A Semantic Framework for the Specification of an IHM : Case of a Video Conference Interface

A. CHOUTRI,
University Mentouri of Constantine
ALGERIA
choutriaicha@yahoo.fr,

ABSTRACT

The Interfaces Human Machine (IHM) are found in almost all computing systems. During the twenty last years, the interfaces controlled by the user became increasingly complex. Thus, the recourse to specification, development, checking and validation models is essential to ensure the principal characteristic properties of the IHM and to increase its flexibility and the facility of its use. Many specification formalisms classifications of the IHMs are published in the literature evaluating and comparing the formalisms. They are founded according to the expression capacity, the usability, the origin and the finality. They show that the formalisms are relatively incomplete, of quite weak expression, of a weak semantic rigour and quite difficult to use.

In our paper, we suggest a new powerful formalism for the formal specification of an interface video conference encouraging its validation before implementation. We formalize the interface in the tile logic which extends rewriting logic, taking into account rewriting with side effects and rewriting synchronization. This new logic offers a remarkable expressivity for the specification of rule-based computational systems behavior in a compositionnal style particularly, reactive systems and open systems.

Key Words: Video conference Interface, Tile Logic, Rewriting Logic, Formal Semantics, Dynamic Behavior.

1. Introduction

The Interfaces Human Machine (IHM) are found in almost all of the computing systems. During the twenty last years, the interfaces controlled by the user became increasingly complex. In the majority of the cases, these IHM constitute the only entrance point of the user in these systems and act like mediator elements. In these interfaces, the interactions are often carried out by a displacement in a data entry form via keys and sometimes they are more advanced such as the voice or the gestures. This complexity increase of the IHM results in a natural difficulty of design and implementation of the interactive systems. So, the risk of errors introduced during the development of the interactive systems increases. According to Myers in [1], the volume of the code of the IHM represents for on average 48% of the time assigned with the development process. Thus, specification, development, checking and validation models as well as IHM description notations are of a great need to control the evoked complexity. The Software engineering methods, particularly the formal methods, allow to bring solutions in many fields such as the critical systems (safety systems, embarked systems etc). Unfortunately, today, these methods are applied only to one portion of this software, which is often qualified as heart of the critical systems. The IHM are thus often excluded, the operation of the system being guaranteed by means of guards on the user actions [1]. The design process of the IHM is complex and must be iterative, so, the most suitable models of design are based on architecture in which the decomposition of the interactive application in various directing elements is necessary. Formal methods having precise and concise semantics adapted to the interactive systems, would be useful on the

one hand for the proof of properties, and on the other hand for generation of code. The engineering of the IHM has some formalisms whose classifications published in the literature [1] evaluating and comparing the formalisms between them are founded according to the expression capacity, the usability, the origin and the finality. These classifications, show that the formalisms are relatively incomplete, of quite weak expression power, of weak semantic rigour and quite difficult to use.

In this paper, we suggest a new powerful formalism: the Tile Logic [4], allowing to better understand the dynamic behavior of a complex system while reasoning in terms of its components and their interactions. The IHM considered, relates to a graphic, concurrent interactive system and of realistic size: it is a video conference system interface.

Tile logic is an extension of rewriting logic(in the unconditional case) taking into account the rewriting with side effects and the rewriting synchronization. The Rewriting logic is introduced by J. Meseguer [2], as a consequence of his work on general logics to describe the concurrent systems. Actually, it is recognized as a general semantic and logical framework and also a programming paradigm [3, 12, 13]. Several languages based on this logic are operational, almost known are : Maude (USA), ELAN (France) and CafeOBJ (Japan). It is a logic of concurrent changes which manipulates the concurrent states and computations naturally. It allows to reason on the possible complex changes corresponding to the atomic actions axiomatized by the rewriting rules. The experimentation of this formalism in the last few years, suggested significant extensions which gave rise to theories of rewriting offering a remarkable expressivity of rewriting logic in many applications However, the tile logic offers a flexible formal framework (meta formalism) to specify rule-based computational systems behavior such as reactive systems, open systems, coordination languages, concurrent systems, mobile calculi [7,8,9,10], etc. Recent applications relate to an interactive view of the logical programming [6] and a concurrent semantics theory centered around the causal and space aspects [7].

