Giving Operational Semantics to Multi-Context Systems Using DEVS

Pablo Pilotti[1], Ana Casali[1,2] and Carlos Chesñevar[3]

[1] Centro Int. Franco-Argentino de Ciencias de la Información y de Sistemas (CIFASIS)

[2] Facultad de Cs. Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario (UNR)

[3] Depto. de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur (UNS) - CONICET
Av. Alem 1253 - B8000CPB Bahía Blanca, Argentina
Email: cic@cs.uns.edu.ar

Abstract. Multi-context systems have proven to be a powerful tool for formalizing complex logical problems in Artificial Intelligence, providing a flexible framework that allows the definition of different formal components and their interrelationships. Several MCS applications are oriented towards multiagent systems, in which several asynchronous tasks (inferences, messaging, etc.) are carried out. In spite of their expressive power, MCS lack of an appropriate mechanism to capture their underlying operational semantics when they are used to provide a computational model for such systems. This paper presents a first approach to give operational semantics to MCS using Discrete Event System Specification (DEVS), a modular and hierarchical formalism for modeling, simulating and analyzing discrete event systems. We show that our proposal provides a flexible model for capturing several features of an agent's reasoning process in an asynchronous setting.

1 Introduction and motivations

Multi-context systems (MCS) [9, 10] have proven to be a powerful tool for formalizing several complex problems in Artificial Intelligence. MCS are defined as a set of (possibly different) logical *contexts* (or *units*) and a set of *bridge rules*. Each context in an MCS can be seen as a logical theory, from which new logic formulas can be inferred using internal inference rules. Different contexts are connected via bridge rules, which allow to exchange information among contexts. The deduction mechanism of MCS is thus based on two kinds of inference rules: internal rules, inside each unit, and bridge rules outside. Internal rules allow to draw consequences within a theory, while bridge rules allow to embed results from a theory into another [8].

One of the advantages of MCS in order to help in the design of complex logical systems is that this framework allows for the independent definition of formal components and their interrelations. MCS have been used in various applications, such as integrating heterogeneous knowledge and data bases [7], formalizing

meta reasoning and propositional attitudes [10], and modeling different aspects of agent architectures and multiagent systems (e.g. [2, 6, 5, 11], among others).

Several MCS applications are oriented towards multiagent systems, in which several asynchronous tasks (inferences, messaging, etc.) are carried out. In spite of their expressive power, MCS lack of an appropriate mechanism to capture their underlying operational semantics when they are used to provide a computational model for such systems.

Inference by bridge rules in MCS is asynchronous, and for many practical applications (e.g. specifying inference procedures in agents in a MAS setting, as discussed before) it turns out to be useful to formalize inference steps as discrete events. Several formalisms have been developed to cope with discrete events systems (DES), such as Petri nets, Statecharts, etc [12].

This paper presents a first approach to giving operational semantics to MCS using Discrete Event System Specification (DEVS), a modular and hierarchical formalism for modeling and analyzing discrete event systems. Our proposal is based on two classes of DEVS components (intended to model contexts and bridge rules, respectively), which are on its turn coupled to model a general MCS. We show that our proposal provides a flexible model for capturing several features of an agent's reasoning process in an asynchronous setting.

The rest of the paper is structured as follows. First, in Section 2 we summarize the main elements characterizing multi-context systems. Section 3 provides an overview of DEVS, discussing the specification of both atomic and coupled models. Section 4 presents our proposal for specifying an operational semantics for a MCS using DEVS. We also include an illustrative example of our proposal. Finally in Section 6 some related work and conclusions are exposed.

2 Multi-Context Systems

A MCS is essentially a set of logical theories, plus a set of inference rules which allow the propagation of consequences among theories. MCS are suitable to specifying and model agent architecture because they support modular decomposition and provide an efficient mean of specifying and executing complex logic [11]. Formally, given a family of languages $\{L_i\}$ over I, a Multi-Context System is a pair $\langle \{C_i\}_{i \in I}, \Delta_{br} \rangle$, where:

- for each $i \in I$, $C_i = \langle L_i, A_i, \Delta_i, T_i \rangle$ where L_i, A_i, Δ_i are the language, axioms and inference rules respectively, and T_i is a set of formulas (also called *theory*) written in the logic L_i
- Δ_{br} is a set of bridge rules. They are rules of inference which relate formulas in different languages, noted by:

$$\frac{C_1 : f_1, \ \ldots \ , C_n : f_n}{C_h : f_h}(Br_j)$$

  This means that Br_j allows us to export the formula f_h to the context C_h because of the fact that all the $f_1 \ldots f_n$ are derivable in the contexts tagged with $C_1, \ \ldots \ , C_n$, respectively.

