \rightarrow_{ss} & \langle ss_l, \mathcal{O}, \mathcal{A}'_{ss}\rangle \ldots \mathcal{AS}'_{ss} \\ (b) & \langle org_n, Ag, \mathcal{A}_{org}\rangle \ldots \mathcal{AS}_{org} & \rightarrow_{org} & \langle org_n, Ag, \mathcal{A}'_{org}\rangle \ldots \mathcal{AS}'_{org} \\ (c) & \langle ag_m, \mathcal{A}\rangle \ldots \mathcal{AS}_{ag} & \rightarrow_{ag} & \langle ag_m, \mathcal{A}'\rangle \ldots \mathcal{AS}'_{ag} \end{array}}$$

$$\begin{array}{llll} \textit{where:} & & & \\ (a) & \mathcal{AS}'_{ss} & = & \mathcal{AS}_{ss} \cup \mathcal{S}_{ss} \\ & \mathcal{A}'_{ss} & = & c(\mathcal{A}_{ss}, \mathcal{AS}_{ss}, \mathcal{S}_{ss}) \\ (b) & \mathcal{AS}'_{org} & = & \mathcal{AS}_{org} \cup \mathcal{S}_{org} \\ & \mathcal{A}'_{org} & = & c(\mathcal{A}_{org}, \mathcal{AS}_{org}, \mathcal{S}_{org}) \\ (c) & \mathcal{AS}'_{ag} & = & \mathcal{AS}_{ag} \cup \{\texttt{act}\} \\ & \mathcal{A}' & = & c(\mathcal{A}, \mathcal{AS}_{ag}, \{\texttt{act}\}) \end{array}$$

(ABSTRACTSEMANTICRULE)

where the tuple $\langle ss_{id}, \mathcal{O}, \mathcal{A}_{ss}\rangle$ represents the social sub-system, with ss_{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 "$\langle ag_m, \mathcal{A}\rangle \ldots \mathcal{AS}_{ag}$" as a simplified representation for "$\{\langle ag_1, A_{ag_1}\rangle, \ldots, \langle ag_n, A_{ag_n}\rangle\}, \mathcal{AS}_{ag}$".

sub-system, and $\mathcal{A}_{ss}$ the set of actions of the social sub-system. The tuple $\langle org_{id}, Ag, \mathcal{A}_{org}\rangle$ represents an organisation, with org_{id} the organisation identifier, Ag a set of agent identifiers, which are populating that organisation, and $\mathcal{A}_{org}$ the set of organisational actions. The tuple $\langle ag_{id}, \mathcal{A}\rangle$ represents an agent in the multi-agent system, with ag_{id} the agent identifier, and $\mathcal{A}$ 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, $\mathcal{AS}_{ag}$, $\mathcal{AS}_{org}$, and $\mathcal{AS}_{ss}$, respectively. When an action is executed by an agent, the *VIC*s 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 `org1` and `org2`. On the top of our multi-agent system, we have the social sub-system `ssys` 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 *VIC*s are not satisfied (none process is satisfied) only the agent level (c) is updated in accordance.

$$\frac{\begin{array}{c} execute(\texttt{interview}) \quad \texttt{interview} \in \mathcal{A} \\ \#VIC_{ag}^{org} \Uparrow [\mathcal{AS}_{org}, \mathcal{A}_{org}, \mathcal{AS}_{ag}, \{\texttt{interview}\}] = \{\ \} \\ \#VIC_{org}^{ss} \Uparrow [\mathcal{AS}_{ss}, \mathcal{A}_{ss}, \mathcal{AS}_{org}, \{\ \}] = \{\ \} \end{array}}{\begin{array}{llcl} (a) & \langle \texttt{ssys}, \mathcal{O}, \mathcal{A}_{ss}\rangle \ldots \mathcal{AS}_{ss} & \rightarrow_{ss} & \langle \texttt{ssys}, \mathcal{O}, \mathcal{A}_{ss}\rangle \ldots \mathcal{AS}_{ss} \\ (b) & \langle \texttt{org}_2, Ag, \mathcal{A}_{org}\rangle \ldots \mathcal{AS}_{org} & \rightarrow_{org} & \langle \texttt{org}_2, Ag, \mathcal{A}_{org}\rangle \ldots \mathcal{AS}_{org} \\ (c) & \langle \texttt{interviewer}, \mathcal{A}\rangle \ldots \mathcal{AS}_{ag} & \rightarrow_{ag} & \langle \texttt{interviewer}, \mathcal{A}'\rangle \ldots \mathcal{AS}'_{ag} \end{array}}$$

