

Chapter 8

Control in Compositional Systems and Multi-Agent Systems

This chapter is dedicated to the phenomenon of control in compositional systems and multi-agent systems. The aim of this chapter is to develop constructs for modelling control in multi-agent systems as a refinement of the semantic structure presented in the previous chapters. In other words, in this chapter, new commitments are introduced that further characterise the constructs provided by the semantic structure. These commitments determine how one component in a compositional system can control another component. Furthermore, additional modelling choices are presented that illustrate how control in a compositional system can be used to model control in a multi-agent system.

The previous chapters discuss the influence of information exchange between components on the dynamics of the components. Under specific conditions, this influence enables one component to control other components. Thus, the semantic structure presented in the previous chapters provides necessary constructs to support control. However, to apply the semantic structure, refinements of the constructs provided by the semantic structure are needed that resolve the following issues. First, although the semantic structure presented in the previous chapters ensures *that* one component can influence another component, the question of *what* is influenced and *what* this influence entails, is not addressed. Second, the constructs provided by the semantic structure do not address questions of how control can be exercised in a *domain-independent* and *compositional* way. Third, the *scope* of influence exercised by each individual component in a compositional system is not clear. These issues are addressed in this chapter.

The structure of this chapter is as follows. To provide a strong foundation from which to pursue the aim of this chapter, first, in Section 8.1, the control phenomenon is characterised, the role of control in compositional systems is discussed, and some perspectives from other research areas are presented. The next two sections have a structure similar to Chapter 2 and Chapter 3. Thus,

8.1: Control and Compositionality

Section 8.2 presents commitments with respect to how control in compositional systems is represented. This section explains how constructs provided by the semantic structure can exercise control over components. Section 8.3 discusses control in multi-agent systems. Although the autonomy of an agent in a multi-agent system seems to be incompatible with control over that agent, the characterisation of control presented in Section 8.1 is also applicable to an autonomous agent. Section 8.3 presents some modelling choices with respect to how control in multi-agent systems can be represented in a compositional system that models the multi-agent system. Section 8.4 provides an example of the material presented in the previous sections. The final topic is the relationship between controlling components and the components they control. The notion of global state developed in the previous chapter is exploited to investigate this relation. This topic is covered in Section 8.5.

8.1 Control and Compositionality

The phenomenon of control appears in many different guises, both in multi-agents systems as well as in conventional Computer Science and Artificial Intelligence. The control phenomenon is encountered whenever a specific part of a system adopts a goal that requires *another part* of the system to reach a specific state. In other words, a part of a system that tries to constrain the future of another part such that this other part reaches a specific state, has to exercise control over the part that is to be constrained. For instance, in a multi-agent system with robot agents, one of the robots may adopt a goal to change something in the environment (e.g., to paint a car in an assembly line), or to change the state of another robot such that this robot moves out of the way. In both cases, the agent faces a control task: it needs to execute actions in the environment, perform observations or exchange information, to exercise influence with the effect of changing the current state of the environment or agent to the desired state.

The previous paragraph deliberately characterises control in terms of *parts* of a system. Indeed, control is exercised in all kinds of systems, by different kinds of parts. In a compositional system, one component may control other components. In a multi-agent system, one agent may control other agents.

Although the control phenomenon appears in many different guises, it is possible to develop a common characterisation of control tasks, as is indicated in Chandrasekaran (1994). Chandrasekaran studies exercising control from a knowledge-level point of view. His goal is to answer the following question: “What is it that unifies the control task in all its manifestations, from the thermostat to the operator of a nuclear power plant?” (Chandrasekaran, 1999). The answer he derives consists of a process model of control, and provides a general characterisation of what comprises exercising control. The general process model for control consists of the following subprocesses. First, either implicitly or explicitly, by observing the controlled components, a controlling component

constructs a descriptive model of the past and present behaviour of the controlled components. The control component then derives a prescriptive extension of this model, for instance by instantiating a (possibly pre-compiled) plan. This extension describes the intended future behaviour of the controlled components. Using this extension, the control component tries to influence the behaviour of the controlled components by transmitting information to them. After that, the control component may obtain new observations with which the model of the past and present behaviour of the controlled components can be updated, after which the control process is repeated.

As the quotation from Chandrasekaran suggests, the control phenomenon spans a very diverse spectrum of domains, from a thermostat to a nuclear power plant operator or the president of a national bank. According to Chandrasekaran, in all of these domains, exercising control has a similar structure, summarised as follows: a controlling component repeatedly attempts to influence other components by transmitting information to these components, and uses information transmitted from these components as feedback. Thus, the essence of control is information exchange. The constructs provided by the semantic structure are sufficient to enable the construction of compositional systems that execute control tasks as characterised by Chandrasekaran because these constructs support information transmission.

