Formalizing Interactive Institution with RBA Calculus

Guoyong Cai¹ Ji Gao¹ Junyan Qian² Lingzhong Zhao²

¹College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, P. R. China
²College of Computer Science and Technology, Guilin University of Technology, Guilin 541004, P. R. China

Abstract

Although organization oriented paradigm is a promising way to design dynamic interactive computational systems in open and heterogeneous networked environment, there is still lack of a suitable formal approach to support this paradigm. RBA, a boxed ambient calculus extended with separated regulating mechanism, is proposed to design such systems. In RBA, an ambient is an encapsulating mobile unit parameterized with role-governors that regulating actions of the corresponding role-players participating into this ambient. Firstly the formal syntax and semantics of RBA are given. Secondly an electronic institution model is applied to illustrate the RBA approach. Based on the bisimulation theory of process calculi, verification and validation of design specifications are supported with RBA approach.

Keywords: Interactive computation, Ambient calculus, Electronic institution, Behavior regulation

1. Introduction

Globally interactive and collaborative computational systems are presently one important type of information-based systems, which forms different kind of virtual organizations penetrating to people’s everyday lives, such as electronic community, electronic commerce systems. To develop such kind of systems efficiently and correctly, different methodologies and frameworks have been proposed to cope with the difficulties while designing such systems; among them agent-oriented methodology and organization-based frameworks are becoming the most promising ways. Agent-oriented paradigm [1] provides good structural methods to identify the main building blocks of such systems exploiting agents, roles and their relations in organizational settings, such as electronic institution or some policy-based frameworks, which provide good starting points toward understanding development of such systems. However currently these techniques haven’t provided adequately support for building this type of systems in a formal way that helps the verification and validation at suitable abstraction level. Especially when concerning the design phase of such systems, the main complexity arises from the complex interactive and regulating patterns among agents and their regulators [2]-[3]. Presently these agent-oriented or organization-oriented paradigms do not provide effective means to deal with these aspects adequately [4]-[6]. Therefore there is a great need to enhance these paradigms by providing an appropriate design level specification language to encapsulate regulative and interactive patterns in suitable abstraction and compositional modes.

On the other hand, due to its compositional and formal features, process calculi has been widely adopted for the design specification of concurrent systems, especially those with communicating, concurrently executing software components. However, most of the efforts are oriented to study process algebras suitable for homologous and closed environment, such as CCS [7] or the π-calculus [8], although another type of process calculi, such as ambient calculus [9]-[10], provide some encapsulation mechanism to process movement in networked environment, it is still not adequate to deal with the complex interactive and regulative patterns appear in virtual organization in open and heterogeneous networked environment. Further more traditional process calculi study does not incorporate itself with well formed concepts originating from organizational theory, such as role, agent, and policy.

To achieve better practice in organization oriented collaborative system development, we argue that both organization oriented and process oriented structural mechanisms are necessary, process calculi should be extended with organization-based regulating mechanisms. In this paper, RBA (Regulating Boxed Ambient), an extended boxed ambient calculus with regulating mechanism under organizational framework is proposed. Electronic institution model are applied to show the usage of the proposed calculus language.

The paper is organized as follows. In Section 2 the feature of ambient calculus is briefly introduced, and then the syntax and semantics of RBA is given. In section 3, an electronic institution model is introduced
which will serve as the meta regulating framework towards open and heterogeneous virtual environment for interactions and regulations. In section 4, RBA is applied to design regulated interactive computational system under electronic institutional framework. Some conclusion is drawn in the final section.