$$\begin{array}{lll} \textit{where:} & & \\ (c) \quad \mathcal{AS}'_{ag} & = & \mathcal{AS}_{ag} \cup \{\texttt{interview}\} \\ \quad\ \ \mathcal{A}' & = & c(\mathcal{A}, \mathcal{AS}_{ag}, \{\texttt{interview}\}) \end{array}$$

(ExInterviewAction)

$$\frac{\begin{array}{c} execute(\texttt{dismiss}) \quad \texttt{dismiss} \in \mathcal{A} \\ \#VIC_{ag}^{org} \Uparrow [\mathcal{AS}_{org}, \mathcal{A}_{org}, \mathcal{AS}_{ag}, \{\texttt{dismiss}\}] = \{\texttt{dismiss}_{\texttt{org}_1}\} \\ \#VIC_{org}^{ss} \Uparrow [\mathcal{AS}_{ss}, \mathcal{A}_{ss}, \mathcal{AS}_{org}, \{\texttt{dismiss}_{\texttt{org}_1}\}] = \{\ \} \end{array}}{\begin{array}{llcl} (a) & \langle \texttt{ssys}, \mathcal{O}, \mathcal{A}_{ss}\rangle \ldots \mathcal{AS}_{ss} & \rightarrow_{ss} & \langle \texttt{ssys}, \mathcal{O}, \mathcal{A}_{ss}\rangle \ldots \mathcal{AS}_{ss} \\ (b) & \langle \texttt{org}_1, Ag, \mathcal{A}_{org}\rangle \ldots \mathcal{AS}_{org} & \rightarrow_{org} & \langle \texttt{org}_1, Ag, \mathcal{A}'_{org}\rangle \ldots \mathcal{AS}'_{org} \\ (c) & \langle \texttt{manager}, \mathcal{A}\rangle \ldots \mathcal{AS}_{ag} & \rightarrow_{ag} & \langle \texttt{manager}, \mathcal{A}'\rangle \ldots \mathcal{AS}'_{ag} \end{array}}$$

$$\begin{array}{lll} \textit{where:} & & \\ (b) \quad \mathcal{AS}'_{org} & = & \mathcal{AS}_{org} \cup \{\texttt{dismiss}_{\texttt{org}_1}\} \\ \quad\ \ \mathcal{A}'_{org} & = & c(\mathcal{A}_{org}, \mathcal{AS}_{org}, \{\texttt{dismiss}_{\texttt{org}_1}\}) \\ (c) \quad \mathcal{AS}'_{ag} & = & \mathcal{AS}_{ag} \cup \{\texttt{dismiss}\} \\ \quad\ \ \mathcal{A}' & = & c(\mathcal{A}, \mathcal{AS}_{ag}, \{\texttt{dismiss}\}) \end{array}$$

(ExDismissAction)

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 $\#VIC_{ag}^{org} \Uparrow$ 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 `ExAppointAction` 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 *VIC*s are satisfied and all levels in the systems are updated.