The objective of our work is to contribute: to the integrated and effective development of the IHM (in particular, the practical realization and the formal analysis of a video conference interface prototype), and to widen the class of the applications of rewriting logic (particularly the tile logic while highlighting its strong points by a new case study). We Formalize the interface in the tile logic to ensure the properties of compositionality and modularity for the resulting tile specification which can then be translated to an executable rewriting specification [14, 15,16].

The rest of the paper is organized as follows: The section 2 presents the theoritical concepts on what the proposal semantic framework is based. The section 3 is devoted to define the video conference with its graphical interface and to construct a tile system describing the behavior and the evolution of a video conference interface case. In the section 4, we conclude our work by an evaluation of the importance of the formalism applied and particularly, for the case study to deduce some future works.

2. Basic concepts

2.1 Rewriting logic extension : Tile logic

The ordinary rewriting rules allow to express naturally state changes and concurrent calculus but they have the disadvantage (in the unconditional case) of beeing independant on the interaction of the environment. In other words, they can be freely instanciated with any term in any context. Thereby and for having a reactive view of the system such as the components can be conceived separately and composed via their behavior, this logic has been extended to tile logic by enriching rewriting rules with observations allowing

to ensure the synchronization and to describe the interaction.

The tile logic was introduced by Ugo Montanari and Fabio Gadducci [4] at the university of Pisa, Corso, Italy, for modular descriptions of a large class of rule-based computational systems. This formalism whose inspiration comes from the world of the term rewriting and the theory of concurrency, allows to describe the dynamic evolution of the concurrent systems thanks to an extension of rewriting logic by cosidering the changes of states with side-effects and the synchronization of the rewritings. The principal idea is to impose, a dynamic restreint to the terms to which a rule can be applied. A set of rules (called tiles) is then employed to define the behavior of partially specified components (i.e, containing variables) called configurations, only in terms of the possible interactions with the inside or outside environment. Thus, the behavior of a system must be described like a coordinated evolution of its local configurations. So, the extension is carried out by the adoption of a format for the representation of the open generic configurations of reactive systems with coordination [5, 6, 8]. The tile logic exploits a three-dimensional view of the concurrent systems: horizontal (space) to model the states and the components according to the structure of the system, vertical (time) to model the labelled steps according to the flow of computation and the third dimension (concurrency) to model the distribution of the activities and the resources

2.2 Basic definitions

Definition1 : Tile

A tile is a 5-tuple (α, s, ,b t) we note by the sequent : $\alpha : s \xrightarrow[b]{a} t$

The name « tile » is due to the graphical representation appearance of such rule (Figure1).

iii, fii: initial and fInal input interfaces

ioi, foi: initial and fInal output interfaces

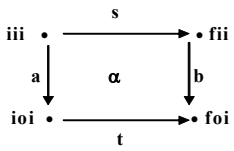

Figure1: Tile graphical representation

The defined tile means that the initial configuration s evolves to the final configuration t via the tile α, producing the effect b, which can be observable by the rest of the system (s can be an open term). Such a rewriting step is allowed only if the components of s (i.e, arguments to which s is connected via its input interface) evolve to the components of t producing an effect which acts like a trigger for α. The effects and the triggers are called observations and the vertices are called interfaces. The observations model the interaction during a computation of the system to be described.

The tiles can be composed horizontally, vertically and in parallel to generate larger steps (Figure2). The horizontal composition (operator ∗) model a synchronization of rewriting (eg, between the evolution of an argument via α and evolution of its environment via β since the effect of α acts like trigger of β and the resulting tile expresses the synchronized behavior of the two tiles). The vertical composition (operator ●) model the sequential composition of computations. The parallel composition (operator ⊗) corresponds to the construction of the concurrent steps where two disjoined configurations (or more) evolve in concurrency. The configurations and the observations are algebraic structures also supposed to be equipped with parallel and sequential operations (operators ⊗ and **;** respectively) allowing to build components in parallel and larger, extended horizontally for the configurations and vertically for the observations.

