A Composition Operator for Systems with Active and Passive Actions

Stefan Strubbe* and Rom Langerak

Twente University, PO BOX 217, 7500AE Enschede, The Netherlands
s.n.strubbe@math.utwente.nl, langerak@cs.utwente.nl

Abstract. We investigate requirements for a composition operator for complex control systems. The operator should be suitable for a context where we have both supervisory control and a system that consists of multiple (two or more) components. We conclude that using both passive (observing) and active (controlling) transitions is advantageous for the specification of supervisory control systems. We introduce a composition operator that meets the requirements. We give both operational and trace semantics for this operator and give necessary and sufficient conditions for commutativity and associativity.

Keywords: Compositional modelling, supervisory control.

1 Introduction

A complex system typically consists of multiple components which are running simultaneously and which are interacting with each other. Modelling these complex systems in a compositional way can be done by using a composition operator which combines the different components of the system. If we denote the composition operator by $||$, then $A||B$ is the system that is composed out of subsystems A and B. The interaction between the two subsystems is regulated by the composition rules of the operator $||$. Because the systems A and B run in parallel (or concurrently), we call $||$ the parallel composition operator.

The goal of this paper is to make clear that for the modelling of many types of systems, it is advantageous and natural to use two types of interaction: blocking-interaction and non-blocking-interaction. We show that this idea can be formalized by distinguishing active and passive actions. The operator we define via structured operational semantics ([1]) can be used with any transition-based model like automata or process algebra. The focus of this paper is to give a careful motivation for this operator.

In Section 2, which is the main part of the paper, we introduce the active/passive framework and we develop a composition operator that can establish several types of interaction between systems (that are built out of active and passive transitions). While we develop the operator step by step, we motivate each step by means of simple and clear examples. Some of these examples

* Supported by the EU-project HYBRIDGE (IST-2001-32460).

F. Wang (Ed.): FORTE 2005, LNCS 3731, pp. 24–37, 2005.

are about supervisory control systems because we think that supervisory control systems provide natural examples for our modelling framework. After the framework has been introduced (including a structural operational semantics for the composition operator), we give a more extensive example in Section 2.4 which shows all features of the framework and the composition operator. In Section 2.5 we take a closer look at supervisory control systems. We explain why we think that our framework has certain advantages over other frameworks in modelling supervisory control systems. Section 2 ends with a technical result on the operator.

In Section 3 we give a denotational semantics in terms of an extension of traces. We think that this semantics gives more insight in the composition operation and consequently more justification for the use of this specific operator.

In the last section we draw some conclusions and point out directions for future research.

2 The Active/Passive Framework

We can distinguish two types of interactions between processes. First, *blocking*-interaction: both partners (for example the controller and the process) need to be able to do the action, otherwise the action will not take place. This is the type of interaction that we see in many process algebra models (e.g. [2,3]). Secondly, *non-blocking*-interaction: one of the partners (the passive one) is not able to block the other (the active one). For example if a person (the passive partner) observes that a light (the active partner) is switched on. The person observes, but is not able to block the switching, i.e. the light could also be switched on without the person observing it.

Non-blocking-interaction can already be found in the literature, e.g. in broadcast systems ([4]) where several listening processes can receive (but not block) a signal, or in I/O automata ([5,6]), where processes should be input-enabled, i.e. ready to receive an input in any state of the process, such that an output will never be blocked.

We see that most formalisms support only blocking interaction and that some formalisms (broadcasting systems, I/O automata) support only non-blocking interaction. However, there exists no formalism that supports both blocking and non-blocking interaction. We will motivate that for our purposes, it is desirable to have a formalism that supports both types of interaction.

The context of processes that we want to specify is the following:

1. We want to specify *supervisory control systems*. We will see that we need both blocking and non-blocking interaction for this.
2. We are also interested in *modular supervisory control*, where a controller may consist of several modules. We will motivate that for this, in addition to blocking and non-blocking interaction, we need to control the scope of non-blocking interaction.
3. We want to specify *complex processes that consist of interacting subprocesses*. Here we will argue that we need the possibility to have multi-way synchronizations.

Now we will introduce our framework, which is based on active and passive actions and which supports both blocking and non-blocking interaction. We introduce the framework in three steps, corresponding to the three points in the above context. In each step we give motivating examples. In the first step we explain how active and passive actions can be used to establish blocking and non-blocking interaction. In the second step we explain how to deal with observations in systems that consist of more than two components. In the third step we treat the issue of multi-way synchronization (can one active action synchronize with multiple passive actions or only with one passive action?).

