Explicit Modeling of Influences, and of Their Absence, in Distributed Systems

Horst F. Wedde and Arnim Wedig

Informatik III, University of Dortmund
D-44221 Dortmund, Germany
{wedde,wedig}@ls3.cs.uni-dortmund.de

Abstract. Specific problems in practical distributed system design arise from *incomplete information* about the cooperation requirements, up to, or even beyond, the final design stage. Events in components *will* occur, or they *may* occur, depending on (local) user decisions. The latter *may* also *not* occur, as a result of yet unknown external influences or design faults. Adequate formal modeling tools should allow for distinguishing between such different event types. Our approach for this purpose to be introduced here is the *formal model of I–Systems*. As a particularly relevant and unique feature, the presence as well as the absence of interactional influences (as part of distributed cooperation requirements) can be *explicitly* modeled, with no side effects. A non-trivial synchronization problem is modeled *incrementally* in order to demonstrate both the modeling and analysis capabilities in I–Systems.

1 Introduction

In modeling distributed systems, specific problems arise from *incomplete information* up to, or even beyond, the final design stage, as much as from the *distribution of control*. While the structure or the behavior of components could be considered manageable this is different for distributed systems. Here conditions and constraints on the cooperation between components are local, i.e. they concern small subsystems of interacting components while typically imposing a global propagation effect that is often not known at the design phase, or even undesirable. Thus expected or needed events in a subsystem may never occur because some of its preconditions may never hold, due to unforeseen propagation effects from restrictions elsewhere. In such cases models turn inadequate during test or debugging phases unless appropriate formal tools were available to trace design or implementation failures back to their origin.
Another complication under distributed control is that the components' behavior, as observed from an interacting component, exhibits two different types of transitions: some *will* occur, some *may* (or may never) occur. The reasons for the latter behavior are not observable from the interacting components. Events that *will* occur will do so (unless prevented through external influences) for two reasons:

J.-P. Katoen and P. Stevens (Eds.): TACAS 2002, LNCS 2280, pp. 127–141, 2002.

- according to the local operational semantics (e.g. for program execution) which are otherwise comparable to *organizational duties* imposed on the employees in a company;
- through implemented external influences that trigger the events.

Events that *may* occur come from two sources of influences:

- Local decisions, e.g. by a user in an interactive mode of operation;
- Incomplete information about unknown, or unpredictable, external influences (as discussed in the previous paragraph).

In order to cope with the problems mentioned and at the same time to develop a both reliable and incremental modeling methodology our approach is based on elementary binary relations between system components and their operational semantics (*action rules*). These specify the restrictive effect on the involved components.
Based on the observation that components in distributed systems may be *passive/ reactive* (like main memory management in a uniprocessor operating system) or *active* (like in an interactive user operating mode), we distinguish in our terminology between *inert* and *autonomous* components, respectively.

Previous and Related Work. In [12] W. Reisig has considered two classes of transitions in Petri Nets specifying *progressive* actions (that *will* occur in the terms of our discussion) and *quiescent* actions (that may occur), respectively. The firing of quiescent transitions depends on information that has not been available at the modeling stage. If a quiescent transition does not fire, however (e.g. because a precondition can never be satisfied, due to a synchronization condition which might be too restrictive), this cannot, by definition of quiescence, be traced back to the originating cause. For the novel trace semantics that we have defined for I-Systems (local) influences and their global propagation are specified in such a way that external influences can be traced back to the originating system component or the originating behavioral restriction (whatever applies). Even the absence of influences (restrictions) is clearly visible from the specification. Reisig's work is the only approach related to our work. While the absence of influences is particularly required in synchronization constraints I–Systems are the only formal structures, up to our knowledge, that allow for explicitly modeling and analyzing such constraints incrementally. Earlier results of our theoretical work were reported in [11] and [13]. Additional information about interactive systems can be found in [3,16,17].

Organization of the Paper. The next section gives a brief survey on I–Systems, their basic concepts, semantics, and those behavioral properties that are needed in the subsequent sections. The main feature in section 3 is a general theorem on representing and at the same time generating an arbitrary local behavior structure through an elementary interaction with a passive component. Section 4 is devoted to demonstrating the incremental potential of I–Systems by stepwise modeling a solution for a non-trivial synchronization problem that includes requirements about the absence of external influences. Conclusions are briefly summarized at the end.

2 I–Systems

In order to model events in a distributed system as well as the cooperation between its components we define I–Systems. System components are presented as *parts*. These are constituted through their local states that are relevant for the interaction. The local states are called *phases*. The only thing we assume about the components is that they are in exactly one state at any time. The interaction is specified through two binary relations, denoted as *coupling* and *excitement relations*, respectively.