Example 1. Consider the MCS $S = \langle \{C_i\}_{i \in I}, \Delta_{br} \rangle$, where:

- For each $i \in I = \{1, 2, 3\}$, $C_i = \langle L_i, A_i, \Delta_i, T_i \rangle$ is defined as follows:

	C_1	C_2	C_3
L	PL	PL	PL
A	$AxPL$	$AxPL$	$AxPL$
Δ	MP	MP	MP
T	$\{a \to b, b \to c, b \to d\}$	$\{b \to a, a \to h\}$	$\{a, c \to e\}$

- $\Delta_{br} = \left\{ \dfrac{C_1 : b, C_2 : h}{C_3 : c}(Br_1),\ \dfrac{C_3 : b}{C_2 : b}(Br_2),\ \dfrac{C_1 : a \to b}{C_3 : a \to b}(Br_3) \right\}$

In this simple example there are three contexts and three bridge rules. All three contexts share *propositional logic (PL)* as underlying representation language, a standard set *AxPL* of axioms[4] and *Modus Ponens (MP)* as internal inference rule. However, their associated theories (set of formulas) for each context are different. Figure 1 shows a graphical representation of this example, where the three contexts are interconnected by the bridge rules Br_1, Br_2 and Br_3. In this situation only Br_3 can be applied (or "fired") since $a \to b \in T_1$ and therefore $a \to b$ can be inferred in C_3. Once Br_3 has been applied, MP can be used locally in context C_3 to infer b from $\{a, a \to b\}$. In what follows we will

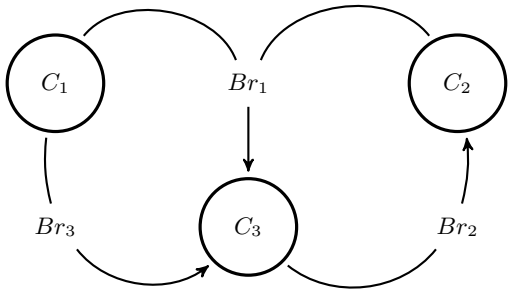

Fig. 1. Graphical representation of a multi context system (example 1)

consider some operators which will help us to model the inference process when performing context-based reasoning:

- $BrIn(C_i)$ is the set of Bridge rules that introduce information in C_i.
- $BrOut(C_i)$ is the set of Bridge rules connected with C_i that bring information to other contexts.
- $ContIn(Br_j)$ is the set of Contexts included in the Precondition of Br_j.
- $ContOut(Br_j)$ is the set of Contexts included in the Postcondition of Br_j.
- $PRE(Br_j)$ is the set of Contexts and formulas included in the Precondition of Br_j.

[4] The set *AxPL* corresponds to Łukasiewicz Axioms: (Ax1) $\Phi \to (\Psi \to \Phi)$ (Ax2) $(\Phi \to (\Psi \to \chi)) \to ((\Phi \to \Psi) \to (\Phi \to \chi))$ and (Ax3) $(\neg\Psi \to \neg\Phi) \to (\Psi \to \Phi)$.

– $POS(Br_j)$ is the Postcondition of Br_j.

Table 1 illustrates the outputs obtained for each operator (displayed in the first column) for their corresponding inputs for the MCS presented in Example 1 (e.g. $BrIn(C_1) = \emptyset$).