In this chapter, the semantic structure is refined (i.e., additional properties of constructs and relations between constructs are distinguished and committed to) with the aim of providing *better support for control*. The refined semantic structure supports building compositional systems in which control is *separated* and as *domain-independent* as possible. As explained in Section 8.1.1, separated, domain-independent control provides a more maintainable and reusable structure. To separate control, the additional properties and relations for control require that control and control information can be distinguished in a compositional system. As, due to the many different guises of the control phenomenon, it is difficult to distinguish control information from other information by a general definition, another approach is adopted: users of the semantic structure can *designate* specific components and information to be control components and information. The additional properties and relations are applicable to the components and information designated to be specific for control. Designating control components and information is the subject of Section 8.1.2. Perspectives on control from other areas are presented in Section 8.1.3 The additional properties and relations are presented in Section 8.2.

8.1.1 Separate, Domain-independent control

In most branches of engineering, systems are partitioned in components to reduce overall complexity and enhance reuse, modification and maintenance. The extent to which these goals are met depends on how a specific system is partitioned. Therefore, an important research topic in Computer Science and Artificial

8.1: Control and Compositionality

Intelligence is the investigation of design principles that guide software engineers in finding a good compositional structure. (Recent developments in this area take place within the study of software architecture (Perry & Wolf, 1992; Bass, Clements, Kazman & Bass, 1998), design patterns (Gamma, Helm, Johnson & Vlissides, 1995) and co-ordination languages (Papadopoulos & Arbab, 1998).)

A design principle applied in Computer Science and Artificial Intelligence is to separate a computation's logic and control, and make both equally explicit. Although this principle appears in many areas of Computer Science and Artificial Intelligence, it is probably most famous as the foundation for logic programming languages as presented in the paper by Kowalski (1979). In the area of knowledge based systems design, Clancey (1983, 1992) and Neches, Swartout and Moore (1985) emphasise the importance of separating domain knowledge (*what* knowledge to apply) from control knowledge (*how* to apply that knowledge). In generic architectures for knowledge based systems, control knowledge is recognised as a separate type of knowledge (Chandrasekaran, Johnson & Smith, 1992; Brazier, Treur & Wijngaards, 1996). In all of these areas, separating control knowledge results in systems that are more maintainable and easier to modify and reuse.

In a compositional system with a recursive structure of subcomponents, subcomponents of subcomponents, and so on, control is separated at each level. This implies that at each level, some components are control components, while other components are controlled components. In the terminology of Kowalski (1979), the controlled components contain the logic of the computation that is carried out collaboratively by the control components and controlled components. In the terminology of knowledge based systems, the controlled components contain domain knowledge such as, for instance, inference relations. In a compositional system, controlled components themselves actively perform computations and may themselves consist of control subcomponents and subcomponents controlled by the control subcomponents. These control subcomponents and controlled subcomponents collaboratively perform the computations of their parent component, which is a controlled component at its level. Thus, although in Kowalski's terms, a controlled component (only) contains the logic of a computation, if the controlled component is composed, there may be control inside the component.

Figure 8.1 illustrates separated control in a compositional system. In this figure, components drawn with thick lines are control components, while the other components are controlled components. All links depicted in Figure 8.1 are used to transmit control information. The figure shows that the control component that is a subcomponent of the controlled component receives control information and provides feedback information for the control component at its parent level. In Figure 8.1, the specific part of each input and output interface that is reserved for control information, is shown in gray.

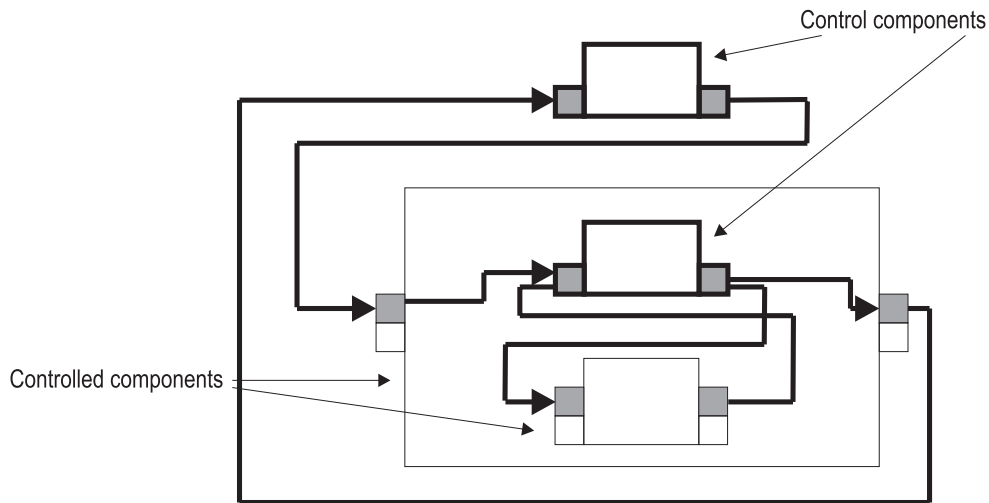


Figure 8.1: Control in a compositional system.

Control and control information separated in a compositional system should be domain independent to support maintainability and ease of modification and reuse. Computations performed by a component are specific for a particular domain. For instance, in a compositional system that represents a nuclear power plant, computations that represent the operator are e.g. co-operation with other operators in the plant and to diagnosis of reactor malfunctioning. These computations process domain specific information, such as e.g. information on the status of the reactor in the nuclear power plant or on the intentions of other operators. Control information should contain as little domain information as possible and should instead consist of domain-independent information on e.g. activating components and enabling information transmission. In other words, control information should be as domain-independent as possible. Note that, apart from control information, other information can also be domain-independent, or *generic*. As an example, consider mathematical knowledge, which is present in almost every domain, but is not control information.