Definition 1 (I–System). A structure $IS = (P, B, \underline{B}, K, E)$ is called *I–System*, iff:

(1) P is a finite set of *phases*
(2) B is a set of *parts* with:
  a) $\forall b \in B : b \subseteq P$
  b) $\bigcup_{b \in B} b = P$
  c) $\forall b_1, b_2 \in B, b_1 \neq b_2 : b_1 \cap b_2 = \emptyset$
(3) $\underline{B}$ is a distinguished set of so-called *inert parts* ($\underline{B} \subseteq B$)
(4) $K \subseteq P \times P$ is the *coupling relation of* IS with:
  a) $K = K^{-1}$
  b) $\forall b \in B \, \forall p, p' \in b, p \neq p' : (p, p') \in K$
(5) $E \subseteq P \times P$ is the *excitement relation of* IS with
  a) $E \cap (E^{-1} \cup K) = \emptyset$

For $p \in P$, $b(p)$ denotes the part of p with $b(p) = b'$ iff $p \in b' \in B$.
$AB(IS) := B \setminus \underline{B}$ is called the *set of autonomous parts*. □

Fig. 1 depicts an I–System IS_1. The small circles are the phases (e.g. p_1, v_2) grouped by ovals, the parts. The parts may have names (e.g. b_1, b_2). The inert parts (e.g. $\underline{b}_3$) have gray fillings or/and an underlined name. Arrows (e.g. from p_3 to q_2) depict the excitement relation and lines between phases (e.g. from p_1 to v_1) the symmetrical coupling relation.

Remark 1. *Inert parts* represent special components in distributed systems. They are found e.g. in reactive systems and reactive software components (see e.g. [2,5,9]). Events in inert parts are always triggered by environmental influences. In the absence of external stimuli nothing will happen in such components. □

The symmetrical coupling relation K specifies the subset of pairs of *mutually exclusive* phases. Condition 4.b implies therefore that two different phases in a part exclude each other from holding at any given time. The coupling relations between phases in the same part are omitted in graphical representations.

Based on local states of components we define global system situations by making use of the causal relationships between parts given by K. We term the

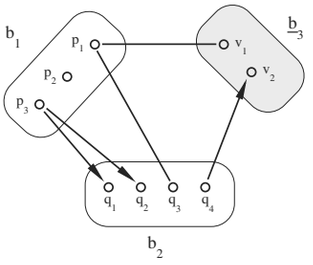


Fig. 1. I–System IS_1

new objects *cases*. A case is a subset c of phases such that $|c \cap b| = 1$ for all $b \in B$ and $(p_1, p_2) \notin K$ for all $p_1, p_2 \in c$. The phases of a case c may hold *concurrently*.
$\{p_1, q_4, v_2\}$ and $\{p_3, q_3, v_1\}$ are cases of the I–System in Fig. 1. $\{p_1, q_3, v_2\}$ is not a case because p_1 and q_3 are mutually exclusive phases.

Events are described by *phase transitions* which lead from one case to another. A phase transition $p \to q$ *may occur* in a case c iff $(c \setminus \{p\}) \cup \{q\}$ is a case. Two phase transitions $p_1 \to q_1$, $p_2 \to q_2$ are called *concurrent in a case* c iff each transition may occur in c and $(q_1, q_2) \notin K$. Two concurrent phase transitions may occur in arbitrary order or simultaneously. Thus they are causally independent, and according to the definition this global causal relationship can be checked by testing *local* static relations.
In the case $\{p_2, q_1, v_1\}$ the phase transition $p_2 \to p_1$ can never occur. In the case $\{p_2, q_1, v_2\}$ of IS_1, the phase transitions $p_2 \to p_1$ and $q_1 \to q_2$ may occur and they are concurrent.

While the coupling relation enables, in a given case, a phase transition to occur it is not guaranteed that it eventually *will* occur. In order explicitly to cover such kind of progress quality we introduce *forces* arising from external influences which enforce phase transitions to occur. Enforced phase transitions *will* occur unless prevented through different external influences. An element (p, q) in the *excitement relation* E expresses a potential excitation from phase p in part b to phase q in part b'. If p and q belong to a case c then b exerts a force on b' to leave p. The other idea behind is that the force is released only after the excited phase has been left. No force vanishes but by b' leaving the excited phase.
For the case $\{p_3, q_2, v_2\}$ of IS_1 in Fig. 1, b_1 influences b_2 to leave q_2. b_1 has to stay in p_3 as long as b_2 is in q_2. In $\{p_2, q_2, v_2\}$ no exciting influence is external.

In order to formalize the effects of influences and to distinguish between phase transitions that *may* and phase transitions that *will* occur, our idea is to assign certain qualities of activity (*phase qualities*) to every phase p, thus refining the concept of case to *global activity state* in which the phase qualities are reflected.