	C_1	C_2	C_3	Br_1	Br_2	Br_3
$BrIn$	$\emptyset$	$\{Br_2\}$	$\{Br_1, Br_3\}$			
$BrOut$	$\{Br_1, Br_3\}$	$\{Br_1\}$	$\{Br_2\}$			
$ContIn$				$\{C_1, C_2\}$	$\{C_3\}$	$\{C_1\}$
$ContOut$				$\{C_3\}$	$\{C_2\}$	$\{C_3\}$
PRE				$\{C_1 : b, C_2 : h\}$	$\{C_3 : b\}$	$\{C_1 : a \rightarrow b\}$
POS				$\{c\}$	$\{b\}$	$\{a \rightarrow b\}$

Table 1. Operators used to model context-based reasoning (outputs for the MCS in Example 1)

3 Discrete event systems specification

The Discrete Event System Specification (DEVS) is a formalism describing entities and behaviors of a system [12]. There are two kinds of models in DEVS: *atomic* and *coupled* models. An atomic model depicts a system as a set of input/output events, internal states and behavior functions. A coupled model consists of a set of models (atomics and/or coupled models), coupling information among the models, along with a set of *input/output ports*.

3.1 Atomic model

Formally an atomic DEVS model is a tuple defined by:

$$M = (X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta)$$

where:

– X is the set of input events.
– Y is the set of output events.
– S is the set of state value.
– ta is the *Time Advance Function*, a function ($ta : S \rightarrow \mathbb{R}_0^+$) that returns the state's *time advance*. This represents how long the system will remain in a given state in absence of inputs events.
– δ_{int} is the *Internal Transition Function*, a function ($\delta_{int} : S \rightarrow S$) that returns the next state the system will take if during the state's *time advance* do not occur input events.
– λ is the *Output Function* ($\lambda : S \rightarrow Y$) that returns the output events after an internal transition.
– δ_{ext} is the *External Transition Function* ($\delta_{ext} : S \times \mathbb{R}_0^+ \times X \rightarrow S$) that returns the next state the system will take after a input event occur. This function takes as argument the state of the system and the time spent in such state.

As an example, consider a simple system that starts in a state s_1 (see Figure 2), and no input events occur during $ta(s_1)$ (see Figure 3), so that the system produces the first output $y_1 = \lambda(s_1)$ (see Figure 4) and takes $s_2 = \delta_{int}(s1)$ as next state. After $e < ta(s_2)$ time, an external event x_1 occurs, so that the system takes $s_3 = \delta_{ext}(s_2, e, x_1)$ as next state. Since no inputs events occur during $ta(s_3)$, the system produces the second output $y_2 = \lambda(s_3)$ and takes $s_4 = \delta_{int}(s_3)$ as next state.

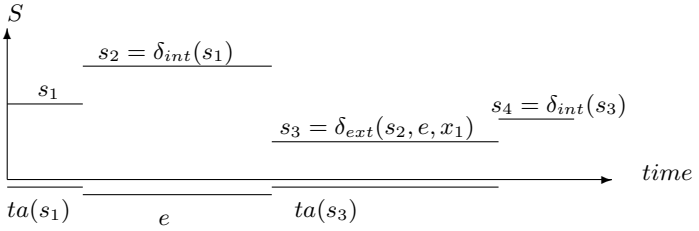

Fig. 2. System State

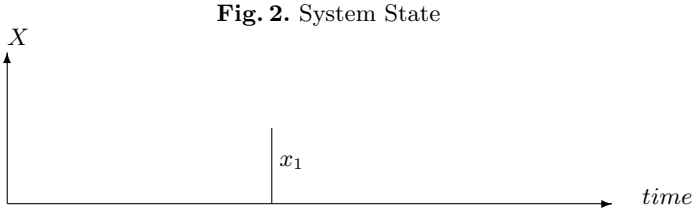

Fig. 3. Input Events

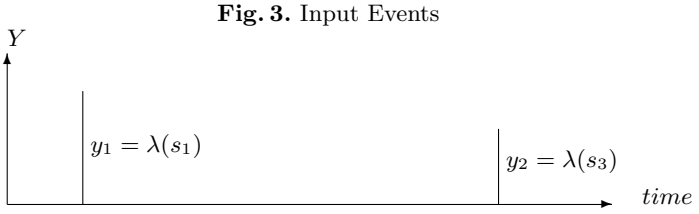

Fig. 4. System Output

3.2 Coupled model

DEVS atomic models are useful to model simple discrete event systems. When modeling complex systems it is easier to describe the system's elemental components and specify how they interact. The coupled models depict how the atomics models interact with each other. Coupled models provide a description of the interconnections in the atomic models using labels (called *ports*). To use atomic models in coupled models, the set of inputs and outputs for each atomic model M is defined as follows:

$$X = \{(p, v) \mid p \in InPorts,\ v \in X_p\}, \quad Y = \{(p, v) \mid p \in OutPorts,\ v \in Y_p\}$$

where:

- $InPorts$ and $OutPorts$ are the sets of Input and Output Ports, respectively.
- X_p and Y_p are the sets of Input and Output event values in the port p.