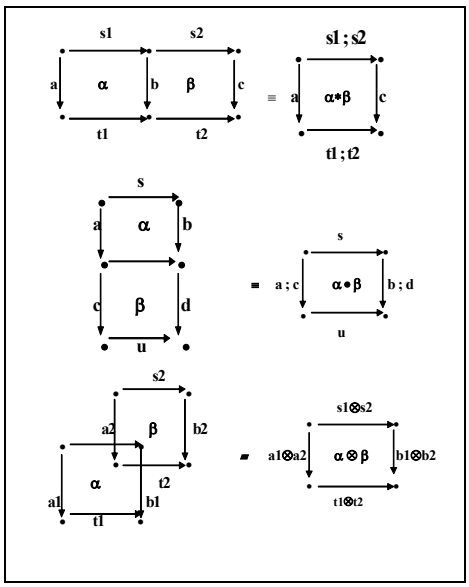

Figure2: Basic tile compositions

Definition2 : Tile system

A tile system is a 4-tuple *R=(H, V,* N, R) where *H, V* are monoïdal categories with the same set of objects *OH=OV,* N being a set of rule names and R: N⟶ *H* x *V* x *V* x *H* a function where for each α in *N*, if R(α)=(*s,a,b,t*), then *s:x⟶y, a:x⟶z, b:y⟶w and* t: z⟶*w*, for suitable objects *x, y, z* and *w*. Such a rule will be written in the form of $\alpha : s \xrightarrow[b]{a} t$ or of a tile (figure1) with *x* and z the input interfaces and, *y* and *w* the output interfaces The arrows of the horizontal category are called configurations and those of the vertical category are called observations. The objects of the two categories are called interfaces. The two categories are defined according to the chosen tile format (eg: Cartesian categories). A standard set of rules indicates how to build all the elements starting from basic tiles of the system via the horizontal, vertical and parallel compositions. Moreover, the horizontal and vertical identities whose intuitive meaning is that the elements of *H* and *V* stay unchanged during a rewriting (showing no effect and using no trigger), are always added to the system and are composed with the basic tiles. is illustrated by the inference rules of of the figure3.

In our paper, we suppose, that the basic concepts of the categories theory are rather familiar to the reader since we will not push far their use. So, we give only unformal explanations.

Rules generating the basic tiles:

$$\frac{R(\alpha)=(s,a,b,t)}{\alpha : s \xrightarrow[b]{a} t}$$

Rules generating the horizontal and vertical identities:

$$\frac{t : x \to y \in H}{id_H : t \xrightarrow[y]{x} t} \qquad \frac{a : x \to z \in V}{id_V : x \xrightarrow[a]{a} z}$$

Horizontal, vertical and parallel compositions

$$\frac{\alpha : s \xrightarrow[b]{a} t \quad \beta : h \xrightarrow[c]{b} f}{\alpha * \beta : s;h \xrightarrow[c]{a} t;f} \qquad \frac{\alpha : s \xrightarrow[b]{a} t \quad \beta : t \xrightarrow[d]{c} h}{\alpha \bullet \beta : s \xrightarrow[b;d]{a;c} h}$$

$$\frac{\alpha : s \xrightarrow[b]{a} t \qquad \beta : h \xrightarrow[d]{a} f}{\alpha \otimes \beta : s \otimes h \xrightarrow[b \otimes d]{a \otimes c} t \otimes f}$$

figure3: Inference Rules of tile logic

2.3 Tile system construction algorithm

The specification of a consistant tile model needs to adopt a methodology of six steps [14] summarized on the following algorithm :

A- Define the set of basic system configurations.

B- Define the interfaces of the basic configurations.

C- Define the basic events we want to observe and their interfaces according to the steps A and B.

D- Repeat if necessary A, B and C until the basic structures are entirely defined.

E- Define the set of the tiles describing the basic behaviors of the system according to the framework chosen in A, B and C.

F- Reiterate D and E until obtaining a consistent definition of a tile system.

3. A tile system for a video conference interface

3.1 Video conference interfece presentation

A video conference is a conversation between two or several interlocutors (one president and two or more participants), possibly located at different sites, which can speak each other, see each other and share working documents thanks to multimedia communicating computers via a communication network. Obviously, the most significant entrance point to such a system is its interface allowing the interaction between users and system through graphic elements (windows, icons and menus) and reactive actions. This type of graphic interface called WINP (Windows Pointing) [19] has achieved very quickly the favour of the users from the associated ergonomics (easy use, concision, coherence, flexibility, errors prevention, helps and follow-up, etc). Two aspects characterize such an IHM and are inspired from the ‹‹Microsoft Net Meeting›› system interface: the static aspect and the dynamic aspect. The first aspect is defined by a set of windows with their components, and of functionalities, whereas the second represents the dynamic behavior of the reactive system defining the interface by a set of states and a set of actions allowing the transition between these states. This on the one hand, on the other hand, this interface is in fact composed of one president interface and as many participant interfaces as participants.