Definition 2 (Local / Global Activity State). Let $IS = (P, B, \underline{B}, K, E)$ be an I–System and $b \in B$. The mapping $z\langle b\rangle : b \longrightarrow b \cup \{0, 1, \mathsf{F}\}$ such that
(L1) $\forall p, p' \in b : (z\langle b\rangle(p) = p' \Rightarrow p' \notin \{p, 0, 1, \mathsf{F}\} \wedge b(p) \in AB(IS))$
(L2) $\exists!\, p \in b : z\langle b\rangle(p) \neq 0$
is called *local activity state* of b.
$LState(b)$ denotes the set of all local activity states of b. For a phase p in part b we interpret the *phase qualities* in the following way:

$$z\langle b\rangle(p) = \begin{cases} q : b \text{ is in } p,\ b \text{ has made the (control) decision to enter } q. \\ \mathsf{F} : b \text{ is in/ assumes } p,\ b \text{ is } \textit{unstable in } p \text{ (as an effect of an external influence), } b \text{ is to take action to leave } p. \\ 1 : b \text{ is in/ assumes } p,\ b \text{ is } \textit{stable in } p \text{ (no decision has been made, and there is no necessity to leave } p \text{ as a result of some external influence).} \\ 0 : b \text{ is not in/ does not assume } p. \end{cases}$$

If $B = \{b_1, \ldots, b_n\}, n \in I\!N$ and $z\langle b_i\rangle \in LState(b_i)$, $i = 1, \ldots, n$, then the mapping $z : P \longrightarrow P \cup \{0, 1, \mathsf{F}\}$ such that
(G1) $z|_{b_i} = z\langle b_i\rangle$
(G2) $\forall p, p' \in P : (z(p) \neq 0 \wedge z(p') \neq 0) \Rightarrow (p, p') \notin K$
is called *global activity state* of IS.
$z|_{b_i}$ is the restriction of z to the part b_i. $GState(IS)$ denotes the set of all global activity states. In this terminology, a global activity state z is a tuple of local activity states. □

According to Remark 1: In our terms, phase transitions in inert parts are always effects of external influences. Inert parts cannot make any decision.

Definition 3 (free / exciting). If a part b does not assume p in a global activity state z with $p \in b$ then we say p *is free in* z iff p is not coupled to any phase p', such that $z(p') \neq 0$ and $p' \notin b$: $(p, p') \notin K$.
If part b assumes p in z and another part b' assumes p' in z and $(p, p') \in E$ then we say that p *is exciting* p' *in* z. □

Action rules formulated as distributed algorithms (the syntax is similar to that in [10]) specify the interaction between the parts of an I–System and compute changes of phase qualities *in a single part. From a global view* the action rules specify how, starting from a global activity state z, the successor states z' would be derived, making use of Definition 3. Because of the strict page limitation we do not present the action rules. These can be found as an appendix in [15]. In the sequel we will present the basic ideas.

Two successive global activity states $z_1.z_2$ represent a specific activity (specified by the action rules) in the parts of an I–System IS. Let p, q be phases of a part b of IS.

AC1. If $z_1(p) = 1$ and $z_2(p) = q$ then we say that b *makes a decision* to enter q. b *may* make a decision or may not. b must be an autonomous part (def.1). b is bound to the decision, and a phase transition from p to q *will* occur unless q is not a free phase in z_1. If q is not free, b influences every part b' of IS with b' beeing in a phase v ($z(v) \neq 0$) and $(v, q) \in K$ to leave v. After every such b' has left v, b will enter q.

AC2. If $z_1(p) = 1$ and $z_2(p) = \mathsf{F}$ then we say that b *becomes unstable*. The cause for this unstability is either an excitation (it exists a phase v with $z_1(v) \neq 0$ and $(v, p) \in E$) or an influence from another part b' to leave p, because $(p, v) \in K$ and v is not free (AC1 or AC2). b *will* eventually leave p unless there exists no free successor phase. In the latter case, b influences every part b' of IS to leave a phase v if b' is in v and $(v, q) \in K$ for any $q \in b$.

AC3. If $z_1(p) = q$ and $z_2(p) = 1$ then we say that *the decision* of b to enter q *is cancelled*. In this case p is exciting a phase w assumed by a part b'.

AC4. If $z_1(p) = q$ and $z_2(q) = 1$ then we say that a *free phase transition* from p to q occurs in b. We call that transition free because it depends on a decision of b that *may* occur anytime (AC1).

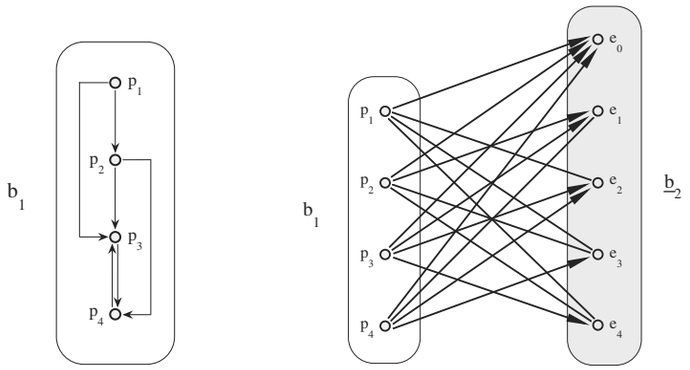
AC5. If $z_1(p) = \mathsf{F}$ and $z_2(q) = 1$ then we say that an *enforced phase transition* from p to q occurs in b. We call that transition enforced because it originates from a condition of unstability that will occur inevitably (AC2).