2.1 Step 1: Establishing Blocking and Non-blocking Interaction

We consider two types of actions: *active* actions (denoted as a, b etc.) and *passive* actions (which are observing actions and are denoted as $\bar{a}$, $\bar{b}$, etc.). Because we want to have both blocking and non-blocking interactions, we have to make clear which interactions are blocking and which are non-blocking. Therefore, we introduce the set A which contains all actions that are involved in blocking interaction. The composition operator will now be denoted by $|A|$. We still use $||$ to denote the composition operator in cases where A is unspecified or irrelevant. Blocking-interaction is now expressed by the following operational rule ([1]):

$$r1.\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{a} L_2'}{L_1|A|L_2 \xrightarrow{a} L_1'|A|L_2'}(a \in A),$$

which says that a blocking-synchronization (or active-active synchronization) on a from joint location $L_1|A|L_2$ to $L_1'|A|L_2'$, can only happen when both partners have the action a available from locations L_1 and L_2 to locations L_1' and L_2' respectively (In order to comply with the terminology used in timed and hybrid automata, we use the term *locations* to designate the *states* in an automaton). In Figure 1, where $a \in A$, we see that both the process P and the controller C have transitions labelled with a from their initial locations P_1 and C_1. This results in the (synchronized) a-transition in the composite system $P||C$. In location P_3, P can do an a-action, but because $a \in A$ and C does not have an

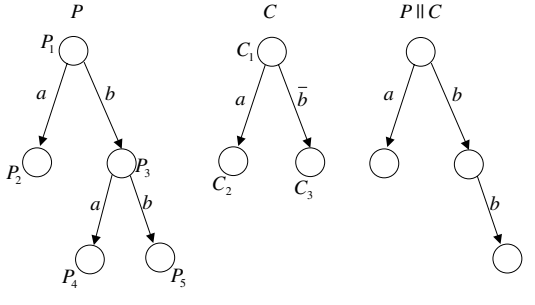

Fig. 1. Process P controlled by C

active a-transition in location C_3, this transition is blocked by C and therefore the transition is not present in $P||C$.

Blocking as expressed in rule r1 can be used in supervisory control where a controller C blocks certain actions of the process P. Another situation where active actions must synchronize (i.e. are in A) is where two partners cooperate on a certain task. If two persons are chatting with each other, then they are cooperating on the chat-task so to say. In other words, both are chatting or both are not chatting, one cannot chat without the other (this situation will be described in the example in Section 2.4). For a situation where two persons P_1 and P_2 can both do a specific action (like hanging up the phone as described in Section 2.4) independently of the other person, then the action can be independently performed (i.e. should not be in A).

For non-blocking-interaction, we need an active a with $a \notin A$ in one partner and a passive $\bar{a}$ in the other partner. Non-blocking interaction is now expressed by

$$r2.\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|A|L_2 \xrightarrow{a} L_1'|A|L_2'}(a \notin A),$$

which means that L_1 executes an a, which is observed by a $\bar{a}$-transition outgoing from L_2. In Figure 1 we see for example that the b-transition from location P_1 in P, is observed by the $\bar{b}$-transition in C. We also need rule r2', which is the mirror rule of r2 (i.e. $L_1 \xrightarrow{\bar{a}} L_1', L_2 \xrightarrow{a} L_2'$ instead of $L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{\bar{a}} L_2'$ etc.).

If L_2 can not observe a-actions (i.e. there is no outgoing $\bar{a}$-transition), L_1 should still be able to execute a, since this execution does not depend on whether or not some other process is observing the action. This is expressed by

$$r3.\frac{L_1 \xrightarrow{a} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|A|L_2 \xrightarrow{a} L_1'|A|L_2}(a \notin A)$$

and its mirror rule

$$r3'.\frac{L_1 \not\xrightarrow{\bar{a}}, L_2 \xrightarrow{a} L_2'}{L_1|A|L_2 \xrightarrow{a} L_1|A|L_2'}(a \notin A).$$

In Figure 1 we see for example that P has a b-transition from location P_3, which cannot be observed by C, but which is still present in $P||C$.

Note that because rules r2, r3 and r3' are the only rules from which active transitions can be derived, it follows that if L_2 *can* observe a, then it also *will* observe a, i.e. if L_2 has a $\bar{a}$-transition, then it can not choose not to observe an a executed by the other component. This makes sense in many situations, e.g. when some system broadcasts a radio signal a and a receiver is able to receive the signal (i.e. it has a passive $\bar{a}$-transition), then we should not allow the possibility that the signal is broadcast while the receiver does not receive (i.e. does not synchronize its passive $\bar{a}$-action with the active a-action).

Negative premises in rules (as in rule r3) may in general lead to complications (see [7]), however in our rules there are no problems, as can easily be seen from

the fact that $\xrightarrow{\bar{a}}$ transitions never depend on $\xrightarrow{a}$ transitions, and therefore no circularities may arise.