2. Ambient calculus with regulating mechanism

2.1. Background and motivation

Ambient Calculus (AC) [10] is a process algebra that focuses on the notions of locations, mobility and authorizations. The underlying model of the AC is based on the notion of ambient. An ambient is a place limited by a boundary where computations or interactions take place. They are hierarchically structured and the evolving path towards a destination is not abstracted away. Processes are confined to structures and the evolving path towards a destination is presented from five aspects: syntax, structural equivalence relation, supervising regulating capabilities, that is every ambient is equipped with a number of governors that supervise the activities of the corresponding components (sub-ambients) running inside the ambient. Hence the ambient notation \( n[P] \) (resp. \( n[G][a[P]] \)) etc. is extended to \( n(G)[P] \) (resp. \( n(G)[a[P]] \)) which denotes that process \( P \) (resp. ambient \( a[P] \)) runs in ambient \( n \) and is regulated by \( G \). If there is no need to have such governors for an ambient, it is also denoted by original notation \( n(P) \). Therefore \( n(P) \) is taken as a special form of \( n(G)[P] \). We call \( n(G)[P] \) supervised ambient and \( n(P) \) non-supervised ambient. Based on this idea originated from organizational supervising theory, in the following section, we propose RBA calculus—an enhanced ambient calculus.

2.2. An extension of boxed ambient

We assume a countable set of names, \( N \), ranged over by \( m, n, \ldots \); \( u, v, w, \ldots \); \( x, y, z, \ldots \) and their decorated versions (\( m', \ldots \)) and vector versions (\( \tau, \nu, \ldots \)). To simplify reading, we shall use \( m, n, \ldots \) to denote ambient names, \( x, y, z, \ldots \) to denote input variables, \( u, v, w, \ldots \) to denote generic names, and \( W, X, \ldots \) to denote the identifier of defined processes or governors. RBA is presented from five aspects: syntax, structural equivalence, process reductions, governor reductions and configuration reductions.

Definition 2.1. (RBA processes, actions and conditions)

\[
P::= 0 | (x)^P | <M^\to P | M.P | u[P] | P|P | (vn)P | !P | P|Q | M.P | P+Q | X(v)
\]

Definition 2.2. (RBA process reductions) The non-supervised ambient.

This definition is an enhance version of classical ambient calculus. It adopts a shared channel way to neighboring communications among parent ambient and child ambient via input \((x)^P\) and output \(<M^\to P\). It also adds new capabilities (\(new\), \(spawn\) and \(box\)) for dynamic generation of name mapping, process and ambient. \(new(n,X)\) is used to create a fresh association between \( n \) and \( X \). \(spawn(P)\), \(P\) activate \( P\) in parallel with \( P\), thus it corresponds to \( box(n,P) \). It is used to create a new ambient, i.e. \( n(P) \). The other notations are the same as in original ambient calculus.

Definition 2.3. (RBA process reductions) The non-supervised process reduction relation, \( \rightarrow \), is the least relation satisfying the following axioms.

\[
P[0]=P \quad P_1[P_2]=P_2[P_1] \\
P_1[!P_2]=(!P_2)[P_1] \quad !P=!P \\
(vn)0=0 \quad (vn)mP=(vm)(vn)P \\
P_1[(vn)P]=((vn)(P_1)[P_2]) \text{ if } n \notin \{P_2\} \\
m[(vn)P]=(vn)m[P] \text{ if } n=m \quad (M.M')P=M.(M'.P)
\]
new \text{new}(?x).P \xrightarrow{n(w(x))} P[x := n]

spawn \text{spawn}(P).P_2 \xrightarrow{\text{spawn}(P)} P_2

box \text{box}(n, P).P_2 \xrightarrow{\text{box}(n, P)} P_2

rec \begin{align*}
X(v) \xrightarrow{a(n)} P' & \quad \text{if } X(v) \not\in P
\end{align*}

in \text{in}(a).P \xrightarrow{\text{in}(a)} P

wrd \begin{align*}
< v >.P \xrightarrow{\sigma} P
\end{align*}

rd \begin{align*}
(x).P \xrightarrow{a(b)} P'[x := v]
\end{align*}

rwr \begin{align*}
< v >.P \xrightarrow{\sigma} P
\end{align*}

rdl \begin{align*}
(x)^\cdot P \xrightarrow{\sigma} P'[x := v]
\end{align*}

(6) \begin{align*}
\text{out}(a).P \xrightarrow{\text{out}(a)} P
\end{align*}

(12) \begin{align*}
\forall i \in \{1, 2\}
\end{align*}

(13) \begin{align*}
\forall i \in \{1, 2\}
\end{align*}

\begin{align*}
\text{G} \in \text{Gov} := \begin{cases}
\text{when } b \ p(x) \text{ then } G \mid \text{when } b
\end{cases}
\end{align*}