A recursive application of AC2 represents a global propagation effect of local conditions. We call the influences in AC1 and AC2 (b influences b' to leave v because b takes action to enter a phase that is not free and coupled to v) *propagated influences*.

A sequence of global activity states as permitted by the action rules defines a *run* in the I–System. The set of all such sequences forms the *behavior* of the I–System.

Definition 4 (Behavior of an I–System). Let IS be an I–System. We define the *behavior* $\mathcal{V}[\![IS]\!]$ of IS as:
$\mathcal{V}[\![IS]\!] = \{z_0 z_1 z_2 \ldots \mid z_0 z_1 z_2 \ldots$ represents a run in IS with $z_i \in GState(IS)$ and $z_i \neq z_{i+1}$ for $i = 0, 1, 2, \ldots\}$. □

A complete and more formal definition can be found in [15]. Properties of the trace semantics 'behavior' (see [15]) can be used for proving that the action rules are complete and are not contradicting each other. As mentioned in section 1, Reisig [12] distinguishes between between two different kinds of actions: 'progressive' ones that will occur and 'quiescent' ones that may occur. Please remember that if a quiescent action does not occur the cause can not be traced back in general. If, for an I–System IS, two successive global activity states in a trace of the behavior $\mathcal{V}[\![IS]\!]$ represent a free or an enforced phase transition (AC4 or AC5, respectively) then we are able to trace back to the local causes (AC1 or AC2, respectively).



(a) $G(b_1)$ (b) IS

Fig. 2. Interaction inducing a state/transition graph

3 Example: Sequential Processes with Enforced and Free Phase Transitions

From the viewpoint of distributed, interacting system components each of them has a certain amount of *autonomy*. (In the extreme case of technical components, some may be passive, though.) Beyond the interaction with other components there may exist an *internal organizational scheme* specifying the component's process structure. On the human administration level such schemes are typically described as *role-based* state/event structures (e.g. in distributed security, see [4,14]). Part of a role is *decision making*. Other actions are the result of *organizational duties*. Each action belongs to one of these classes.
In I–Systems, actions occur in phases (local states). When a decision has been made then *all possible* actions in this phase are considered autonomous (*decision situations*). In every other situation *all* actions meant to be *organizationally enforced* (*procedural situations*). The formal approach discussed in section 2 follows exactly this organizational context. In the components (parts) there are two types of local situations (phases). Transitions from one type of phases are free (see AC4), the other phases allow for enforced transitions only (see AC5). In this way an organizational scheme for the phases of any part is defined. (External influences may superpose the local organizational scheme.)

Definition 5 (Organizational Scheme). A phase/transition structure in a part b of an I–System consisting of decision and procedural situations will be called an *organizational scheme for* b. □

If one disregards the difference between decision and procedural situations an organizational scheme is just a state/transition graph (see Fig. 2.a). As a possible interpretation let us assume that every transition in a phase/transition

structure is free. Then the following general representation theorem expresses that the behavior in a given part b_1, specified through a directed graph structure over b_1 (e.g. $G(b_1)$ in Fig. 2.a), can be interpreted as originating from a simple interaction between b_1 and an additional inert part $\underline{b}_2$ where $|\underline{b}_2| = |b_1| + 1$ (see IS in Fig. 2.b). $\underline{b}_2$ is only interconnected to b_1, so $\underline{b}_2$ cannot be influenced from elsewhere.