8.1.2 Designating Control Information

The previous subsection presented reasons for separating control in a compositional system. This subsection explains *which* information to separate as control information. As stated at the beginning of Section 8.1, the control phenomenon is encountered whenever part of a system tries to influence another part of the system in an attempt to constrain the future of the other part. In a compositional system, these parts are components. Components use information exchange to influence each other. Therefore, all information exchange has a potential control effect. In other words, it is difficult to distinguish information

8.1: Control and Compositionality

transmission for control from other information transmission. Consider, for example, the president of a national bank, who tries to control the economic behaviour of companies and customers. The bank president can influence the behaviour of companies and customers in several ways. One way is to change the official interest rate and transmit information on the new rate to all companies and customers. However, this information does not *force* a change of behaviour, as companies and customers are free to continue their behaviour. Nevertheless, an interest rate change is usually explained as an attempt to control the behaviour of domestic customers and companies. However, the interest rate may have been changed for other reasons, or other information provided by the bank president also changed the behaviour of customers and companies. As every information transmission changes the state of destination components, all information transmission has a potential control effect. (This example domain has been suggested by Chandrasekaran.)

For reasons put forward in the previous subsection, it is important to distinguish information transmission for control from other information transmission. As all information transmission has a potential control effect, an additional criterion has to be developed. There are two possibilities for such a criterion. The first possibility is to require that information transmissions must be performed *intentionally* for control to classify as transmissions of control information. In the example of the bank president, transmitting information on a new interest rate is only a transmission for controlling domestic customers and companies if the president has the intention of changing the economic behaviour of those domestic customers and companies and changed the interest rate because of this intention. This possibility requires insight in the intentions and relationship between intentions and transmitted information to distinguish information transmission for control from other information transmission. This might be suitable in a multi-agent system in which intentions of agents are explicitly modelled. However, in the more general context of compositional systems, this possibility is not realistic.

The other possibility is to avoid committing to a specific characterisation and instead provide a flexible way for users of the semantic structure to *designate* specific information as control information. Specific properties of constructs provided by the semantic structure and specific relations between constructs are only applicable to information designated as control information. Users of the semantic structure can choose which information they wish to subject to these specific properties and relations. Information is designated as control information by associating it, for example, with a reserved part of the input and output interfaces of a component. Consequently, for the state of a component, two substates can be distinguished: called the control state and the domain state.

The specific properties and relations for control provided by the semantic structure enable separating control that is as domain-independent as possible. Moreover, there is an additional property of information designated as control:

control information sent to a controlled component not merely influences this component, it also constrains the behaviour of the controlled component. Control information specifies one or more possible futures for the reserved control information substate of a controlled component. The properties and relations committed to (see Section 8.2) guarantee that the controlled component cannot choose another future but the specified one. This is the most important property of control information and should determine, for a specific application, which information is designated as control information.

As an example, the information on a new interest rate transmitted by the bank president should probably not be designated as information transmission for control: a specific customer might choose to neglect the changed interest rate and continue with his or her behaviour (although this might not be rational). As an example of information transmission that should be designated as a control transmission, consider a nuclear power plant operator. If he or she detects a situation that requires an immediate stop of the reactor, probably by pushing a button he or she commands a specific component of the plant to drop bars of a moderating material into the reactor. The content information of pushing the button (pushing the button is the encapsulated form of the content information, see Section 2.2.4) is: “drop the moderator”, or “stop the reactor”. The reactor cannot choose to neglect this command: pushing the button breaks a circuit to an electromagnetic device holding the bars containing the moderator. The laws of physics guarantee that the bars fall into the reactor, which slows down or stops. The specified future is not guaranteed: if the button is broken or if someone has removed the moderator bars, the future may not be as expected. However, this is beyond the reactor’s control.

8.1.3 Some Perspectives on Control

Different perspectives on the concept of control can be found in related areas:

- In the context of (formal) languages for the specification of algorithms, such as programming languages, the concept of control refers to the sequencing of steps in an algorithm. The characterisation of control developed by Chandrasekaran is also applicable in this context. In his terminology, the specification or (compiled) computer program tries to influence a state space (e.g., a computer memory) to constrain the set of all possible contents of this space. In this context, operations to influence and observe the state space are generally assumed to be infallible (e.g., read and write operations in a computer). The specification or program text itself can be viewed as knowledge provided by a software engineer that is used by the compiled program to, in Chandrasekaran’s terminology, derive a prescriptive extension of the model of the behaviour of the state space. This knowledge is formalised in different ways. In imperative programming languages, so-called *control structures* such as ‘if ... then’ and ‘do ... while’ are used. In

8.1: Control and Compositionality

logic programming languages, control is fixed and implicit in the semantics of the language. In (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, in press), formalisms for specification of dynamics in Software Engineering and Information Systems are surveyed from a Knowledge Engineering perspective. They identify two major differences between the formalisms studied. First, there is a distinction between constructive and constraining control specifications. In a constructive specification, control is specified by stating how valid sequences of states are constructed, often in the style of an imperative programming language. In a constraining specification, control is specified by a set of expressions, such as, for instance, temporal logic formulae, that constrain the set of all possible sequences. Second, there is a distinction between step-based and sequence-based specification of control. In a step-based specification, control can only be described in terms of the relation between a begin state and an end state. In a sequence-based specification, control can also be described by reference to intermediate states.