The coupled model is a tuple defined as

$$N = (X_N, Y_N, D, M_d, EIC, EOC, IC, Select)$$

where:

- The set of input values is $X_N = \{(p, v) \mid p \in InPorts,\ v \in X_p\}$ where $InPorts$ is the set of input port of the coupled model, and X_p is the set of input value of the port p.
- The set of output values is $Y_N = \{(p, v) \mid p \in OutPorts,\ v \in Y_p\}$ where $OutPorts$ is the set of output port of the coupled model, and Y_p is the set of output value of the port p.
- D is the set of references to the associated components (i.e. other DEVS models).
- For each $d \in D$, $M_d = (X_d, Y_d, S_d, \delta_{int_d}, \delta_{ext_d}, \lambda_d, ta_d)$, where
  - $X_d = \{(p, v) \mid p \in InPorts_d,\ v \in X_{p_d}\}$
  - $Y_d = \{(p, v) \mid p \in OutPorts_d,\ v \in Y_{p_d}\}$
- The set EIC (external input coupling) describes the links between external inputs and components inputs:

  $$EIC \subseteq \{((N, ip_N), (d, ip_d)) \mid ip_N \in Inports,\ d \in D,\ ip_d \in InPorts_d\}$$

- The set EOC (external output coupling) describes the links between external outputs and components outputs:

  $$EOC \subseteq \{((d, op_d), (N, op_N)) \mid op_N \in Outports,\ d \in D,\ op_d \in OutPorts_d\}$$

- The set IC (internal coupling) describes the link between external outputs and components outputs:

  $$IC \subseteq \{((a, op_a), (b, ip_b)) \mid a, b \in D, a \neq b, op_a \in Outports_a,\ ip_b \in InPorts_b\}$$

- $Select : 2^D \rightarrow D$ is a function that selects which component will make a transition in case of simultaneous events. ($Select(E) \in E$).

4 A multi-context system in DEVS

A general multi-context system can be modeled using two classes of atomics DEVS models. One of them is used to model contexts and the other is used to model bridge rules. For each context C_i in a multicontext system S, an atomic DEVS (referenced also as C_i) is defined by the tuple M_i (specified below), its $InPorts$ and $OutPorts$ are defined respectively as $BrIn(C_i)$ as $BrOut(C_i)$ (as defined in Section 2). On the other hand, for each bridge rule Br_j in S, an atomic DEVS (referenced as Br_j) is defined by the tuple Mr_j (specified below), and its ports are defined by the sets $ContIn(Br_j)$ and $ContOut(Br_j)$. It is important to note that models of the same class are not coupled among themselves. Then it is possible to represent any multi-context system by coupling those atomics models in a coupled DEVS model N.

Formally, a MCS $S = \langle \{C_i\}_{i \in I}, \Delta_{br} \rangle$ (for each $i \in I$, $C_i = \langle L_i, A_i, \Delta_i, T_i \rangle$) can be modeled by a coupled DEVS model:

$$N = (X_N, Y_N, D, \{M_d\}, EIC, EOC, IC, Select)$$

where each component is defined as follows:

- $X_N = \emptyset$.
- $Y_N = \emptyset$.
- $D = \{C_i\}_{i \in I} \cup \{Br_j\}_{j \in \Delta_{br}}$
- $\{M_d\} = \{M_i\}_{i \in I} \cup \{Mr_j\}_{j \in \Delta_{br}}$
- $EIC = \emptyset$.
- $EOC = \emptyset$.
- $IC = \{\ ((C_i, Br_j), (Br_j, C_i))\ \mid C_i \in ContIn(Br_j),\ Br_j \in BrOut(C_i)\}$
  $\cup \{\ ((Br_j, C_i), (C_i, Br_j))\ \mid C_i \in ContOut(Br_j),\ Br_j \in BrIn(C_i)\}$
- *Select* is defined at the implementation time.