Although, various windows are accessible via a video conference application such as application issuing, video, dialogue, exchange of files, notebook, white board, we restrict our study by considering only the principal and conversation windows. The principal window of the president PWPr (represented by the buttons b1, b2, b3 and b4 considered as subcomponents) gives the president the total control on the video conference in addition to participant's fonctionalities. Only one participant can take a speech at once. The principal window of a participant, gives to this one the access to the other functional windows. But, in our case, for simplification reasons, we consider only the closure window button. Any conversation window then allows to manage the communication using messages.

Values attributed to buttons represent their possible states. For examle, the value 0 for b1 corresponds to the state "no participant is chosen by the presedent", while the value 1 or 2 says that "the participant 1 or 2 rspectivly is chosen" and the value # corresponds to the state"no participant can be chosen as the application ends". For b2, 0 corresponds to "speech not given to the chosen participant" and 1 to the opposite state. The button b7 or ib7 allows to identify the received or transmited message destination.

Likely, we deduce the functions and the values of the other buttons from tables 1 and 2 that summarize with figure4, the static aspect.

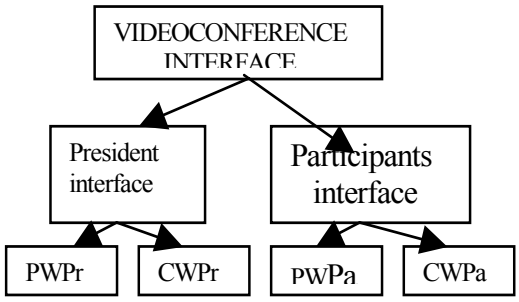

Figure4: Video conference components

In both following tables, m, n are strings and i and j are naturals.

	Button code	Possibles Values	effect	Effect code
P	b1	0, 1, 2, #	Choose participant i.	c1(i)
W	b2	0, 1	Give speech.	c2
P	b3	0, 1	Take speech.	c3
r	b4	0, 1	Close window	c4
C	b5	0, 1	Seize message.	c5(m)
W	b6	0, 1	Receive message.	c6(n)
P	b7	0, 1, 2	Identify participant i.	c7(i)
r	b8	0, 1	Send message.	c8

Table 1: President Interface

	Button code	Possible values	effect	Effect code
P W P a	ib4	0, 1	Close window.	ic4
C	ib5	0, 1	Seize Message.	ic5(m)
W	ib6	0, 1	Receive message.	ic6(n)
P	ib7	0, j j≠i	Identify participant j.	ic7(j) j≠i
a	ib8	0, 1	Send message.	ic8

Table 2: Participant Interface

Note: We prefer to use a quite simple code in order to simplify tiles writing during the interface formalization.
A button click is coded by the letter c for the president and by ic for a participant i.

3.2 Interface formalization.

We apply the algorithm of paragraph **2.3** to define the dynamics of our interface. We will present a model equipped with the notion of observation able to capture the causal dependences and localities : Tile model where the observations and configurations are similar algebraic structures. The category of configurations is a term algebra of states, and the category of observations is a free monoid of actions strings A tuple of terms (with variables) represents a component or a configuration and can be seen as an arrow of a Cartesian category where the compositions ⊗ and **;** correspond respectively to the tupling of terms and the term substitution [8].
To avoid a great number of notations, each basic component state is given using the associated code as a fonction to apply on its possible values.