In the supervisory control context, observing as active/passive-synchronization means that the controller observes actions of the process. Outside the supervisory control context, this mechanism can be used for other kinds of observation (e.g. a person that observes an alarm signal as described in Section 2.4).

2.2 Step 2: Controlling the Scope of the Observations

Passive transitions are intended to observe active transitions. This means that outside an open-systems context (i.e. when the system is closed and will not interact with other systems), passive transitions are supposed not to be present, since there is nothing to observe. Suppose we have a system that consists of a process P and a controller C, where the controller observes the actions of P. One could argue that $|A|$ should be defined such that after composition, there are no passive transitions anymore (which is the case if we only consider rules r1,r2 and r3). But that would not be suitable for a broader composition context like modular supervisory control: Suppose we have a system with one process P and two controllers C_1 and C_2, where both C_1 and C_2 are concurrently observing P. If we define $C := C_1||C_2$ as the composed controller, then C should still observe P, with other words, C should still contain the passive transitions from C_1 and C_2 to observe P. This means that the 'real' observations happen when we compose C with P and not when we compose C_1 with C_2.

One solution one could think of is to indicate within the composition operator where the 'real' observations take place. We could use an observation set O and then $||_O$ (or together with A, $|^O_A|$) means that observations take place in this composition for the events in O. Then in the compound $(C_1||C_2)||_O P$ there are no passive transitions with labels from O anymore.

For the double-controller situation, $||_O$ seems to be a good solution. However, it seems that for modelling the following situation, we need a different solution: Suppose three persons P_1, P_2 and P_3 are working on a problem. All persons start working independently on this problem. Once one of the persons found the solution, all three persons stop with the problem. This can be modelled as in Figure 2, where the signal *ready* is 'broadcast' by one of the persons as soon as this person solved the problem, and is then received by the others. In this situation, every P_i should be able to hear every P_j $(i \neq j)$. With $||_O$ we can not express this. (In $(P_1||P_2)||_O P_3$, P_1 does not observe P_2. In $(P_1||_O P_2)||P_3$, P_3 does not observe P_1 and P_2 etc.).

Instead of using $||_O$, we use the closing operator $[\cdot]_C$. $[X]_C$ discards all passive transitions in X with labels from C. With this closing operator, we can solve both the double-controller and the 'three persons' problem. The composition rules for this operator are

$$r7.\frac{L \xrightarrow{a} L'}{[L]_C \xrightarrow{a} [L']_C}$$

$$r8.\frac{L \xrightarrow{\bar{a}} L'}{[L]_C \xrightarrow{\bar{a}} [L']_C}(\bar{a} \notin C).$$

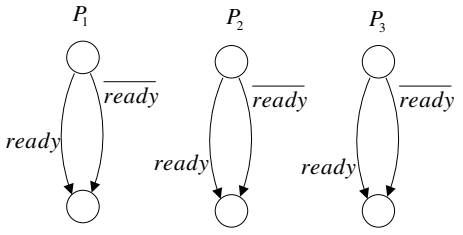

Fig. 2. Situation where all persons can observe all others

With $[\cdot]_C$ we can control the scope of the observations. Now, $||$ should be defined such that in $X||Y$, X can observe Y (and vice versa), but also $X||Y$ can still observe Z in $(X||Y)||Z$. This is expressed by

$$r4.\frac{L_1 \xrightarrow{\bar{a}} L_1'}{L_1|A|L_2 \xrightarrow{\bar{a}} L_1'|A|L_2}$$

and its mirror r4' which say that after the composition, every passive transition 'remains', such that the component to which this transition belongs, is still able to observe a new component which might be added to the composite system in a new composition operation. If, for example, we know that X and Y only observe each other and will (or can) not observe Z or other components, then we could specify this as $[X||Y]||Z$. ($[\cdot]$ is shorthand for $[\cdot]_{\bar{\Sigma}}$ with $\bar{\Sigma}$ the set of all passive actions).

2.3 Step 3: Multi-way Synchronization

Consider the modular supervisory control situation again where we have two controllers C_1 and C_2 and a process C_3. Now suppose that C_3 has an action a which can be observed by both C_1 and C_2 (i.e. both controllers have $\bar{a}$-transitions). If we allow that both controllers can observe a concurrently (which is most natural), then we need to express the possibility of a multi-way synchronization (in this case: two passive actions and one active action). Another example (outside the supervisory control context) where we need a multi-way synchronization is the situation where an alarm signal in an office is heard (observed) by two different employees working in that office (this example is also described in Section 2.4).

The question now is whether there are situations we want to model that do not allow multi-way synchronizations. One such example (also considered in Section 2.4) is the telephone situation: a telephone in an office rings and only one of the employees may answer the call. Although both employees hear the telephone, only one may answer it (i.e. may synchronize its passive action with the active telephone signal).