\begin{align*}
p \in \text{Perm} := \begin{cases}
\text{new} \mid \text{spawn} \mid \text{in} \mid \text{out} \mid \text{lcom} \\
\text{wrdn} \mid \text{rddn}
\end{cases}
\end{align*}

\begin{align*}
b \in \text{Bool} := \bot \mid \top \mid v_1 = v_2 \mid b_1 \lor b_2 \mid b_1 \land b_2 \mid \neg b
\end{align*}

The basic construct for governors is when \_ p(x) then G, where p is the permitted action, x is a sequence of variables bound by the construct, b is a boolean expression, and G is a governor process. It means that if b is evaluated to true, then p(x) is permitted and the governor evolves to G. The basic permission set Perm is inspired by the co-actions in safe ambient calculus [11]. But here we use co-actions as regulated authorizations, not used as those in synchronous calculi. GI p(x) denotes authorization allowed by G except p(\_). The other constructs are similar to those used logic expressions.

\textbf{Definition 2.5.} (RBA governor process reductions) The governor reduction relation, \(
\rightarrow_I\), is the least relation satisfying the following axiom and rules:

\begin{align*}
\text{perm}_1: & \quad \begin{align*}
b \chi \vdash \top \quad \text{when } b \ p(x) \text{ then } G \xrightarrow{\chi} G'
\end{align*}
\end{align*}

\begin{align*}
\text{perm}_2: & \quad \begin{align*}
\neg b \chi \vdash \top \quad \text{true } G \xrightarrow{\chi} G'
\end{align*}
\end{align*}

\begin{align*}
\text{perm}_3: & \quad \begin{align*}
\text{when } b \ G \chi \vdash \top \text{ and } b \ G \chi \vdash \top
\end{align*}
\end{align*}

\begin{align*}
\text{ref} & \quad \begin{align*}
G \chi \vdash \top \quad \text{when } G \chi \vdash \top
\end{align*}
\end{align*}

The other constructs are similar to those used logic expressions.
Electronic institutional models

Electronic institution (EI) is a promising framework to develop regulated interactive computation system in open networked environment. To illustrate the RBA capability, we use RBA to formally represent the EI model. But before presenting the procedure, we first briefly introduce the EI model. Based on the work [12], EI is formulated as follows.

Definition 3.1. An electronic institution is defined as a 5-tuple $EI =< PS, IR, ssd, N_{PS} >$, where:

1. $PS$ stands for a performative structure;
2. $IR$ is a subset of roles representing the institutional roles;
3. $ssd$ stands for the hierarchy partial order over the roles;
4. $ssd$ is the set of static separation of duties between roles; and
5. $N_{PS}$ stands for a set of normative rules.

Definition 3.2. The performative structure $PS$ is $PS =< S, T, s_0, s_T, E, f_j, f_j^0, C, ML, \mu >$ where:

1. $S$ is a set of scenes;
2. $T$ is a set of transitions;
3. $s_R \subseteq S$ is the initial scene;
4. $s_T \subseteq S$ is the final scene;
5. $E = E^1 \cup E^0$ is a set of arc identifiers where $E^1 \subseteq S \times T$ is a set of edges from scenes to transitions and $E^0 \subseteq T \times S$ is a set of edges from transitions to scenes;
6. $f_j : E \rightarrow DNF$ maps each arc to a disjunctive normal form of pairs of agent variable and role identifier representing the arc label; (7) $f_j : T \rightarrow \mathbb{P}$ maps each transition to its type; (8) $f_j^0 : E^0 \rightarrow \mathbb{P}$ maps each arc to its type (one, some, all or new); (9) $C : E \rightarrow ML$ maps each arc to a meta-language expression of type Boolean, i.e. a formula representing the arc’s constraints that agents must satisfy to traverse the arc; (10) $ML$ is a meta-language; (11) $\mu : S \rightarrow \{0,1\}$ states whether a scene can be multiply instantiated at run time or not.