We can consider that MCS are closed system. The information exchange with the environment may be considered as the interchange with another context especially defined for representing this environment. For this reason, X, Y, EIC, and EOC are defined as empty sets. The reference set D is the union of two sets: the first set of contexts references and the second is the set of bridge rules references. The atomics DEVS class for M_i and Mr_j are defined in the next subsections. The internal coupling set (IC) defines the internal connection between ports (e.g. the port labeled as Br_j of the context C_i will be connect with the port labeled as C_i of the bridge rule Br_j iff $C_i \in ContIn(Br_j)$ and $Br_j \in BrOut(C_i)$).

Atomic DEVS to model contexts

In this section we define the atomic DEVS used to model contexts. They are referenced as C_i, their $InPorts$ are defined as $BrIn(C_i)$ and $OutPorts$ as $BrOut(C_i)$. The main idea is that this atomic DEVS takes account of a set of formulas, and regularly makes an internal transition adding deduced formulas from axioms, inference rules and its own state. Then, it checks if exists a formula that appears as part of the premise of a bridge rule When an external event occurs, a formula is introduced into the state. Formally, a context $C_i = \langle L_i, A_i, \Delta_i, T_i \rangle$ where L_i, A_i, Δ_i are the language, axioms and inference rules respectively, and T_i is a set of formulas written in the language L_i, can be modeled by an atomic DEVS model:

$$M_i = (X_i, Y_i, S_i, \delta_{i_{int}}, \delta_{i_{ext}}, \lambda_i, ta_i)$$

where each component is defined as follows:

- $X_i = \{(Br_j, s) \mid Br_j \in BrIn(C_i), s \in L_i\}$.
- $Y_i = \{(Br_j, s) \mid Br_j \in BrOut(C_i), s \in L_i\}$.
- $S_i = 2^{L_i} \times \mathbb{R}_0^+$.
- $ta_i(s) = ta_i(g, \sigma) = \sigma$.

- $\delta_{i_{int}}(s) = \delta_{i_{int}}(g, \sigma) = (g \cup \Delta'_i(g), \sigma_i)$.
- $\lambda_i(s) = \lambda_i(g, \sigma) = \begin{cases} (Br_j, f) & \text{if } \exists\ Br_j, f : f \in g \wedge \\ & (C_i : f) \in PRE(Br_j) \wedge \\ & Br_j \in BrOut(C_i) \\ \emptyset & \text{otherwise} \end{cases}$
- $\delta_{i_{ext}}(g, e, x) = \delta_{i_{ext}}((s, \sigma), e, (Br_j, f)) = (s \cup f, \sigma - e)$

The input set (X_i) has as elements pairs of $InPort$ and formulas in the language L_i (the same applies for the output set Y_i). The set of value states (S_i) has as elements pairs: set of formulas in the language L_i (denoted as g) and a real number (denoted as σ). The *Time Advance Function* (ta_i) defines that the second element of the pair (σ) is the time that a system will remain in a state in absence of input events. The *Internal Transition Function* ($\delta_{i_{int}}$) introduce a formula (if there is any) returned by Δ'_i function and set to σ_i the time to the next internal transition. $\Delta'_i(g)$ is a function that makes a deduction using axioms (A_i) and inference rules (Δ_i) of the context C_i, and it is explicitly defined at implementation time since it is possible make more than one deduction from a set of formulas. The *Output Function* (λ_i) returns to the Br_j bridge rule a formula f whenever f is in g, $C_i : f$ is a precondition of Br_j and Br_j is a bridge rule connected with C_j that bring information to another context. Since an external event $x = (Br_j, f)$ stands for the presence of some input information, the *External Transition Function* ($\delta_{i_{ext}}$) adds f to the set of formula, and it updates the new σ in order the internal transition ($\delta_{i_{int}}$) takes place in the time previously set.