We see that it is desirable to distinguish two types of passive actions: passive actions for which multi-way synchronization is allowed and passive actions for which this is not allowed. Therefore we introduce the set P, which contains all

passive actions for which multi-way synchronization is allowed. The composition operator will now be denoted by $|^P_A|$. Multi-way synchronization is expressed by

$$r5.\frac{L_1 \xrightarrow{\bar{a}} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1'|^P_A|L_2'}(\bar{a} \in P),$$

which means that two passive $\bar{a}$-transitions, one in C_1 and one in C_2, synchronize, which results in a $\bar{a}$-transition for the composite system $C_1||C_2$. This synchronized passive transition can observe an a in C_3, which results in a new (multi-way) synchronized transition in $(C_1||C_2)||C_3$ which then expresses that C_1 and C_2 concurrently observe C_3.

If $a \in P$ and C_1 can observe a, but C_2 cannot observe a, then in $(C_1||C_2)||C_3$, C_1 should synchronize its passive $\bar{a}$ with the active a of C_3, while C_2 idles. This situation is expressed by

$$r6.\frac{L_1 \xrightarrow{\bar{a}} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1'|^P_A|L_2}(\bar{a} \in P)$$

and its mirror rule r6'. In the new situation (where we have introduced the set P), we can see that rule r4 (which expresses that passive transitions should interleave) only applies for passive actions not in P. Therefore r4 should be changed to

$$r4.\frac{L_1 \xrightarrow{\bar{a}} L_1'}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1'|^P_A|L_2}(\bar{a} \notin P).$$

With the set P as part of the operator, rules r1,r2 and r3 should be changed to

$$r1.\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{a} L_2'}{L_1|^P_A|L_2 \xrightarrow{a} L_1'|^P_A|L_2'}(a \in A),$$

$$r2.\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|^P_A|L_2 \xrightarrow{a} L_1'|^P_A|L_2'}(a \notin A),$$

$$r3.\frac{L_1 \xrightarrow{a} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|^P_A|L_2 \xrightarrow{a} L_1'|^P_A|L_2}(a \notin A).$$

Now rules r1 till r6 (and the mirror rules r2',r3',r4' and r6') form the structural operational semantics for $|^P_A|$ and rule r7 and r8 form the structural operational semantics for $[\cdot]_C$.

Note that the synchronization-mechanisms for active transitions and passive transitions are different. If an active action a is synchronizing (i.e. $a \in A$), then a component can execute the action only when the other component in the composition can also execute the action (and vice versa). If a passive action $\bar{a}$ is synchronizing (i.e. $\bar{a} \in P$), then if both components can execute the action, they have to synchronize. However, if only one component can execute the action, the action can still be executed without the other component synchronizing with it.

This is expressed in rule r6. For active-active synchronization we do not have an equivalent rule as r6. Because of this difference between the synchronization-mechanisms, we need to use different composition rules for both the active and the passive actions (i.e. we can not combine active and passive synchronization in the same rules).

2.4 Example

In Figure 3 we see an example where we find back all interaction structures we described before: non-synchronizing active transitions, synchronizing active transitions, non-synchronizing passive transitions, synchronizing passive transitions and active-passive synchronization.

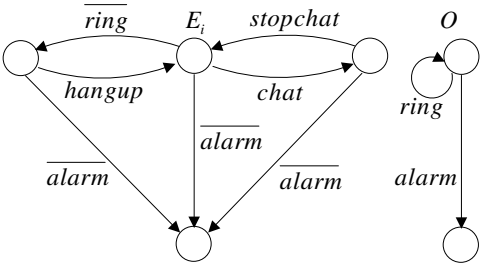

Fig. 3. Example with different kinds of interaction

The example of Figure 3 concerns an office O with two employees E_1 and E_2. In the office there are two sources which can produce a signal: A telephone and an alarm. The telephone rings when somebody calls the office, the alarm fires when there is danger, which means that the employees should leave the office when they hear the alarm. Both the telephone and the alarm execute their signals independently from the employees. Therefore they are modelled as active transitions labelled *ring* and *alarm* respectively (see Figure 3). If the telephone rings, O makes a self-loop which means that O stays in the same location. This location has the meaning of 'normal-working-conditions'. If the alarm goes, O jumps to a second location which has as meaning 'dangerous-working-conditions'.