- Control of one agent over another seems to be incompatible with the autonomy of agents in a multi-agent system. Indeed, a property of an autonomous agent is that it is impossible for other agents to unconditionally set the agent's goals in a way that is inescapable for that agent. However, the concept of control sketched at the beginning of Section 8.1 does not assume that this is possible: an agent that controls another agent merely tries to influence the controlled agent to adopt a specific goal. In the area of multi-agent systems, control is related to co-ordination and management. This topic is addressed further in Section 8.3.
- In the CommonKADS approach to Knowledge Engineering (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, Velde & Wielinga, 1999), the control phenomenon appears in to some extent in most of the models that constitute the CommonKADS model suite. (The CommonKADS model suite consists of six models, divided over three levels. At the highest level, the organisation, task and agent models describe the environment in which knowledge-intensive systems are applied. The middle level consists of the knowledge model and the communications model, which describe knowledge-intensive tasks and information exchange in a conceptual and implementation-independent way. The lowest level consists of the design model, which describes the (computer) system that is constructed to implement the knowledge and communication models.) First, at the environment level, control is scattered over all three models. The agent model describes various attributes of agents, including competences, responsibilities and communication relations of individual agents. These attributes indicate, implicitly, possible control relations. The organisation model describes formal and informal power relations between agents in the

organisation. In a sense, this is a static view on control, which focuses on the question which agent has the power to control which other agent(s). The task model covers knowledge of time-related aspects of tasks that constitute the process modelled by a CommonKADS model. Thus, this is control *over* tasks. Presumably, specific agents have to exercise control to ensure proper timing and order of tasks. However, the CommonKADS approach does not provide constructs to make explicit which agents control which tasks. Second, control for knowledge-intensive tasks identified in the task model is described in the knowledge model. This is control *within* the task, and describes how the goals of a task can be achieved in terms of subtasks and inferences. This form of control is described using a pseudo-code language in an imperative style. However, control in the knowledge model is only used to describe the task at a conceptual level, in a sense to help characterise the task. In the knowledge model, the entity that is responsible for exercising control to ensure that subtasks and inferences are executed in the order specified, is not distinguished. Third, the design model covers control in the (computer) system that is constructed to implement the knowledge model (and communication model). The design model consists of an architecture design, an application design, and a platform design. There is no commitment to a specific form of control. Instead, properties of control depends on the choices made for the skeletal architecture (e.g., rule-based), the application design (e.g., search), and the platform (i.e., languages and support environment, such as OPS5). However, the preferred choice in the CommonKADS approach is a design model that preserves the structure of the knowledge and communication models. Such a structure-preserving design inherits properties of control from the knowledge and communication models.

8.2 Constructs for Control in Compositional Systems

Chapter 2 presented commitments that characterise the constructs for information transmission in the semantic structure developed in this thesis. In this section, a similar characterisation of additional commitments for support of control is presented. In the next subsections, the following commitments are discussed:

- All control is explicitly represented in the compositional system. If, for any component, control is to be exercised, a component responsible for exercising this control is appointed (Section 8.2.1);
- Control is exercised by components that do not have a domain-dependent function (Section 8.2.2);
- For each component, a part of the state of both the component's input and output is distinguished as the control substate (Section 8.2.3);

8.2: Constructs for Control in Compositional Systems

- Control information is transmitted using dedicated links, called *control links* (which, apart from their dedication, are normal links). These control links connect control components and (the control part of the interfaces of) the controlled components (Section 8.2.4);
- A control component can directly activate control links to which it is connected (Section 8.2.5);
- Control information constrains the future of controlled components (Section 8.2.6).

8.2.1 Component Responsible for Control

The control phenomenon is often encountered in the following way. During analysis or design of a multi-agent system, the need for a specific order of subprocesses within an agent is identified. Control is needed for this purpose. The question is: which, if any, process is responsible for exercising the control needed?

In a compositional system that represents the multi-agent system, each component may, in principle, be active completely independent from all other components. There is no 'automatic', or default order or execution of processes. Therefore, the execution order determined has to be actively established by either the subcomponents that represent the subprocesses themselves, or by other components. The semantic structure developed in this thesis is not committed to either choice. However, it is not possible to leave the choice implicit.