Atomic DEVS to model bridge rules

We define in this section Mr_j models to represent bridges rules. They are referenced as Br_j, their $InPorts$ are defined as $ContIn(Br_j)$ and $OutPorts$ as $ContOut(Br_j)$. The main idea is that this atomic DEVS takes account of a set of formula and their context of origin. In the case that an external event occurs, a formula and its context are introduced into the state; if not, the state remains unchanged. When the precondition of the bridge rule is part of the state, then the postcondition of the bridge rule is applied. Formally it can be model as:

$$Mr_j = (X_j, Y_j, S_j, \delta_{j_{int}}, \delta_{j_{ext}}, \lambda_j, ta_j)$$

where each component is defined as follows:

- $X_j = \{(C_i, f) \mid C_i \in ContIn(Br_j), f \in PRE(Br_j)\}$
- $Y_j = \{(ContOut(Br_j), POS(Br_j))\}$
- $S_j = 2^{PRE(Br_j)}$
- $ta_j(s) = \begin{cases} 0 & \text{if } PRE(Br_j) \subset s \\ \infty & \text{otherwise} \end{cases}$
- $\delta_{j_{int}}(s) = \emptyset$
- $\lambda_j(s) = (ContOut(Br_j), POS(Br_j))$

– $\delta_{j_{ext}}(s, e, x) = \delta_{j_{ext}}(s, e, (C_i, f)) = s \cup \{C_i : f\}$

The input set has as elements pairs of $InPort$ and formulae of the Br_j precondition. The set of value states has as elements a set of formulas in the precondition of Br_j. The *Time Advance Function* defines the system will remain in the same state until Br_j precondition holds in the states formulas. When the precondition holds, an internal transition makes empty the state, and the *Output Function* return the postcondition of Br_j.

5 Modelling MCS using DEVS: A Worked Example

When modelling complex systems it is easier to describe the system's elemental components and then specify how they interact. In this section, the Example 1 is modeled using the DEVS formalism. First of all, the associated atomic models are defined, i.e. contexts and bridge rules, and then the coupled model which specifies the interconnections:

I-Operator Computations: We compute the operators PRE, POS, $BrIn()$, $BrOut()$, $ContIn()$ and $ContOut()$ for each context and bridge rule of the Example 1, and show them in Table 1. They will be used to build the atomic and coupled DEVS models.

II-Modelling Contexts: For each context C_i, define $InPorts_{C_i}$ as $BrIn(C_i)$, $OutPorts_{C_i}$ as $BrIn(C_i)$ and construct the atomic DEVS model $M_i = (X_i, Y_i, S_i, \delta_{i_{int}}, \delta_{i_{ext}}, \lambda_i, ta_i)$ as described in Section 4. For the sake of example, the context C_2 is chosen to be specified, and the M_2 components are defined as follows:

– $X_2 = \{(Br_2, s) \mid s \in PL\}$.
– $Y_2 = \{(Br_1, s) \mid s \in PL\}$.
– $S_2 = 2^{PL} \times \mathbb{R}_0^+$.
– $ta_2(s) = ta_2(g, \sigma) = \sigma$.
– $\delta_{2_{int}}(s) = \delta_{2_{int}}(g, \sigma) = (g \cup \Delta_2'(g), \sigma_2)$.
– $\lambda_2(s) = \lambda_2(g, \sigma) = \begin{cases} (Br_1, f) \text{ if } \exists\ f : & f \in g \wedge \\ & (C_2 : f) \in \{C_1 : b, C_2 : h\} \\ \emptyset \quad \text{otherwise} \end{cases}$
– $\delta_{2_{ext}}(g, e, x) = \delta_{2_{ext}}((s, \sigma), e, (Br_2, f)) = (s \cup f, \sigma - e)$

III-Modelling Bridge Rules: For each bridge rule Br_j, construct the atomic DEVS Model $Mr_j = (X_j, Y_j, S_j, \delta_{j_{int}}, \delta_{j_{ext}}, \lambda_j, ta_j)$ as described in Section 4. For the sake of example, the bridge rule Br_1 is chosen to be specified, and the Mr_1 components are defined as follows:

– $X_j = \{(C_i, f) \mid C_i \in ContIn(Br_j), f \in PRE(Br_j)\}$
– $Y_j = \{(ContOut(Br_j), POS(Br_j))\}$
– $S_j = 2^{PRE(Br_j)}$
– $ta_j(s) = \begin{cases} 0 \text{ if } PRE(Br_j) \subset s \\ \infty \text{ otherwise} \end{cases}$

- $\delta_{j_{int}}(s) = \emptyset$
- $\lambda_j(s) = (ContOut(Br_j), POS(Br_j))$
- $\delta_{j_{ext}}(s, e, x) = \delta_{j_{ext}}(s, e, (C_i, f)) = s \cup \{C_i : f\}$