E_1 and E_2 have the same automaton-structure. From E_1 (or E_2), the employee can exhibit three different actions: He can chat with his fellow employee, he can pick up the phone and he can leave the office. Leaving the office only happens when the alarm goes off. This action is modelled as a passive transition which synchronizes with the independent active alarm-transition in O. We see that from every location, the employee can react on the alarm. When the phone rings, the employee can pick up the phone by synchronizing its passive *ring* event with the active *ring* from O. The employee hangs up with an active *hangup* event. The employee can start a chat with his office-mate via the active *chat* event. This should synchronize with an active *chat* event of the office-mate

(both should be willing or able to chat). The chat will be ended with an active-active synchronization of *stopchat*.

From the model we see that if the employees are chatting, they first have to end the chat before one of them can pick up the phone. This means that they will probably miss the first *ring* signal, but can maybe interact on the second *ring* signal (the phone might give multiple *ring* signals when somebody calls). Also, if one employee is phoning, he first has to hang up before a chat can be started.

Where do we see the different kinds of interaction? The passive *ring* transitions of E_1 and E_2 should be non-synchronizing since only one passive transition is allowed to synchronize with the active *ring* transition in O. The passive *alarm* transitions in E_1 and E_2 should synchronize because both employees react synchronously on the alarm. The active *hangup* transitions in E_1 and E_2 should be non-synchronizing (only one employee hangs up). The active *chat* and *stopchat* transitions in E_1 and E_2 should synchronize (both employees chat).

From the above follows that for the composition $E_1|^P_A|E_2$ we get $A = \{chat, stopchat\}$ and $P = \{\overline{alarm}\}$. For the composition $(E_1|^{\overline{alarm}}_{chat,stopchat}|E_2)|^P_A|O$ we get $A = \emptyset$ and $P = \bar{\Sigma}$. The total specification is then

$$(E_1|^{\overline{alarm}}_{chat,stopchat}|E_2)|^{\bar{\Sigma}}_{\emptyset}|O. \tag{1}$$

If (1) is the system that we want to analyze, then we could close down all observation channels (i.e. the passive transitions) with the $[\cdot]$ operator. Now suppose that (1) is only one chamber of a bigger office consisting of multiple chambers. Then this chamber is only one unit and lets call it U_1. The other units then are $U_2 = (E_1'|^{\overline{alarm}}_{chat,stopchat}|E_2')|^{\bar{\Sigma}}_{\emptyset}|O'$, $U_3 = (E_1''|^{\overline{alarm}}_{chat,stopchat}|E_2'')|^{\bar{\Sigma}}_{\emptyset}|O''$ etc., where $E_i = E_i' = E_i''$, $O = O' = O''$ etc. The telephones in each office are local. With other words, if a telephone rings in one office, only the employees in that office hear the phone and can answer it. The alarm however is global. If there is danger in the building, the alarm goes synchronously in all offices. To specify that the phones are local, we use $[\cdot]$ and get $[U_i]_{ring}$. To specify that the alarm is global, we use *alarm* as a synchronization event in the composition of the units. The total composition of the whole building is then

$$U_1'|_{alarm}|U_2'|_{alarm}|U_3'|_{alarm}|\cdots,$$

where $U_i' = [U_i]_{ring}$.

2.5 Supervisory Control with $|^P_A|$

In this section we want to take a closer look at supervisory control. We will shortly describe the main concepts of supervisory control and thereafter we compare how specification of control systems can be done in our active/passive framework to how it can be done in a framework where there is only blocking-interaction.

In the supervisory control paradigm ([8,9]), the actions of a process can be observable or unobservable and they can be controllable or uncontrollable. Observable actions can be observed by the controller (and unobservable actions can

not). Controllable actions can be controlled by the controller (and uncontrollable actions can not). This means that the controller can block these actions, i.e. can prevent them from happening.

If we specify the control system within an automata framework, then the process and the controller are modelled as two separate automata which can interact. The plant executing an action is then modelled as a transition (labelled with this action) from one process-location to another process-location. The controller observing an action is then modelled as a transition in the controller that synchronizes with the to-be-observed transition in the process.

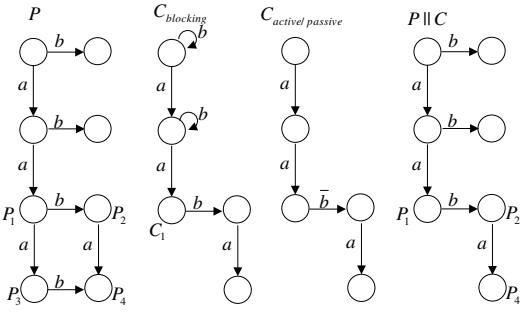

Fig. 4. Control in only-blocking framework and in active/passive framework

Consider the process P in Figure 4 with a controllable/observable and b uncontrollable/observable. We want to control this process such that the behavior of P is restricted to $P||C$ in Figure 4.