$$\frac{\begin{array}{c} execute(\texttt{appoint}) \quad \texttt{appoint} \in \mathcal{A} \\ \#VIC_{ag}^{org} \Uparrow [\mathcal{AS}_{org}, \mathcal{A}_{org}, \mathcal{AS}_{ag}, \{\texttt{appoint}\}] = \{\texttt{appoint}_{\texttt{org}_2}\} \\ \#VIC_{org}^{ss} \Uparrow [\mathcal{AS}_{ss}, \mathcal{A}_{ss}, \mathcal{AS}_{org}, \{\texttt{appoint}_{\texttt{org}_2}\}] = \{\texttt{norm-conf\_exchange}\} \end{array}}{\begin{array}{llcl} (a) & \langle \texttt{ssys}, \mathcal{O}, \mathcal{A}_{ss}\rangle \ldots \mathcal{AS}_{ss} & \rightarrow_{ss} & \langle \texttt{ssys}, \mathcal{O}, \mathcal{A}'_{ss}\rangle \ldots \mathcal{AS}'_{ss} \\ (b) & \langle \texttt{org}_2, Ag, \mathcal{A}_{org}\rangle \ldots \mathcal{AS}_{org} & \rightarrow_{org} & \langle \texttt{org}_2, Ag, \mathcal{A}'_{org}\rangle \ldots \mathcal{AS}'_{org} \\ (c) & \langle \texttt{manager}, \mathcal{A}\rangle \ldots \mathcal{AS}_{ag} & \rightarrow_{ag} & \langle \texttt{manager}, \mathcal{A}'\rangle \ldots \mathcal{AS}'_{ag} \end{array}}$$

$$\begin{array}{lll} \textit{where:} & & \\ (a) \quad \mathcal{AS}'_{ss} & = & \mathcal{AS}_{ss} \cup \{\texttt{norm-conf\_exchange}\} \\ \quad\ \ \mathcal{A}'_{ss} & = & c(\mathcal{A}_{ss}, \mathcal{AS}_{ss}, \{\texttt{norm-conf\_exchange}\}) \\ (b) \quad \mathcal{AS}'_{org} & = & \mathcal{AS}_{org} \cup \{\texttt{appoint}_{\texttt{org}_2}\} \\ \quad\ \ \mathcal{A}'_{org} & = & c(\mathcal{A}_{org}, \mathcal{AS}_{org}, \{\texttt{appoint}_{\texttt{org}_2}\}) \\ (c) \quad \mathcal{AS}'_{ag} & = & \mathcal{AS}_{ag} \cup \{\texttt{appoint}\} \\ \quad\ \ \mathcal{A}' & = & c(\mathcal{A}, \mathcal{AS}_{ag}, \{\texttt{appoint}\}) \end{array}$$

(ExAppointAction)

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

[1] Olivier Boissier, Rafael H Bordini, Jomi F Hübner, Alessandro Ricci, and Andrea Santi, 'Multi-agent oriented programming with jacamo', *Science of Computer Programming*, **78**(6), 747–761, (2013).

[2] Antônio Carlos Rocha Costa and Graçaliz Pereira Dimuro, 'A minimal dynamical mas organization model', *Handbook of Research on Multia-Agent Systems: Semantics and Dynamics of Organizational Models*, 419–445, (2009).

[3] Pablo Noriega, *Agent mediated auctions: the fishmarket metaphor*, Institut d'Investigació en Intelligència Artificial, 1999.

[4] Alison R Panisson, Rafael H Bordini, and Antônio Carlos Rocha Costa, 'Towards multi-level semantics for multi-agent systems', in *3th Workshop-School on Theoretical Computer Science*, (2015).

[5] Alison R. Panisson, Felipe Meneguzzi, Moser Fagundes, Renata Vieira, and Rafael H. Bordini, 'Formal semantics of speech acts for argumentative dialogues', in *AAMAS*, pp. 1437–1438, (2014).

[6] Alison R. Panisson, Felipe Meneguzzi, Renata Vieira, and Rafael H. Bordini, 'Towards practical argumentation in multi-agent systems', in *BRACIS*, (2015).

[7] Gordon D Plotkin, 'A structural approach to operational semantics', (1981).

[8] Renata Vieira, Álvaro Moreira, Michael Wooldridge, and Rafael H. Bordini, 'On the formal semantics of speech-act based communication in an agent-oriented programming language', *J. Artif. Int. Res.*, **29**(1), 221–267, (June 2007).