IV-Modelling the MCS: As described in Section 4, we can now build the coupled DEVS model $N = (X_N, Y_N, D, \{M_d\}, EIC, EOC, IC, Select)$. The DEVS models of the MCS is defined as: The sets X_N, Y_N, EIC, EOC, which are empty sets; the *Reference set* defined as $D = \{C_1, C_2, C_3, Br_1, Br_2, Br_3\}$, and the *Internal Coupling set* (IC) defined as

$$\begin{aligned} IC = \{&((C_1, Br_1), (Br_1, C_1)), ((C_1, Br_2), (Br_2, C_1)), ((Br_1, C_3), (C_3, Br_1)), \\ &((C_2, Br_1), (Br_1, C_2)), ((Br_2, C_3), (C_3, Br_2)), ((C_3, Br_3), (Br_3, C_3)), \\ &((Br_3, C_2), (C_3, Br_2))\} \end{aligned}$$

The *select* function selects which component will make a transition in case of simultaneous events. This function will be defined at simulation time.

At this point it is possible to depict an sketch of the system components and their connections. Figure 5 shows the system components with their $InPorts$ and $Outports$.

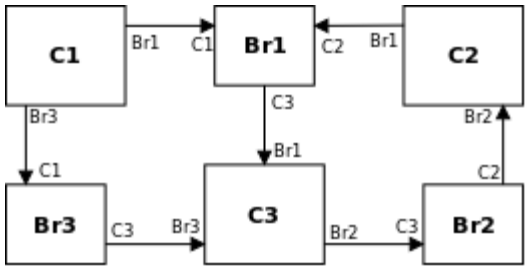

Fig. 5. A DEVS model: structural interconnections (Example 1)

Modelling inference

Next, we show a sample situation for modelling inference in our proposal. For each atomic DEVS corresponding to a context C_i we define $\sigma_i = 1$, Δ_i' as a function that only applies MP when it is possible, and the initial state $s_i = (T_i, \sigma_i)$. We assume that all atomics DEVS corresponding to bridge rules correspond to an initial state $s_j = \emptyset$. Then *select* is defined as the priority list $[C_1, C_2, C_3, Br_1, Br_2, Br_3]$. Figure 6 shows the context evolution from $time = 1$ to $time = 5$. Initially each theory is in its proper context state. At $t = 1$ all contexts make an internal transition ordered by the *select* function. Context C_1 does not add a formula to its state, but since $(C_1 : a \rightarrow b) \in PRE(Br_3)$ and $a \rightarrow b$ holds in the C_1 state, C_1 sends the external event $a \rightarrow b$ to Br_3. This make Br_3 take a transient state, sending $a \rightarrow b$ to C_3 and returning it to its initial state.

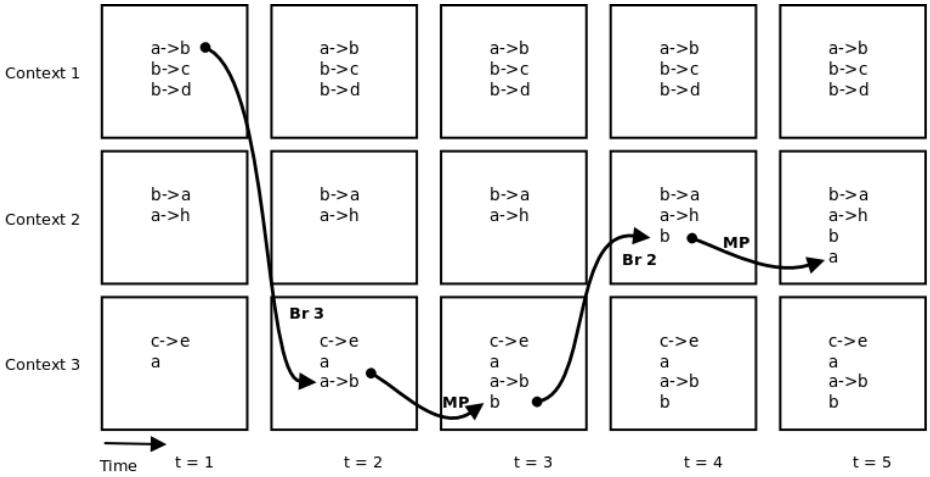

Fig. 6. Simulating the DEVS model of the system (Example 1)

6 Related work. Conclusions

During the last years, MCS have found several applications for modelling intelligent systems, such as merging heterogeneous knowledge, modeling propositional attitudes and specifying agent architectures in multiagent systems, etc. In spite of their expressive power, MCS lack of an appropriate mechanism to capture their underlying operational semantics when they are used to provide a computational counterpart for such systems.