3. 2. 1. The configurations and their interfaces

An interface configuration (or a state) is defined by a combination of the various displayed windows states that we note by:
ivc = ipr ⊗ ipa1⊗ ipa2
As we consider one president and two participants, the corresponding interface states ipr, ipa1 and ipa2 are also a combination of their windows states. We will use then, the following notations:
ipr = pwpr ⊗ cwpr
Each window state pwpr or cwpr is also a combination of the states of its composing buttons (see tables 1 and 2). We note a button state by its name as a function applied to its value.
pwpr = <b1(v1), b2(v2), b3(v4), b4(v4)>
cwpr = <b5(v5), b6(v6), b7(v7), b8(v8)>
In the same way, we define the notations corresponding to a participant j interface:
ipaj = pwpa ⊗ cwpa
pwpa = < ib4(iv4)>
cwpa = <ib5(iv5), ib6(iv5), ib6(iv6), ib7(iv7), ib8(iv8)>
Note: To exploit to the maximum the principle of compositionnality of the applied formalism, we should:
Phase1: treat the evolution of each subcomponent independently of the others, then,
Phase2.: treat the evolution of the interface of each interlocutor independently of the others also, and finally,
phase3: see the case of the global interface.
Nevertheless, since our major objective is to show the importance of the tile logic in the natural description of the interface and

with the aim of facilitating the assimilation to the reader considering the space limitation constreint, we will directly present a sufficient part of phases 2 and 3.

President Configurations

Possible values of pwpr are :

pwpr0=<0, 0, 0, 0> ready state
pwpr1=<i, 0, 0, 0> participant i chosen
pwpr2=<i, 1, 0, 0> speach given to i
pwpr3=<i, 0, 1, 0> retaking speach
pwpr4=<i, 0, 1, 1> window closure
pwpr#=<#, 0, 0, 0> final state

Possible values of cwpr are :

cwpr0=<0, 0, 0, 0> ready state
cwpr1=<1, 0, j, 0> j≠0 message seizen
cwpr2=<1, 0, j, 1> message sent
cwpr3= <0, 1, j, 0> message received
cwpr4= <1, 1, j, 0> copy received message in seizeing field

Composing the states pwpr0, pwpr1 and pwpr2 with all cwpr then pwpr3, pwpr4 and pwpr# with cwpr0, we obtain the configurations ei 0≤ i ≤15 and e# as final configuration (application end) corresponding to different values of ipr.

Examples: e0=<0, 0, 0, 0> ⊗ <0, 0, 0, 0>
e1=<0, 0, 0, 0> ⊗ <1, 0, j, 0>
e5=<i, 0, 0, 0> ⊗ <0, 0, 0, 0>
e#=<#, 0, 0, 0> ⊗ <0, 0, 0, 0>

We determine the configurations of the participant interface in the same way.

Participant Configurations

The possible values of ipwpa are:

ipwpa0=<0> ready state
ipwpa1=<1> window closure

The possible values of icwpa are

icwpak= cwprk 0≤k≤3, j≠i
icwpa#= < 0, 0, #, 0> final state

Composing the state ipwpa0 with all the states icwpak, 0≤k≤3 and K = # and ipwpa1 with cwpr0, we obtain the configurations isj 0≤ j ≤8 and is# like final configuration (application end) corresponding to the various possible values of ipa1 or ipa2.

Note : Each state of any button can be an interface for the configuration in which it appears.

3. 2. 2. The observations and their interfaces

The effects of the clicks on the buttons of the various windows given in tables 1 and 2 are all in fact of the basic observations. In tables 3, we present an additive list of observations necessary to describe well the evolution of the interface

Observation	Code (president)	Code (participant)
Copy or consume received message	c9	ic9
Delete received message	c10	
Finish sending message.	c11	ic11
Stop application	c12	ic12

Table 3: Other observations

Note: The observation, Stop application, is in fact, Leave application, for a participant. The observation ic10 is not defined because the participant consumes each received message.

Intuitively, the interfaces represent points of connections (entered and left) between different configurations of the system and between consecutive observations of the same component. We then present them through the effect of the various observations in the table 4.

Examples:

The observation c1(j) (choose participant j) for j=1 or j=2 applied to b1(0) gives like result b1(j) and to b1(j), it gives the result b1(0).