In contrast, it is possible to leave implicit which part of a system is responsible for exercising control in many other modelling frameworks. In those frameworks, the execution order determined during analysis and/or design is specified using a notation such as pseudo-code (as in CommonKADS, see Section 8.1.3), some form of dynamic logic, or temporal logic. The semantics of such a notation specifies how this notation is to be interpreted. If the semantics is specified in an operational style (e.g., Plotkin, 1981,1982), the interpretation is given in the form of a mechanism that is able to exercise control as specified. However, this mechanism is only given as a means to find the interpretation of the specification of the proper execution order for the subprocesses. Such frameworks do not explicitly assume the existence of an entity that actually controls the subprocesses and which itself is a part of the system. As an example, consider an algorithm specified in an imperative programming language such as Pascal. The designer of the algorithm specifies the proper execution order of the primitive Pascal statements by means of the control statements provided by the language (e.g., repeat ... until, do ... while). The microprocessor of the computer that executes the program (more or less automatically) carries out the primitive statement in the order specified, and this is well known to all users of the language. However, it is important to note that the component that is actually responsible for the proper execution order is not represented explicitly in the specification of the algorithm. (For a language

designed for sequential computers such as Pascal, it is implicitly assumed that there is only one component that is active at a time, therefore explicit representation of this component is irrelevant.)

In a multi-agent system, there is by definition more than one entity in the system that is active and, at least in principle, able to exercise control over other entities. Nevertheless, it is possible to specify that subprocesses of agents are to be executed in a specific order. In this case, however, confusion may arise as to which component is responsible for exercising the control needed to ensure that subprocesses are executed in the specified order. Usually, a fixed choice is made in the semantics of the notation used for the specification.

For the refinement of the semantic structure developed in this chapter, the situation is slightly different, as there is no specification notation associated with the semantic structure or its refinement. Apart from this, the question of which component is responsible for exercising control is equally relevant. As stated at the beginning of this section, in the refinement of the semantic structure, there is no default choice, and the question has to be answered explicitly.

8.2.2 Dedicated Control Components

The second commitment is the commitment to dedicated control components. This commitment states that, in a compositional system, control is exercised by specifically designated components, called *control components*, that do not have a domain-dependent function. However, apart from their dedication, control components are not special: they are normal components. Control components carry out the process described by Chandrasekaran. In other words, the process of exercising control can be found in these components. A control component can only control components that are subcomponents of the same parent component as the control component itself. As a control component is, apart from its dedication, a normal component, a control component itself may be a composed component. In this case, one of the subcomponents of the control component is itself a control component, which controls the other subcomponents. However, at the level of the parent component, this is not important.

An additional commitment can be distinguished with respect to dedicated control components. This additional commitment is discussed in Section 8.2.2.1.

8.2.2.1 Multiplicity of Control Components

In the semantic structure developed in this thesis, it is assumed that there is exactly one control component in each composed component. As a consequence, all other subcomponents are controlled components, and these controlled components are controlled by the unique control subcomponent of the parent component. This commitment is the most simple option.

An alternative approach is to allow more than one control component within a specific composed component. In this case, the set of controlled components has to

8.2: Constructs for Control in Compositional Systems

be partitioned in subsets of subcomponents that are controlled by the same control component to avoid conflicting requirements imposed by different control components on the same controlled component. Such a partition could be enforced by the introduction of a new composed component for each partition with all controlled components in the partition and their control component as subcomponent. In this case, the result is that each composed component has exactly one control component.

As each composed component has exactly one control component, all subcomponents that are not control components are controlled components. (If composed components that do not have a control component were allowed, then subcomponents of that component would not be controlled at all: they are able to behave in any conceivable way, without any influence by other components.) As a consequence, all components that are not control components are potentially constrained in their behaviour. However, applications of the semantic structure may choose, for specific composed components, to specify a control component that enforces none, or almost no restrictions on the behaviour of the subcomponents it controls.

8.2.3 Dedicated Control Substates

As explained in Section 2.1.1, the semantic structure distinguishes a state for each component and link. The state of a component consists of three substates: the state of the input interface, the state of the output interface and an internal state. To separate control as described in Section 8.1.1, the substates are partitioned further: each of the three substates is divided in a control part and a domain part. Formally, this can be accomplished by defining the sets $S_{C,in}$, $S_{C,int}$ and $S_{C,out}$ for a component C as unions of sets of control states and domain states. The partition of each substate in a control part and a domain part also holds for the interfaces: each interface is divided in a control part and a domain part. Likewise, the state of a link is divided in a control substate and a domain substate.

As explained in Section 8.1.2, applications of the semantic structure can designate specific information to be control information. The semantic structure uses the following principle for designating information to be control information: by placing information in the control part of an interface, it is designated to be control information. Consequently, control information is the information that determines the control part of each substate. Control is thus separated from domain information in a compositional system and appears in specifically designated control components and in the control part of the substates of all components. The (behaviour of the) control parts is subject to the constraint mentioned in Section 8.1.2: a controlled component behaves as directed. This is made more precise in Section 8.2.6. In Chapter 9, syntactic structures for specifying information as control information are presented. With these structures, control states can be described using domain-independent terms from an ontology for the specification of control (Chandrasekaran, Josephson & Benjamins, 1998.)

Formally, control information is captured as follows:

Definition 8.1. (Control and domain states).