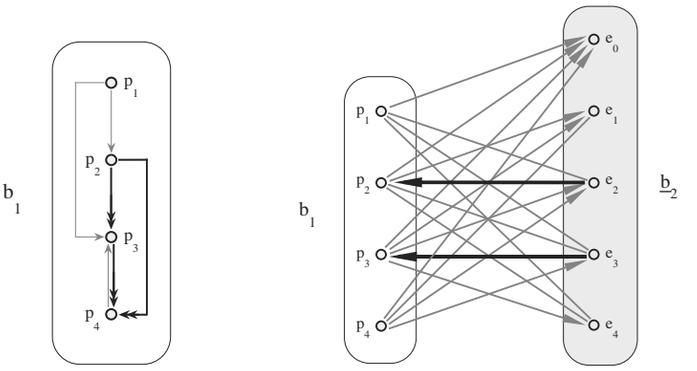
Theorem 1 (Induction of State/Transition Graphs). Let $IS = (P,\ \{b_1, \underline{b}_2\},\ \{\underline{b}_2\},\ K, E)$ be an I–System with $b_1 = \{p_1, p_2, \ldots, p_m\}$, $\underline{b}_2 = \{e_0, e_1, e_2, \ldots, e_m\}$, $m \in I\!N$.
Let $G(b_1) = (b_1, \rightarrowtail)$ be a directed loop-free graph structure over b_1.
Assume that the following structural preconditions hold for $j = 1, 2, \ldots, m$ [1]:
$K(e_j) \setminus \underline{b}_2 = \{p \in b_1 \setminus \{p_j\} \mid (p_j, p) \notin \rightarrowtail\}$, $K(e_0) \setminus \underline{b}_2 = \emptyset$,
$E^{-1}(e_j) = \{p \in b_1 \mid (p_j, p) \in \rightarrowtail\}$, $E^{-1}(e_0) = b_1$,
$E(e_j) = E(e_0) = \emptyset$.
Then the following holds for every $z_0 z_1 z_2 \ldots z_i \ldots \in \mathcal{V}[\![IS]\!]$, $i \in I\!N$, $p, q \in b_1$:
possible phase transitions
$((p,q) \in \rightarrowtail) \Leftrightarrow ((z_i(p) \neq \emptyset) \Rightarrow (\exists z'_1, z'_2, \ldots, z'_k \in GState(IS),\ k \in I\!N :$
$z_0 z_1 z_2 \ldots z_i z'_1 z'_2 \ldots z'_k \ldots \in \mathcal{V}[\![IS]\!]$ with $z'_1(p) \neq 0, z'_2(p) \neq 0, \ldots, z'_{k-1}(p) \neq 0$
and $z'_k(q) \neq 0))$ □

The number of phases in $\underline{b}_2$ (equalling $b_1 + 1$) does not depend on the complexity of the process structure. In fact, through $\underline{b}_2$ there is a one-to-one correspondence between the graph structure $G(b_1)$ and a specific standard interaction between b_1 and $\underline{b}_2$. The proof of the theorem checks the action rules and utilizes semantic properties of $\mathcal{V}[\![IS]\!]$. (The properties can be found in [15].)

In organizational schemes as defined in Definition 5, we distinguish between decision and procedural situations. In the corresponding graphs (e.g. $G'(b_1)$ in Fig. 3.a) we depict these two types of situations by two different sets of edges. Arrows with a single head are representing transitions from decision situations, and arrows with a double head start from procedural situations.
Following this example we extend the idea behind Theorem 1 such that any organizational scheme for b can be generated, or imposed, by an additional inert part connected solely to b.

Corollary 1 (Induction of Organizational Schemes). Let IS' be an I–System with parts and phases like IS from Theorem 1.
Let $G'(b_1) = (b_1, \rightarrowtail_1, \rightarrowtail_2)$ be a directed loop-free graph structure over b_1 with $\{p \in b_1 \mid \rightarrowtail_1(p) \neq \emptyset \wedge \rightarrowtail_2(p) \neq \emptyset\} = \emptyset$ [2]. Set $\rightarrowtail := \rightarrowtail_1 \cup \rightarrowtail_2$.
Assume that the structural preconditions from Theorem 1 hold, *but* with:

[1] For a binary relation $R \subseteq P \times P$ we define $R(p) := \{p' \mid (p, p') \in R\}$ and $R^{-1}(p) := \{p' \mid (p', p) \in R\}$.
[2] p represents either a decision situation or a procedural situation.



(a) $G'(b_1)$ (b) IS'

Fig. 3. Decision and procedural situations. A double arrow head depicts an enforced, a single arrow head depicts a free phase transition.

$$E(e_j) = \begin{cases} \{p_j\} \text{ iff } \rightarrowtail_2(p_j) \neq \emptyset \\ \emptyset \text{ else} \end{cases}.$$

Then the following holds for every $z_0 z_1 z_2 \ldots z_i \ldots \in \mathcal{V}[\![IS']\!]$, $i \in I\!N$, $p, q \in b_1$:

a) *possible phase transitions*
Analogous to Theorem 1 with IS' instead of IS.

b) *free phase transition*
$((p,q) \in \rightarrowtail_1) \Leftrightarrow ((z_i(p) \neq 0 \wedge z_{i+1}(q) \neq 0) \Rightarrow (z_i(p) = q \wedge z_{i+1}(q) = 1))$

c) *enforced phase transition*
$((p,q) \in \rightarrowtail_2) \Leftrightarrow ((z_i(p) \neq 0 \wedge z_{i+1}(q) \neq 0) \Rightarrow (z_i(p) = \mathsf{F} \wedge z_{i+1}(q) = 1))$ □

The idea behind this variation of the construction in Theorem 1 is that if b_1 is in a phase p_i representing a procedural situation then we need an additional influence that forces b_1 to leave p_i. We realize such an influence by simply adding an excitement edge $(e_i, p_i) \in E$. Now, if b_1 is in p_i then $\underline{b}_2$ eventually enters e_i which, in turn, will excite p_i. p_i becomes unstable and eventually enters a successor phase.
In Fig. 3.a p_2 and p_3 represent procedural situations. In order to model the behavior we extend the interaction IS from Fig. 2.b by two excitement edges (e_2, p_2) and (e_3, p_3). We get IS' in Fig. 3.b.