First lets see how this can be done in the only-blocking-interaction framework. Since a and b are both observable, we mark them as synchronization actions (i.e. in the composition operation these actions must synchronize while non-synchronization actions will be interleaved). The controller that does the job is $C_{blocking}$ in Figure 4. We see that we need two self-loops on action b, because otherwise the two upper b-transitions of P are blocked and that is not what we want according to $P||C$. We see that, according to the specification $P||C$, transition $P_1 \xrightarrow{a} P_3$ of P is blocked because of the absence of an a-transition in $C_{blocking}$ at location C_1.

For the active/passive framework, $C_{active/passive}$ from Figure 4 does the job. Because a is controllable (i.e. blockable), $a \in A$ and because b is not controllable, $b \notin A$. Blocking the a transition from P_1 to P_3 is done in the same way as in the only-blocking framework. The difference however is, that here we do not need the self-loops on b.

We could say that in the active/passive framework, the controller will observe only (by means of a passive transition) when it needs the information. In Figure 4, $C_{active/passive}$ observes $P_1 \xrightarrow{b} P_2$, but does not observe the upper b-transitions of P, because P may execute them without the controller 'knowing' it. In the only-blocking framework, the controller must also synchronize

on the upper b-transitions, because otherwise they will be blocked. We think that this is an important advantage of using active/passive transitions: for uncontrollable/observable actions, transitions are only needed where observations are needed. If there is only blocking-interaction, transitions in the controller are needed everywhere the process executes an uncontrollable/observable action (otherwise they will be blocked and that is not allowed).

In fact we can say that in the supervisory control context, A (from $P|^P_A|C$) contains the controllable actions of the process P. Then, we resume: uncontrollable actions are observed by passive transitions and controllable actions are observed and controlled by active transitions. Note that an active transition of C labelled with a controllable event of P is both controlling (it allows P to execute the action) and observing (the controller synchronizes this transition with the one from P).

2.6 Commutativity and Associativity of $|^P_A|$

Theorem 1. *$|^P_A|$ is commutative for all A and P. $|^P_A|$ is associative if and only if for all events a we have: $a \notin A \Rightarrow \bar{a} \in P$.*

The proof of this theorem can be found on www.cs.utwente.nl/~strubbesn.

If, according to the above theorem, $|^P_{A_1}|$ and $|^P_{A_2}|$ are associative operators, we have $(X|^P_{A_i}|Y)|^P_{A_i}|Z = X|^P_{A_i}|(Y|^P_{A_i}|Z)$ for $i = 1, 2$, and therefore we could write $X|^P_{A_i}|Y|^P_{A_i}|Z$ instead. Note, however, that in general $(X|^P_{A_1}|Y)|^P_{A_2}|Z \neq X|^P_{A_1}|(Y|^P_{A_2}|Z)$ just as in general we do not have $(X|A_1|Y)|A_2|Z \neq X|A_1|(Y|A_2|Z)$ in for example CSP or LOTOS.

3 Trace Semantics

We want to give a trace semantics for the operator $|^P_A|$. This operator is used in a context where we have active actions (like a) and passive actions (like $\bar{a}$). A trace of an automaton could then be $ab\bar{a}$ for example. However, this notion of trace is not strong enough to give a trace semantics for composition (i.e. the sets of traces of the components are not enough to determine the set of traces of the composition).

See for example Figure 5. There the set of traces of X_1 and X_2 are both equals to $\{\epsilon, a, ab, aba, a\bar{c}\}$. The set of traces of Y equals $\{\epsilon, c\}$. If we could determine the trace set of a composite system from the trace sets of the components, then $X_1||Y$ and $X_2||Y$ should have the same trace set (because X_1 and X_2 have the same trace set). But this is not the case, since $acba$ is a trace of $X_1||Y$ while it is not a trace of $X_2||Y$. Therefore this notion of traces is not strong enough for a trace semantics of $|^P_A|$.

Another option for a trace semantics would be to look at traces of the form σP with $\sigma \in (\Sigma \cup \bar{\Sigma})^*$ and $P \subset \bar{\Sigma}$, where σP means that the process can execute the trace σ such that it ends up in a state where the set of passive events that are enabled equals P. This can be seen as an analogy of refusal traces in [10]. However, this notion of trace is also not strong enough because the processes

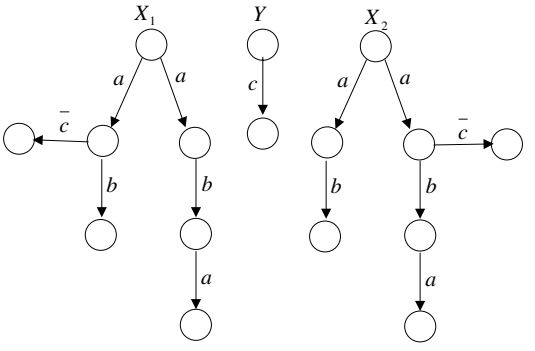

Fig. 5. Processes X_1, X_2 and Y

X_1 and X_2 in Figure 5 have the same traces of the form σP, while $X_1||Y$ and $X_2||Y$ have different traces.