The observation c9 is an imaginary click that corresponds to an action copying or consuming a received message indicated by b6(1). If no message seizure is in hand b7(0), the president consumes the message and the output interface will be (b5(0), b6(0), b7(0)), else b7(j), the president recopies the message in the fields of seizure of the conversation window and the result will be (b5(1), b6(1), b7(j)). In the

case of a participant i, ic9 allows to consume the message only.

observ	input	output
c1(j)	b1(0) b1(j)	b1(j) b1(0)
cj 2≤j≤8 & j≠7	bj(0) bj(1)	bj(1) bj(0)
c7(j)	b7(k)	b7(j)
c9	b5(0), b6(1), b7(0) b5(0), b6(1), b7(j)	b5(0), b6(0), b7(0) b5(1), b6(1), b7(j)
c10	b5(1), b6(1)	b5(1), b6(0)
c11	b5(1), b7(j), b8(1)	b5(0), b7(0), b8(0)
c12	b1(0), b3(0), b4(0) b1(i), b3(1), b4(1)	b1(#), b3(0), b4(0) b1(#),b3(0),b4(0)
icj j=1 or 4≤j≤8 et j≠7	ibj(0) ibj(1)	ibj(1) ibj(0)
ic7(j)	ib7(k)	ib7(j)
ic9	ib6(1)	ib6(0)
ic10	ib6(1), ib7(j)	ib6(0), ib7(0)
ic11	ib5(1), ib8(1)	ib5(0), ib8(0)
ic12	ib7(j)	ib7(#)

Table 4: Basic I/O Interfaces
Observ: observation

3. 2. 3 The tiles

3. 2. 3. 1 President Interface

The basic tiles describing the configurations changes of the president interface will have the form:

$$tpr_r : ek \xrightarrow[b]{a} el$$

r ≥ 1, 0≤ k ≤15, 0≤ l ≤ 15 or l = #

$$tpr_1 : e0 \xrightarrow[c7(j)]{c5(m)} e1$$

$$tpr_2 : e0 \xrightarrow[c7(j)]{c6(m)} e3$$

$$tpr_3 : e3 \xrightarrow[id]{c9} e0$$

$$tpr_4 : e1 \xrightarrow[c11]{c8} e0$$

$$tpr_5 : e3 \xrightarrow[c10]{c9} e1$$

$$tpr_6 : e0 \xrightarrow[id]{c1(i)} e5$$

$$tpr_7 : e5 \xrightarrow[c7(j)]{c5(m)} e6$$

$$tpr_8 : e6 \xrightarrow[id]{c9} e5$$

$$tpr_9 : e6 \xrightarrow[c10]{c9} e5$$

$$tpr_{10} : e5 \xrightarrow[c7(j)]{c6(m)} e8$$

$$tpr_{11} : e8 \xrightarrow[c10]{c9} e6$$

$$tpr_{12} : e8 \xrightarrow[id]{c9} e5$$

$$tpr_{13} : e5 \xrightarrow[id]{c2} e10$$

$$tpr_{14} : e10 \xrightarrow[c7(j)]{c5(m)} e11$$

$$tpr_{15} : e11 \xrightarrow[c11]{c8} e10$$

$$tpr_{16} : e10 \xrightarrow[c7(j)]{c6(m)} e13$$

$$tpr_{17} : e13 \xrightarrow[id]{c9} e10$$

$$tpr_{18} : e12 \xrightarrow[c10]{c9} e10$$

$$tpr_{19} : e0 \xrightarrow[id]{c12} e\#$$

$$tpr_{20} : e10 \xrightarrow[id]{c3} e15$$

$$tpr_{21} : e15 \xrightarrow[id]{c1(j)} e5$$

$$tp_{22} : e15 \xrightarrow[c12]{c4} e\#$$

By composing these basic tiles, we can determine other tiles describing logical sequences of local behaviors of president interface as it is shown in the following examples:

Seizeing and sending a message:

tpr_{23} : tpr1 ● tpr_3

Receiving and consuming a message :

$tpr_{24} : tpr2 \bullet tpr4$

receiving and retransmiting a message :

$tpr_{25} : tpr2 \bullet tpr5 \bullet tpr4$

3. 2. 3. 2 Participant interface

The basic tiles describing the configurations changes of a participant i interface will have the form:

$$tpa_r : isk \xrightarrow[b]{a} isl$$

r ≥1, 0≤ k ≤4, 0≤ l ≤ 4 or l = #

$$itpa_1 : is0 \xrightarrow[ic7(j)]{ic5(m)} is1$$

$$itpa_2 : is0 \xrightarrow[ic7(j)]{ic6(m)} is3$$

$$itpa_3 : is1 \xrightarrow[ic11]{ic8} is0$$

$$itpa_4 : is3 \xrightarrow[id]{ic9} is0$$

$$itpa_5 : is0 \xrightarrow[ic12]{ic4} is\#$$

We notice that the number of the tiles in this case is limited due to the limitation of the conversation types for a participant. As

in the case of the president interface, other tiles can be built such as: Seizeing and sending a message:

$itpa_6 : itpa1 \bullet itpa3$

Receiving and consuming a message:

$itpa_7 : itpa2 \bullet itpa4$

3. 2. 3. 3 Video conference interface

The tiles describing the configurations changes of all the video conference interface with an initial state ivc0=e0⊗1s0⊗2s0 and a final state ivc#=e#⊗1s#⊗2s#, will be noted by tiv_k, k≥1 and will be obtained by applying the tile composition rules. We must take into account all the interactions and obviously their synchronization. In our paper, we cannot treat all the cases but, we show how to build a tile system by considering a case of interaction.

Case 1: We suppose that the participant is able to recognize the veritable sender of any received message. If we suppose that participant 1 (1pa) seizes and sends a message to participant 2 (2pa), the president (Pr) must then receive the message and retransmit it to the 2pa (because any message must pass through the president). In this case, the behavior of

- 1pa is descibed by the tile $1tpa_6$ such as

$1tpa_1 : 1s0 \xrightarrow[ic7(2)]{ic5(m)} 1s1$ and

$itpa_3 : is1 \xrightarrow[ic11]{ic8} is0$ with i=1

- pr is descibed by the tile tpr_{25} such as

$tpr_2 : e0 \xrightarrow[c7(2)]{c6(m)} e3$ and

$tpr_2 : e0 \xrightarrow[c7(j)]{c6(m)} e3$

- 2tpa is descibed by the tile $2tpa_7$ such as

$2tpa_2 : 2s0 \xrightarrow[ic7(2)]{ic6(m)} 2s3$ and

$itpa_4 : is3 \xrightarrow[id]{ic9} is0$ with i=2

We note that 4 tiles are necessary to carry out the synchronization of these tiles for describing the interaction:

- A tile providings the trigger c6(m) of tpr_{25}:

$tiv_1 : 1s0 \otimes e0 \otimes x \xrightarrow[c6(m)]{1c7(2);1c11} 1s0 \otimes e0 \otimes x$

where x is a variable representing an unspecified state of ipa2.

- A tile describing the synchronized behaviors of 1pa and Pr:

$tiv_2 : tiv_1 * tpr_{25}$

-A tile providing the trigger 2c6(m) of $2tpa_7$

$tiv_3 : 1s0 \otimes e0 \otimes 2s0 \xrightarrow[2c6(m)]{c7(2);c10;c11} 1s0 \otimes e0 \otimes 2s0$

- A tile synchronizing the behavior of the 3 interventions

$tiv_4 : tiv_3 * 2tpa_7$

Cases 2: The tile that describes the reception and the retransmission of a message by pr in parallel with the seizure and the emission of a message by 2pa can be writen as follows:

$tiv_5 : tpr_{25} \otimes 2tpa_6$

The interfaces are deduced according to the composition rules applied and the observations that there participate.

4. Conclusion

In this paper, we contribute to the suggestion of the tile logic as powerful formalism with an expressive semantic well adapted to the open and reactive systems, for specifying formally an IHM. Our study case being the interface of a video conference, the specification has been realized according to well structured steps to define clearly the configurations, the observations with their interfaces and a tile system. The latter describes in very natural manner the evolution of the interface while offering the possibility to observe interactions and the composition of the components through their behavior. We have considered a restricted interface, for clarity, conciseness and simplification reason. Nevertheless, it is clear that through the model presented, its generalization to more realistic cases is very feasible as the system is open. Compared with the ordinary rewriting rules that freely can be instanciated, the tiles have the advantage of to impose rewriting all while taking the environment into account. This would resolve certainly the problem of the combinatorial explosion of the states during the verification. Thus, we

confirm the expressive strength of the applied logic to describe the dynamics of a system and more generally, the generalization of the rewriting logic as semantic and logical framework of the concurrency. We also recognize particularly, the richness in observational semantic of tile logic. This work encourages us to continue it by the translation of the resultant specification in an executable specification in rewriting logic through its Maude language [18] while exploiting the reflection and the meta strategies [17] to control the rewritings and while exploiting the works of U. Montanari, J. Meseguer and R. Bruni [14, 15, 16, 17].