- Let C be a component with set of states $\mathcal{S}_C = \langle \mathcal{S}_{in}, \mathcal{S}_{int}, \mathcal{S}_{out} \rangle$ such that $\mathcal{S}_X = \mathcal{S}_{X,c} \times \mathcal{S}_{X,d}$ for $X \in \{in, int, out\}$. Then $\mathcal{S}_{in,c}$ is the component's set of control states and $\mathcal{S}_{X,d}$ is the component's set of domain states.
- Let I be a link with set of states \mathcal{S}_I such that $\mathcal{S}_I = \mathcal{S}_{I,c} \times \mathcal{S}_{I,d}$. Then $\mathcal{S}_{I,c}$ is the link's set of control states and $\mathcal{S}_{I,d}$ is the link's set of domain states.

8.2.4 Dedicated Control Links

Chandrasekaran characterises control as an attempt (by a control component) to influence another component, based on information on the state of the other component. Thus, control is exercised using information transmission. Similar to control components and control substates, information transmission specific for control is separated from other information transmission. In the semantic structure, control information is transmitted by dedicated control links, which are, apart from their dedication, normal links. There are four kinds of control links. First, there are control links from a control component to a controlled component or link. These control links, called *downward control links*, transmit control information that specifies the initial part of the future of the controlled component or link. Second, there are control links from a controlled component or link to a control component. These control links, called *upward control links*, transmit feedback control information that is used by the control component to evaluate the effect of control it exercised. Third, there are *import control links* which transmit control information from the control part of the input interface of a composed component to the control part of the input interface of the control component in this composed component. Fourth, there are *export control links* which transmit control information from the control part of the output interface of the control component in a composed component to the control part of the output interface of this composed component.

Thus, dedicated control components and dedicated control links are distinguished. As explained in Chapter 5, the structure of a compositional system is described by a structure hierarchy. The designation of dedicated control components and links within a structure hierarchy is formally captured by the definition of a special class of structure hierarchies, called *structure hierarchies with control*. Informally, a structure hierarchy $SH = \langle Comp; Lnk; \prec; dom; cdom \rangle$ is a structure hierarchy with control if the following requirements hold:

- The set of components $Comp$ can be partitioned in a set of control components $Contr$ and a set of controlled components $Comp'$;
- And the set of links Lnk can be partitioned in a set of upward control links $UCLnk$, a set of downward control links $DCLnk$, a set of import control links $ICLnk$, a set of export control links $ECLnk$ and a set of controlled links Lnk' ;

8.2: Constructs for Control in Compositional Systems

- And for each control component, there is a (control or controlled) component of which it is a subcomponent, together with an import control link I_1 from the input interface of the parent component to the control component and an export control link from the control component to the output interface of the parent component;
- And for each controlled component C that is not primitive, there is exactly one control component C' such that C' is a subcomponent of C and for each other subcomponent or link S of C , there is a downward control link I_1 from C' to S and an upward control link I_2 from S to C' ;
- And for each downward control link I in a composed component C , there is a control component C' and a controlled subcomponent or link S of C such that I is a link from C' to S ;
- And for each upward control link I in a composed component C , there is a control component C' and a controlled subcomponent or link S of C such that I is a link from S to C' ;
- And for each import control link I , there is a control component C and a component P such that C is a subcomponent of P and I is a link from the input interface of P to the input interface of C ;
- And for each export control link I , there is a control component C and a component P such that C is a subcomponent of P and I is a link from the output interface of C to the output interface of P ;

The eight informal requirements directly correspond with the eight requirements in the following definition:

Definition 8.2. (Structure hierarchy with control). Let $SH = \langle \text{Comp}; \text{Lnk}; \prec; \text{dom}; \text{cdom} \rangle$ be a structure hierarchy. This structure hierarchy is a structure hierarchy with control if:

- $\text{Comp} = \text{Contr} \cup \text{Comp}'$ with Contr and Comp' disjoint;
- $\text{Lnk} = \text{Lnk}' \cup \text{UCLnk} \cup \text{DCLnk} \cup \text{ICLnk} \cup \text{ECLnk}$ with Lnk' , UCLnk , DCLnk , ICLnk and ECLnk pairwise disjoint;
- For all $C \in \text{Contr}$ there is a component $C' \in \text{Comp}$, a link $I_1 \in \text{ICLnk}$ and a link $I_2 \in \text{ECLnk}$ such that $C \prec C'$, $I_1 \prec C'$, and $I_2 \prec C'$, $\text{dom}(I_1) = C'$, $\text{cdom}(I_1) = C$, $\text{dom}(I_2) = C$ and $\text{cdom}(I_2) = C'$;
- For all $C \in \text{Comp}'$ such that $C \notin \text{Prim}(SH)$, there is exactly one $C' \in \text{Contr}$ such that $C' \prec C$ and for all S in $\text{Comp} \cup \text{Lnk}$, if $S \prec C$ and $S \neq C'$, then there is an $I_1 \in \text{DCLnk}$: $\text{dom}(I_1) = C'$ and $\text{cdom}(I_1) = S$ and there is an $I_2 \in \text{UCLnk}$: $\text{dom}(I_2) = S$ and $\text{cdom}(I_2) = C'$;