We now introduce a trace concept called pie-traces (which stands for passive-information-extended-traces). We will see that this notion of trace is indeed strong enough to give a trace semantics for $|^P_A|$. A pie-trace in this semantics looks like

$$P_1\alpha_1P_2\alpha_2\cdots P_n\alpha_n,$$

where $\alpha_i \in \Sigma \cup \bar{\Sigma}$ and $P_i \subset \bar{\Sigma}$. We could say that part of the tree-structure (as far as it concerns the passive actions) is contained in these pie-traces. $P_1\alpha_1P_2\alpha_2\cdots P_n\alpha_n$ means that within the transition system, there exists an execution of trace $\alpha_1\cdots\alpha_n$ such that at the state in the transition system where α_i is to be executed there are outgoing passive transitions for each $\bar{\alpha} \in P_i$ $(i = 1\cdots n)$ and there are no outgoing passive transitions for α if $\alpha \notin P_i$.

The processes in figure 5 have the following pie-traces: $Tr_{pie}(X_1) = \{a[\bar{c}]b, aba, ab, a\}$. Here $a[\bar{c}]b$ means P_1aP_2b with $P_1 = \emptyset$, $P_2 = \{\bar{c}\}$ etc. $Tr_{pie}(X_2) = \{a[\bar{c}]ba, ab, a[\bar{c}]b, a\}$ and $Tr_{pie}(Y) = \{c\}$. With this notion of trace semantics, X_1 and X_2 have different semantics, thus they are not equivalent with respect to this notion of trace semantics. We now show that our notion of trace semantics is strong enough for a trace semantics of the composition operator $|^P_A|$:

Let σ_1 and σ_2 be pie-traces. Then we define $\sigma_1|^P_A|\sigma_2$, which will turn out to be the set of interleavings of σ_1 and σ_2 with respect to $|^P_A|$, as follows:

- $\epsilon|^P_A|\epsilon := \{\epsilon\}$
- $(R_1\alpha_1\sigma_1')|^P_A|\epsilon := \{\epsilon\} \cup S_1$, where $S_1 := R_1\alpha_1(\sigma_1'|^P_A|\epsilon)$ if $(\alpha_1 \in \Sigma$ and $\alpha_1 \notin A)$ or $\alpha_1 \in \bar{\Sigma}$ else $S_1 := \emptyset$.
- $\epsilon|^P_A|(R_2\alpha_2\sigma_2') := \{\epsilon\} \cup S_1$, where $S_1 := R_2\alpha_2(\epsilon|^P_A|\sigma_2')$ if $(\alpha_2 \in \Sigma$ and $\alpha_2 \notin A)$ or $\alpha_2 \in \bar{\Sigma}$ else $S_1 := \emptyset$.
- $(R_1\alpha_1\sigma_1')|^P_A|(R_2\alpha_2\sigma_2') := \{\epsilon\} \cup S_1 \cup S_2 \cup S_3 \cup S_4$, where
  $S_1 := (R_1 \cup R_2)\alpha_1(\sigma_1'|^P_A|\sigma_2)$ if one of the cases r3,r4 or r6 is true, else $S_1 := \emptyset$,
  $S_2 := (R_1 \cup R_2)\alpha_1(\sigma_1'|^P_A|\sigma_2')$ if one of the cases r1,r2 or r5 is true, else $S_2 := \emptyset$,
  $S_3 := (R_1 \cup R_2)\alpha_2(\sigma_1|^P_A|\sigma_2')$ if one of the cases r3',r4' or r6' is true, else

$S_3 := \emptyset$,
$S_4 := (R_1 \cup R_2)\alpha_2(\sigma_1'|_A^P|\sigma_2')$ if case r2' is true, else $S_3 := \emptyset$.