Remark 2. Since the inert parts are connected solely to those part where they induce the organizational scheme the specification of further external influences between parts is not changed (superposition of restrictive influences). □

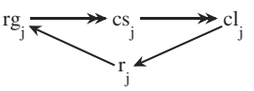
As a formal **convention** *for I–Systems, if no internal behavior is specified in b it is assumed that transitions between any 2 phases are possible.*

4 Application: Problems about Ensuring Priorities among Distributed Processes

Let us assume that in a distributed computer system we have two components, b_1 and b_2, which at times want to download time critical jobs on a special high-speed machine M. In order to avoid conflicts about accessing M between the involved components priority regulations are frequently implemented. Due to organizational reasons b_2 is assumed to have higher priority than b_1.
Throughout this section we will discuss various aspects of priority handling in the context of the example addressed. Formally we model b_1 and b_2 by one part of an I–System $\widetilde{IS}$ each. We will focus on the basic ideas in modeling mutual influences, omitting proofs (because of page limitations) and the detailed semantical specification of $\mathcal{V}[\![\widetilde{IS}]\!]$.

4.1 Local Behavior of b_1 and b_2

The local behavior of b_j, $j = 1, 2$, can be depicted by the organizational scheme



The procedure of accessing M is described by a 3-phase process structure: The registration phase rg_j is the only predecessor of the critical section phase cs_j in which b_j is allowed to access M. The clearing phase cl_j is the only successor of the critical section phase. The transitions $rg_j \rightarrow cs_j$ and $cs_j \rightarrow cl_j$ are enforced and will occur if no global influences impose further blocking restrictions. The other activities in b_j will be summarized into a remainder phase r_j, from which, and to which, phase transitions are assumed to be free. They depend on local control decisions e.g. resulting from interactions with users.

The local behavior in part b_j can be generated by an I–System according to Corollary 1.

4.2 Static Access Priority

If the access to a shared resource is to be arranged on the basis of mutual exclusion then the resulting priority requirement is that if two processes are both ready to access the resource then the one of higher priority is to go ahead while the other one has to wait. In the context of the example in 4.1 this condition can be phrased as follows.

Requirements 1.

PR0. If b_j is in rg_j, $j = 1, 2$, then only b_2 may enter cs_2 while b_1 has to wait.

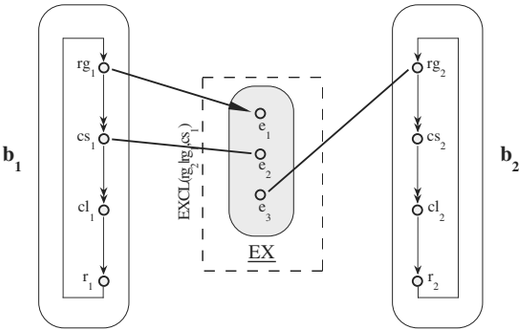


Fig. 4. Minimal realization of PR0

In order to realize the restriction PR0 we use the formal construction $EXCL(rg_2 \mid rg_1, cs_1)$. Here an inert part $\underline{EX}$ interacts with b_1 and b_2 as depicted in Fig. 4. For every global activity state of $\widetilde{IS}$, the interaction satisfies the following requirements:

E1. If b_2 is in rg_2 then no transition $rg_1 \rightarrow cs_1$ is possible in b_1.
E2. b_2 cannot be prevented from entering or leaving rg_2. b_1 has no influence on b_2 as to leaving rg_2.
E3. Once b_2 has left rg_2, b_1 cannot be prevented from performing the phase transition $rg_1 \rightarrow cs_1$ unless b_2 reenters rg_2.
E4. b_2 has no influence on b_1 as to leaving rg_1 or cs_1.

The interaction is *minimal* in the sense that through this connection no further restriction is imposed on b_1 or b_2. PR0 is an immediate consequence of E1, E2, and the local behavior of b_1. (Please remember from the end of section 3 that transitions between any 2 phases in $\underline{EX}$ are possible yet would occur only through external influence.) The proof of E1-E4 would be done by analyzing $\mathcal{V}[\![\widetilde{IS}]\!]$.

Remark 3. $EXCL(rg_2 \mid rg_1, cs_1)$ is an example for a *standard construction*. Through standard constructions we are able to model a large set of complex interaction and synchronization relationships, including the behavioral patterns unique for Interaction Systems. The correctness proofs are done by analyzing the trace semantics. Due to page limitations we refer to [15] for more details. □

4.3 Enforcing Access Priority

In the scenario described at the beginning of section 4 at most one of the components b_1, b_2 may have a job residing and executing on M (sensitive data). Also, if b_1 is executing a job on M and a job is waiting for execution at b_2, then because of b_2's higher priority some influence should be exerted on b_1

(e.g. through a kind of notification) to finish up and give room to b_2. (b_1, due to its lower priority, should not impose a waiting influence on b_2, i.e., *priority inversion* should be excluded.) In terms of our formal representation, these conditions can be rephrased as follows:

Requirements 2.