Definition 3.3. A scene of EI is a tuple $S=< R, DF, W, w_0, W, (WA)_{r \in R}, (WE)_{r \in R}, \lambda, min, max >$ where:

1. $R$ is the set of scene roles involved in that scene;
2. $DF$ is the restriction to the scene $s$ of the EI dialogical framework defined below; (3) $W$ is the set of scene states; (4) $w_0 \in W$ is the initial state; (5) $W \in W$ is the set of final states; (6) $(WA)_{r \in R} \subseteq W$ is a family of sets such that $WA_r$ stands for the set of access states for role $r \in R$; (7) $(WE)_{r \in R} \subseteq W$ is a family of non-empty sets such that $WE_r$ stands for the set of exit states for role $r \in R$; (8) $\lambda \subseteq W \times W$ is a set of directed edges; (9) $\lambda : \theta \rightarrow L$ is a labelling function, where $L$ can be a timeout, or an illocution schemata and a list of constraints; (10) $min, max : R \rightarrow \mathbb{P}$ and $max(r)$ return the minimum and maximum number of agents that must and can play role $r \in R$.

Definition 3.4. A dialogue framework $DF$ is a tuple $DF =< O, L, I, RI, RE, RS >$, where:

1. $O$ stands for the EI domain ontology;
2. $L$ stands for a content language to express the information exchanged between agents; (3) $I$ is the set of illocutionary particles, usually it takes the form of $i(a, r, a', r', m, t)$, meaning that agent $a$ playing role $r$ sends illocution $i$ with content $m$ to agent $a'$ playing role $r'$ at time $t$.
4. $RI$ is the set of internal roles; (5) $RE$ is the set of external roles; (6) $RS$ is the set of relationships over roles.

Dialogue framework, scene, performative structure provide a basic interactive space of agents. Based on these structural notions, some predicates or functions can be defined. These definitions will further applied to specify normative rules for an EI. For example, if we define the following predicates, one type of normative rules takes the form of definition 2.5.

(1) uttered$(s, w, i)$ denoting that a grounded illocution unifying with the illocution scheme $i$ has been uttered at state $w$ of scene $s$.

(2) uttered$(s, i)$ denoting that a grounded illocution unifying with the illocution scheme $i$ has been uttered at some (unspecified) state of scene $s$.

Definition 3.5. Normative rules are first-order formulae of the form:

($\wedge_{j=1}^m \neg \text{uttered}(s_j, [w_{j_k}], i_j) \land \wedge_{k=1}^{n} e_{i_k}$) →

($\wedge_{j=1}^m \text{uttered}(s_j, [w_{j_k}], i_j) \land \wedge_{k=1}^{n} e_{i_k}$) where: $s_j, s_j$ are scene identifiers, $w_{j_k}, w'_{j_k}$ are states of $s_j$ and $s_j$ respectively; $i_j, i'_{j_k}$ are illocution schemata $i_j$ of scenes $s_j$ and $s_j$ respectively, and $e_{i_k}, e'_{i_k}$ are Boolean.
expressions over variables from the illocution schemata $i_j$ and $i'_j$ respectively.

The intuitive meaning of normative rules is that if grounded illocutions matching $i_1, \ldots, i_n$ are uttered in the corresponding scene states and the expressions $e_1, \ldots, e_m$ are satisfied, then grounded illocutions matching $i'_1, \ldots, i'_n$ satisfying the expressions $e'_1, \ldots, e'_m$ must be uttered in the corresponding scene states.