References:

[1] M. Baron, ”Vers une approche sûre du développement des Interfaces Homme-Machine”, Thesis, LIP6, 2004.

[2] J. Meseguer, ”Rewriting as a unified model of concurrency”, Technical Report SRI-CSL-90-02R, SRI International, Computer Science Laboratory, 1990.

[3] J. Meseguer, “Conditional Rewriting Logic as a unified model of concurrency”, Theoretical Computer Science, 1992, pp.73-155.

[4] F. Gadducci. and U. Montanari, “The tile model”, G. Plotkin, C. Stirling, & M. Tofte editorss, Proof, Language and Interaction: Essays in Honour of Robin Milner,. MIT Press, Cambridge, MA 2000.

[5] F. Gadducci and U. Montanari., “Tiles, Rewriting Rules and CCS”, J. Meseguer editor, Proceeding of First International Workshop on Rewriting Logic and its Applications, vol. 4 of ENTCS, Elsevier Editor, 1996.

[6] R. Bruni and U. Montanari , “An Interactive Semantics of Logic Programming”, Theory and Practice of Logic Programming, Vol. 1, 2001, pp. 547-690.

[7] R. Bruni,, J. Meseguer and U. Montanari, **“**Tiling Transactions in Rewriting Logic”, in ENTC, Vol. 71, 2003, 20 pages.

[8] G.L., Ferrari and U. Montanari, “Tile Formats for Located and Mobile Systems”, Informatica and.Computing, Vol.156, 2000, pp.173–235.

[9] R. Bruni, J. Meseguer, U. Montanari, “Dynamic Connectors for Concurrency”, Theoretical Computer Science, 2002, Vol 281, pp.31 – 176.

[10] R. Bruni, “Tile logic for Synchronized Rewriting of Concurrent Systems”, Ph.D. Thesis, Computer Science department, University of Pisa, 1999.

[11] R. Bruni, J. Meseguer, U. Montanari, “Process and term tile logic”, Technical Report SRI-CSL-98-06, SRI International, 1998. Also Technical Report TR-98-09, Computer. Sci.ence Department, University of Pisa.

[12] N. M. Oliet and J. Meseguer, ”Rewriting logic: roadmap and bibliography”, TCS , Rewriting logic and its applications Elsevier Science Publishers, 2002, pp.121–154.

[13] R. Bruni and J. Meseguer ”Generelazed Rewrite Theories”, In Proceeding of Thirtieth International Colloquium on Automata, Languages and Programming, Springer-Verlag LNCS, Vol 2719, 2003, pp.252-266.

[14] J. Meseguer and U. Montanari, “Executable tile Specifications for Process calculi”, in: J.-P. Finance Editor, Proceeding of FASE’99, Fundamental Approaches to Software Engineering, LNCS, Vol. 1577, Springer, Berlin, 1999, pp.60–76.

[15] J. Meseguer and U. Montanari, “Mapping Tile Logic into Rewriting Logic”, Proceedings of wadt’97, 12th workshop on recent trends in algebraic development techniques, LNCS, Vol. 1376. Springer Verlag, 1997, pp.62–91.

[16] R. Bruni, J. Meseguer, U. Montanari, “Implementing Tile Systems: Some Examples from Process Calculi”, Proceedings 6th ICTCS'98, World Scientific, 1998, pp.168-179.

[17] R. Bruni, J. Meseguer, and U. Montanari, “Internal Strategies in a Rewriting Implementation of Tile Systems”, Claude Kirchner and Helene Kirchner Editors, International Workshop on Rewriting Logic and its Applications, ENTCS, Vol. 15, Elsivier, Amsterdam, 1998, pp.95-116.

[18] F. Durant, J. Meseguer, "Parametrazed Theories and Views in Full Maude 2.0", ENTCS 36 ,Elsivier, Amsterdam, 2000, pp.319-337.

[19] S. Imai, S.Konno, T. Suganuma, and T. Kinoshita, “Design and Implementation of Knowledge-based Video Conference System”, 17th International Conference on advanced Information Networking and Applications (AINA’03), Tohoka University, Xi’an, China, march 2003.