In this paper we have presented a first approach to model MCS using DEVS, a modular and hierarchical formalism for modeling and analyzing discrete event systems. Our proposal relies on a methodology in which arbitrary contexts and bridge rules can be modeled through atomic DEVS. A coupled DEVS can be then defined, representing a particular MCS, by suitably coupling those atomics models. We show that the resulting approach provides a flexible model for capturing an agent's reasoning process in an asynchronous setting.

There have been some approaches to giving semantics to MCS, notably the Multi-context Calculus (MCC) proposed in [4]. This calculus is based on Ambient calculus [3] and is able to specify different kinds of MCSs and particularly, the authors have shown how a graded BDI agent model [5], specified using MCS, can be mapped to this language.

In contrast, our proposal is more focused on giving a practical operational semantics to MCS, with the purpose of obtaining a flexible computational model for capturing complex inference processes in agent reasoning. DEVs models result very suitable to represent all the MCS components (context and bridge rules) in a modular and natural way and the emphasis of this specification is on the asynchronous aspects. The deduction mechanism of these systems is based

on two kinds of asynchronous rules, internal rules inside each unit, and bridge rules outside. We found that DEVs are also suitable to represent in a clear way these different kinds of deductions and their syncronization. Also, we can take advantage of the variety of DEVs simulators to obtain computational models for MCSs and in particular, for capturing complex inference processes in agent reasoning.

Part of our current work involves two research directions. On the one hand, we are studying the theoretical results and emerging properties associated with our proposal. On the other hand, we are concerned with modeling argumentation capabilities in agents using a MCS approach, which seems a very promising alternative from an abstract viewpoint (as shown in [1]). Our DEVS-based methodology could be applied in argumentation frameworks to render easier the implementation of argument-based reasoning processes, as well the study and analysis of their operational semantics.

References

1. Gerhard Brewka and Thomas Eiter. Argumentation context systems: A framework for abstract group argumentation. In *Proc. of LPNMR Conf.*, pages 44–57, 2009.
2. Gerhard Brewka, Floris Roelofsen, and Luciano Serafini. Contextual default reasoning. In Manuela M. Veloso, editor, *IJCAI*, pages 268–273, 2007.
3. Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *FoSSaCS '98: Proceedings of the First International Conference on Foundations of Software Science and Computation Structure*, pages 140–155, London, UK, 1998. Springer-Verlag.
4. Ana Casali, Lluís Godo, and Carles Sierra. A language for the execution of graded bdi agents. pages 65–82, 2007.
5. Ana Casali, Lluis Godo, and Carles Sierra. g-bdi: A graded intensional agent model for practical reasoning. In Vicenç Torra, Yasuo Narukawa, and Masahiro Inuiguchi, editors, *MDAI*, volume 5861 of *Lecture Notes in Computer Science*, pages 5–20. Springer, 2009.
6. Alessandro Cimatti and Luciano Serafini. Multi-agent reasoning with belief contexts: The approach and a case study. In Michael Wooldridge and Nicholas R. Jennings, editors, *ECAI Workshop on Agent Theories, Architectures, and Languages*, volume 890 of *Lecture Notes in Computer Science*, pages 71–85. Springer, 1994.
7. Adam Farquhar, Richard Fikes, and Wanda Pratt. Integrating information sources using context logic. In *In AAAI-95 Spring Symposium on Information Gathering from Distributed Heterogeneous Environments*, 1995.
8. Chiara Ghidini and Fausto Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127:2001, 2001.
9. Fausto Giunchiglia and Fausto Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, 345:345–364, 1992.
10. Fausto Giunchiglia and Luciano Serafini. Multilanguage hierarchical logics (or: How we can do without modal logics), 1994.
11. Jordi Sabater, Carles Sierra, Simon Parsons, and Nicholas R. Jennings. Engineering executable agents using multi-context systems, 1999.
12. Bernard P. Zeigler. *Theory of Modeling and Simulation.* John Wiley, 1976.