PR1. cs_1 and cs_2 are mutually exclusive phases.
PR2. If b_2 is in rg_2, it influences b_1 to leave cs_1.
PR3. If b_1 is in rg_1, it does not exert any influence on b_2 as to leaving cs_2.

We realize this form of priority through a construction with an inert part $\underline{PR}$ which is shown in Fig. 5.

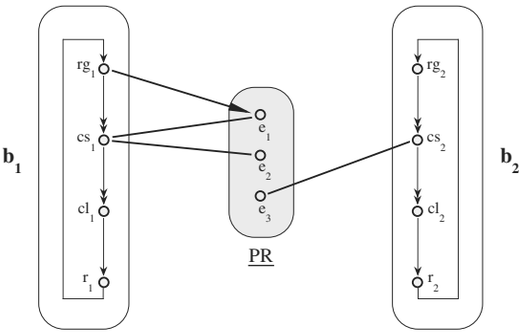


Fig. 5. Minimal realization of PR1, PR2, PR3

This construction is a variation of the standard construction in Fig. 4, with edge (cs_2, e_3) instead of (rg_2, e_3), and by adding a coupling edge (e_1, cs_1). This edge represents an additional interaction/influence that is necessary because property E4 satisfied by the previous standard construction contradicts requirement PR2 for enforcing access priority. PR1 is satisfied as well by this simple extension. E2 is still valid (with cs_2 instead of rg_2) and leads to PR3.

4.4 Reserved Access Right

In the context of our application example let us assume that b_1 has a series of time critical jobs to download onto M. They carry short deadlines. Each of them is supposed to be rather small in comparison to the jobs without deadlines coming from b_2. M's scheduler may therefore, in order to meet the deadlines, decide to start executing a job of b_1 instead of processing a job of b_2, despite of b_2's high priority. Obviously this problematic configuration (another example of *priority inversion*) may be repeated indefinitely often resulting in starvation of b_2. In order to avoid this undesirable effect we formulate the following conditions.

Requirements 3.

PR4. Through the interaction, b_j, $j = 1, 2$, cannot be prevented from entering rg_j anytime.

PR5. If b_2 is in rg_2 while b_1 is in cs_1 then b_1 can proceed only as far as rg_1 as long as b_2 has not entered cs_2.

PR6. After b_2 has left cs_2, b_1 has the chance to reenter cs_1.

The main idea is to establish the needed communication between b_1 and b_2 through a single inert part $\underline{CP}$ the behavior of which is to be restricted by another inert part $\underline{EX'}$. The formal construction including the necessary relations is to be found in Fig. 6. The second inert part is part of a parametrization of the standard construction introduced in section 4.2. Its known properties E1, E2, E3, E4 in conjunction with the interaction through $\underline{CP}$ are the basis for the proof of PR4, PR5, PR6.

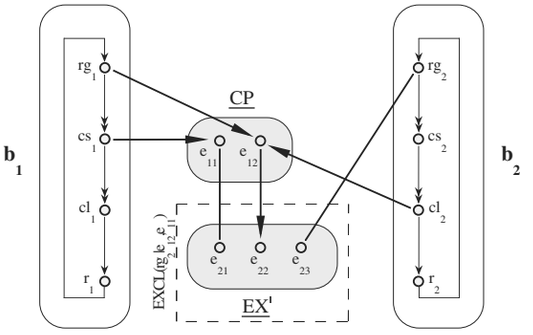


Fig. 6. Minimal realization of PR4, PR5, PR6

4.5 Guaranteed Access under Priority

If we assume, in the course of our developing example, that the critical sections in b_1 and b_2 underlie the restrictions PR1, PR2, PR3, PR4, PR5, PR6 then we would simply combine the constructions in Fig. 5 and Fig. 6 in order to overcome the shortcomings of the solution that is solely taking care of the priority restrictions. We can combine the effects of the two constructions since *they have no side-effects.* Graphically this is depicted in Fig. 7.

While the discussion in the context of our example scenario could easily continue by further elaborating on open priority problems our purpose here was simply to demonstrate, in the dimension of a large real system design, the *modularity* and *incrementality* of our formal tools. Recent research related to compositionality/ modularity within the design and analysis of distributed systems can be found in [1,6,8].