- For all $I \in DCLnk$ such that $I < C$ for $C \in Comp$, there is a $C' \in Contr$ and an $S \in Comp \cup Lnk$ such that $dom(I) = C'$ and $cdom(I) = S$ and $S, C' < C$;
- For all $I \in UCLnk$ such that $I < C$ for $C \in Comp$, there is a $C' \in Contr$ and an $S \in Comp \cup Lnk$ such that $dom(I) = S$ and $cdom(I) = C'$ and $S, C' < C$;
- For all $I \in ICLnk$, there is a component $C \in Contr$ and a component $P \in Comp$ such that $C < P$, $dom(I) = P$ and $cdom(I) = C$;
- For all $I \in ECLnk$, there is a component $C \in Contr$ and a component $P \in Comp$ such that $C < P$, $dom(I) = C$ and $cdom(I) = P$.

Example 8.3. In Chapter 5 (Example 5.10), a structure hierarchy $sh = \langle Comp; Lnk; <; dom; cdom \rangle$ was presented, with

- $Comp = \{toplevel, user_1, broker, ASP, OPC\}$;
- $Lnk = \{user_1_to_broker, broker_to_user_1\}$;
- $< = \{ \langle ASP; broker \rangle, \langle OPC; broker \rangle, \langle user_1; topLevel \rangle, \langle broker; topLevel \rangle, \langle user_1_to_broker; topLevel \rangle, \langle broker_to_user_1; topLevel \rangle \}$;
- $dom = \{ \langle user_1_to_broker; user_1 \rangle, \langle broker_to_user_1; broker \rangle \}$;
- $cdom = \{ \langle user_1_to_broker; broker \rangle, \langle broker_to_user_1; user_1 \rangle \}$;

This structure hierarchy describes the structure of a part of the running example system. (Only one user agent was included in the example structure hierarchy to keep the example concise.) The structure hierarchy with control for this example is the structure hierarchy $sh' = \langle Comp'; Lnk'; <; dom'; cdom' \rangle$, with:

- $Comp' = Comp \cup Contr$, with $Contr = \{toplevel_control, broker_control\}$;
- $Lnk' = Lnk \cup UCLnk \cup DCLnk \cup ICLnk \cup ECLnk$, with:
 - $UCLnk = \{broker_UCL, user_1_UCL, OPC_UCL, ASP_UCL\}$;
 - $DCLnk = \{broker_DCL, user_1_DCL, OPC_DCL, ASP_DCL\}$;
 - $ICLnk = \{toplevel_ICL, broker_ICL\}$;
 - $ECLnk = \{toplevel_ECL, broker_ECL\}$;
- $<' = < \cup \{ \langle broker_control; broker \rangle, \langle topLevel_control; topLevel \rangle, \langle topLevel_ICL; topLevel \rangle, \langle topLevel_ECL; topLevel \rangle, \langle broker_ICL; broker \rangle, \langle broker_ECL; broker \rangle, \langle broker_UCL; topLevel \rangle, \langle broker_DCL; topLevel \rangle, \langle user_1_UCL; topLevel \rangle, \langle user_1_DCL; topLevel \rangle, \langle OPC_UCL; broker \rangle, \langle OPC_DCL; broker \rangle, \langle ASP_UCL; broker \rangle, \langle ASP_DCL; broker \rangle \}$;
- $dom' = dom \cup \{ \langle topLevel_ICL; topLevel \rangle, \langle topLevel_ECL; topLevel_control \rangle, \langle broker_ICL; broker \rangle, \langle broker_ECL; broker_control \rangle, \langle broker_UCL; broker \rangle, \langle broker_DCL; topLevel_control \rangle, \langle user_1_UCL; user_1 \rangle, \langle user_1_DCL; topLevel_control \rangle, \langle OPC_UCL; OPC \rangle, \langle OPC_DCL; broker_control \rangle, \langle ASP_UCL; ASP \rangle, \langle ASP_DCL; broker_control \rangle \}$

8.2: Constructs for Control in Compositional Systems

- $cdom' = cdom \cup \{ \langle \text{toplevel_ICL}; \text{toplevel_control} \rangle, \langle \text{toplevel_ECL}; \text{toplevel} \rangle, \langle \text{broker_ICL}; \text{broker_control} \rangle, \langle \text{broker_ECL}; \text{broker} \rangle, \langle \text{broker_UCL}; \text{toplevel_control} \rangle, \langle \text{broker_DCL}; \text{broker} \rangle, \langle \text{user_1_UCL}; \text{toplevel_control} \rangle, \langle \text{user_1_DCL}; \text{user_1} \rangle, \langle \text{OPC_UCL}; \text{broker_control} \rangle, \langle \text{OPC_DCL}; \text{OPC} \rangle, \langle \text{ASP_UCL}; \text{broker_control} \rangle, \langle \text{ASP_DCL}; \text{ASP} \rangle \}$.