Although EI model has been studied by several researchers and this type of EI representation provides a quite clear image to electronic institutions, however it is rather informal and concurrency haven’t been considered in appropriate way [3][6][12]-[13]. As lack of concrete syntax and semantics definition for EI, we can not execute the EI specification and prove some properties that an EI should maintained with structural techniques. Thus in the following, we propose to represent EI with RBA, and then EI verification or validation can be coped with process bisimulation theory.

4. Design electronic institutions with RBA

From the above presentation of RBA and EI, there is an attractive mapping of scenes to immobile ambients with governors, where the conversation can take place via speech acts, and of agents to mobile ambients without governors, which move from scene to scene under its internal decisions given the appropriate role-playing capability. That is for each scene, a corresponding ambient with role governors will be built to regulate the joint task activities performed by the joining role playing agents, which itself will be modelled by an mobile ambient with role playing capabilities. For instance, we use ambient $s_l[f_1, f_2][PS]$ to denote a scene named $s_l$ with two role governors $r_1$ and $r_2$, and $PS$ denotes the process own by the scene itself; and we use $ag[r_i]$ to denote an agent named $ag$ which has capability of playing role $r_i$. Combining these two parts, we can generate a scene instance, such as $s_l(f_1, f_2)[PS, ag_1[r_1], ag_2[r_1], ag_3[r_2], ag_4[r_3]]$. In this instance, there are two agents ($ag_1$ and $ag_2$) who move into the scene and play the role $r_1$, while only one agent ($ag_3$) moves into and plays the role $r_2$.

Generally an EI is mapped to RBA constructs according to the criteria listed in table 1.

<table>
<thead>
<tr>
<th>EI RBA notation</th>
<th>Mobile ambient with role playing capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene name</td>
<td>Immobile ambient name</td>
</tr>
<tr>
<td>Immobile</td>
<td>n,s,...$\in$ Nam</td>
</tr>
</tbody>
</table>

To further illustrate the application of RBA to EI design, a classical example designed with RBA is presented in the following. Suppose that there is a need to construct an academic workshop institution which supports to deal with the working scenes of organizing a workshop. We suppose there are three main scenes involving in this institution. That is submitting papers, reviewing papers and paying publication fees. For each scene, there exist a scene manager and participating agents. For example, in the submitting scene, there are one manager role (manager1) and one author role. The three scenes and attached roles are shown in figure 1. The illocutions related to domain activities are labelled as tags along the lines in figure 1. To understand easily the data flow, institutional data files are explicitly separated from institutional scenes in this figure.
According to the criteria, the RBA of this EI is specified in Table 2. Line (1) indicates that this institution is composed of four scenes (scene0-3). Scene0 is the top scene that represents the register scene in which the agent can register to play roles of the institution. If an agent successfully registers into the institution, a certificate is issued to the agent, with which authorized agents can join other corresponding scenes if they decide to do so. For example, if an agent registers successfully as an author player in scene0, then it can join submitting scene to submit a paper to the workshop or query the information related to its paper submitted. During the scene interaction, author player agent can only do the authorized actions according to the author governor. Other actions will not be permitted.

With RBA formalism of EI, EI design can be verified with process theory. This aspect will not be presented here for beyond the scope of this paper.

5. Conclusions

Developing trustable interactive system is a difficult work in open and heterogeneous environment. Social and organizational theories provide suitable starting point to guide the development of EI. There is still lack of systemic engineering method to make them much practicable usage. For instance, EI has been discussed by some papers [8]-[9], but many of them focus on modeling EI with deontic logic or state-based formalism, focusing on design and implementing EI. The work in this paper proposes to design globally interactive collaboration system starting from electronic institution model and specified with RBA. RBA is designed to present the complex interactive and regulative patterns in system design specification. Through formal defined operational semantic rules, system requirement norms can be verified at design level. A proof of concept example has been working out and the experience shows some great advantages over other approaches of such type of interaction complex systems. As far as future work is concerned, we are studying on verifying EI with RBA formalism.

Acknowledgement

This work is partially supported by National Nature Science Foundation of Guangxi Province, China (Grant No. 0728089).

References