Cases:
r1: $\alpha_1 = \alpha_2 \in \Sigma$ and $\alpha_1 \in A$
r2: $\alpha_1 \in \Sigma$ and $\alpha_2 = \bar{\alpha}_1$ and $\alpha_1 \notin A$
r3: $\alpha_1 \in \Sigma$ and $\bar{\alpha}_1 \notin R_2$ and $\alpha_1 \notin A$
r4: $\alpha_1 \in \bar{\Sigma}$ and $\alpha_1 \notin P$
r5: $\alpha_1 = \alpha_2 \in \bar{\Sigma}$ and $\alpha_1 \in P$
r6: $\alpha_1 \in \bar{\Sigma}$ and $\alpha_1 \notin R_2$ and $\alpha_1 \in P$
r2': $\alpha_2 \in \Sigma$ and $\alpha_1 = \bar{\alpha}_2$ and $\alpha_2 \notin A$
r3': $\alpha_2 \in \Sigma$ and $\bar{\alpha}_2 \notin R_1$ and $\alpha_2 \notin A$
r4': $\alpha_2 \in \bar{\Sigma}$ and $\alpha_2 \notin P$
r6': $\alpha_2 \in \bar{\Sigma}$ and $\alpha_2 \notin R_1$ and $\alpha_2 \in P$

Theorem 2. *σ is a pie-trace of $X|_A^P|Y$ if and only if there exist pie-traces σ_x and σ_y of X and Y respectively such that $\sigma \in \sigma_x|_A^P|\sigma_y$.*

Proof. It can be seen that the cases r1 till r6 and r2',r3',r4' and r6' correspond to the composition rules r1 till r6, r2',r3',r4' and r6'. This correspondence is such that when case r1 is true (from the initial states), then composition rule r1 can be applied in the composition and when case r2 is true then composition rule r2 can be applied, etc. Now it is easy to check that $\sigma_x|_A^P|\sigma_y$ is exactly the set of all interleavings (including synchronizations) of σ_x and σ_y that are accepted by the composition rules of $|_A^P|$, from which the result follows.

Similar to the definition of refusals in [10], we can now define pie-refusal traces of the form σX, where σ is a pie-trace and X a set of active actions that can be refused after σ. In this way a semantics is defined that reduces to the standard testing semantics in the absence of passive actions.

4 Conclusions and Outlook

In this paper we used active and passive transitions to model both blocking and non-blocking interaction between processes. We motivated that the use of active/passive transitions is particularly interesting for modelling supervisory control systems. Supervisory controllers have two distinct actions: observing and controlling, which can be modelled naturally with passive and active transitions respectively. With the use of active/passive transitions, we are able to avoid the problems that arise in the 'blocking' framework when it comes to modelling uncontrollable process actions.

We introduced the composition operator $|_A^P|$ for the active/passive framework by means of structural operational rules. By using *pie-traces*, we have also given a trace semantics for this operator.

With the active/passive framework and the operators $[\cdot]_C$ and $|_A^P|$, we have given two tools for the modular specification of control systems. First, with

$[\cdot]_C$ we can control the scope of the observations within the composite system. Secondly, with $|^P_A|$ we can establish synchronization of observations (which results in multi-may synchronizations) via the set P.

We think that many phenomena, in particular supervisory control systems, can be modelled naturally within the active/passive framework by making use of operators like $|^P_A|$ and $[\cdot]_C$. For future research, we want to explore the use of $|^P_A|$ in a hybrid (supervisory control) context. First steps in this direction can be found in [11]. Another interesting question is whether stochastic aspects can be incorporated in the $|^P_A|$-active/passive framework. In [12], we see the use of $|^{\bar{\Sigma}}_{\emptyset}|$ enhanced with stochastic aspects for the specification of Piecewise Deterministic Markov Processes.

References

1. Plotkin, G.D.: A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, Comp. Sci. Dept. (1981)
2. Hoare, C.: Communicating Sequential Processes. Prentice-Hall (1985)
3. Bolognesi, T., Brinksma, E.: Introduction to the iso specification language lotos. Comp. Networks and ISDN Systems 14 (1987) 2559
4. Prasad, K.: A Calculus of Broadcasting Systems. In: Proc. 16th Colloquium on Trees in Algebra and Programming. Volume 493. (1991) 338358
5. Lynch, N.A., Segala, R., Vaandrager, F.W.: Hybrid I/O automata. Information and Computation 185(1) (2003) 105157
6. Lynch, N.A., Tuttle, M.R.: An introduction to input/output automata. CWI Quarterly 2 (1988) 219246
7. Groote, J.F.: Process Algebra and Structured Operational Semantics. PhD thesis, University of Amsterdam (1991)
8. Ramadge, P., Wonham, W.: The control of discrete event systems. Proceedings of the IEEE 77 (1989) 8198
9. Cassandras, C.G., Lafortune, S.: Introduction to discrete event systems. Kluwer Academic Publishers (1999)
10. Hennesy, M.: Algebraic Theory of Processes. MIT Press (1988)
11. Julius, A.A., Strubbe, S.N., van der Schaft, A.J.: Control of hybrid behavioral automata by interconnection. In: Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03. (2003) 135140
12. Strubbe, S.N., Julius, A.A., van der Schaft, A.J.: Communicating piecewise deterministic markov processes. In: Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03. (2003) 349354