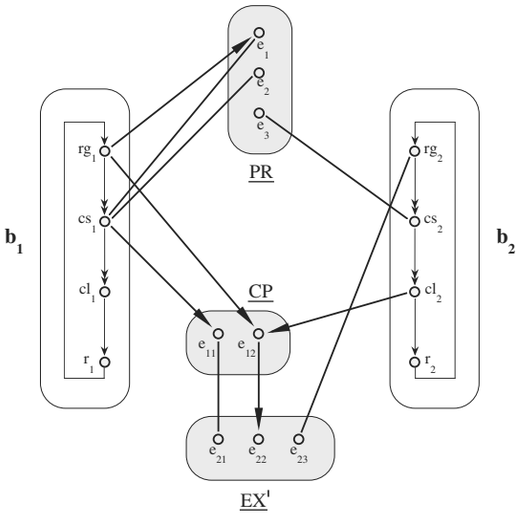


Fig. 7. Minimal realization of PR1, PR2, PR3, PR4, PR5, PR6

5 Conclusion

In order to model events in distributed components as well as cooperation between components in a formally unified framework we defined I–Systems. System components are presented as *parts*. These are constituted through their relevant (local) states, denoted by *phases*. The interaction between parts is specified through two binary relations, denoted as *coupling* and *excitement* relations, respectively. Their (restrictive) influences on the involved parts is defined through action rules for which novel trace semantics have been constructed thus formally introducing the *global effect* of the local cooperation, in terms of propagating local influences.

Modeling distributed behavior in large systems is done in two different ways. Traditional approaches start with modeling behavior of components and next interaction between components. The problem is to understand/ analyze the effect of the interaction on the component behavior. In our approach we model interaction between components starting with elementary interaction relations and by explicitly defining their influence on the behavior of the involved components. We have proven to cover the range of representations given by the first approach but our tools are even more powerful. They allow for explicitly distinguishing between *free* and *enforced* actions (transitions) which is crucial for adequately modeling *organizational processes*. We have shown the additional advantage through I–Systems to *incrementally* model complex distributed (software) systems and to guarantee their correctness at the same time.

We have done much research on finding a tool set of *standard constructions* for the efficient formal representation of all forms of synchronization and cooperation between distributed components. Currently we are finalizing our efforts

to generate these formal constructions through a very small set of elementary restrictions, and through particular principles for their refinement and modification.

References

1. Rajeev Alur, Luca de Alfaro, Thomas A. Henzinger, and Freddy Y. C. Mang. Automating Modular Verification. In *CONCUR'99*, volume 1664 of *LNCS*, pages 82–97. Springer-Verlag, 1999.
2. Rajeev Alur and Thomas A. Henzinger. Reactive Modules. In *Formal Methods in System Design*, volume 15, pages 7–48. Kluwer Academic Publishers, 1999.
3. Ralph Back, Anna Mikhajlova, and Joakim von Wright. Reasoning About Interactive Systems. In *FM'99, Vol. II*, volume 1709 of *LNCS*, pages 1460–1476. Springer-Verlag, 1999.
4. Piero Bonatti, Sabrina de Capitani di, and Pierangela Samarati. A modular approach to composing access control policies. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, pages 164 – 173, Athens Greece, nov 2000.
5. Paul Caspi, Alain Girault, and Daniel Pilaud. Automatic Distribution of Reactive Systems for Asynchronous Networks of Processors. *IEEE Transactions on Software Engineering*, 25(3):416–427, 1999.
6. Michael Charpentier and K. Mani Chany. Towards a Compositional Approach to the Design and Verification of Distributed Systems. In *FM'99, Vol. I*, volume 1708 of *LNCS*, pages 570–589. Springer-Verlag, 1999.
7. V. Diekert and G. Rozenberg. *The Book of Traces*. World Scientific, 1995.
8. Valérie Issarny, Luc Bellissard, Michel Riveill, and Apostolos Zarras. Component-Based Programming of Distributed Applications. In *Distributed Systems*, volume 1752 of *LNCS*, pages 327–353. Springer-Verlag, 2000.
9. Man Lin, Jacek Malec, and Simin Nadjm-Tehrani. On Semantics and Correctness of Reactive Rule-Based Programs. In *PSI'99*, volume 1755 of *LNCS*, pages 235–246. Springer-Verlag, 2000.
10. Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
11. Andrea Maggiolo-Schettini, Horst F. Wedde, and Józef Winkowski. Modeling a Solution for a Control Problem in Distributed Systems by Restrictions. *Theoretical Computer Science*, 13:61–83, 1981.
12. W. Reisig. *Elements of Distributed Algorithms*. Springer-Verlag, 1998.
13. Horst F. Wedde. An Iterative and Starvation-free Solution for a General Class of Distributed Control Problems Based on Interaction Primitives. *Theoretical Computer Science*, 24, 1983.
14. Horst F. Wedde and Mario Lischka. Modular authorization. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*, Chantilly, Virginia, May 3-4 2001.
15. Horst F. Wedde and Arnim Wedig. Explicit Modeling of Influences, and of Their Absence, in Distributed Systems (Extended version). Technical report, University of Dortmund, October 2001.
    http://ls3-www.cs.uni-dortmund.de/IS/P/techrep1001.ps.
16. Peter Wegner. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5):81–91, May 1997.
17. Peter Wegner and Dina Goldin. Interaction as a Framework for Modeling. In *LNCS*, volume 1565. Springer-Verlag, 1999.