The requirements for a structure hierarchy with control can easily be checked:

- $Comp'$ and Lnk' are partitioned in pairwise disjoint sets as required;
- $\langle \text{broker_control}; \text{broker} \rangle \in \prec$, $\langle \text{toplevel_control}; \text{toplevel} \rangle \in \prec$, $\langle \text{broker_ICL}; \text{broker} \rangle \in \prec$, $\langle \text{broker_ECL}; \text{broker} \rangle \in \prec$, $\langle \text{toplevel_ICL}; \text{toplevel} \rangle \in \prec$, and $\langle \text{toplevel_ECL}; \text{toplevel} \rangle \in \prec$, so for each $C \in Contr$, there is a $C' \in Comp$, a $I_1 \in ICLnk$ and a $I_2 \in ECLnk$ such that $C \prec C'$, $I_1 \prec C'$ and $I_2 \prec C'$;
- Components toplevel and broker are the only composed components in sh' . For both, there is exactly one component $C' \in Contr$ such that C' is a subcomponent of the composed component;
- For each component or link $S \in Comp \cup Lnk$, there are upward and downward control links connected to S and the control component $C' \in Contr$ such that $C' \prec P$ for $S \prec P$;
- All links in $DCLnk$, $UCLnk$, $ICLnk$ and $ECLnk$ are connected as requested.

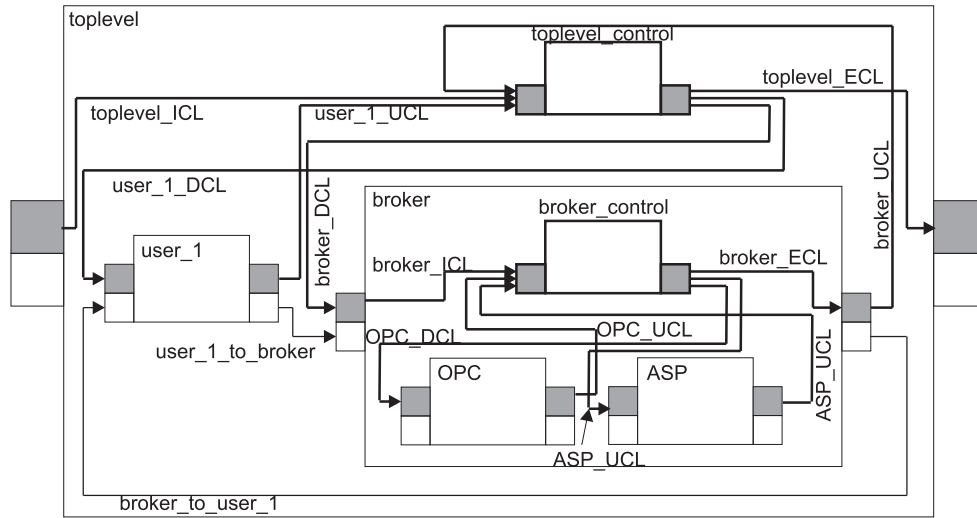


Figure 8.2: Example of a structure hierarchy with control.

The structure hierarchy with control is depicted in Figure 8.2. According to their names, in this example, for the broker agent, there seem to be *two* control components: the dedicated control component broker_control to which all dedicated

control links are connected, and OPC (Own Process Control). However, OPC is not a dedicated control component. Instead, it is a regular component that does not differ from ASP. In the generic agent model presented in Chapter 3, OPC is the component responsible for strategic reasoning, including goal adaptation and priorities. Component OPC executes high-level control which is most often dependent on the domain. The information it processes should therefore *not* be designated as control information. This is illustrated in the example in Section 8.4.■

8.2.5 Channel State and Channel Activation

A control component exercises control over a set of components and links. To exercise control, a control component exchanges information with the controlled components and links using control links. As stated before, control links are, apart from their dedication, normal links. This raises a question: how are control links controlled? The answer consists of a commitment that is specific for control components and links: the state of a control link is fixed and cannot be controlled dynamically. The state of control links is such that they continuously try to transmit control information that appears in the output interface of its source component. This behaviour cannot be influenced by anything in the compositional system. However, if a component wants to delay transmission of control information, it can delay putting this information in its output interface. An advantage of this alternative is that control links are completely independent from the control components to which they are connected.

Another option is to introduce a special kind of link that is used as a control link. This kind of link is special in the sense that it can be activated directly by the control component to which it is connected. This is unlike normal links, which cannot be controlled by the components to which they are connected. (As stated above, normal links are controlled via control links by a control component, similar to components.)

8.2.6 Constraints Imposed by Control

As Chandrasekaran's (1994) abstract characterisation of control indicated, control restricts the future of controlled components and links. The semantic structure therefore requires that a relation between a control component and the components and links it controls is defined that describes in what respect control restricts the future of controlled components and links. However, for reasons explained in Section 8.5, the semantic structure is not committed to a specific relation. Instead, in Section 8.5, a general notion is presented that enables such relations to be defined. Section 8.5 also presents a requirement, the co-ordinated control requirement, that is advised for application of the semantic structure.

8.3 Control in a Multi-Agent System

Section 8.2 presented additional commitments to properties of constructs and relations between constructs provided by the semantic structure that support separate, domain-independent control in a compositional system. The properties to which commitment has been made are encountered in applications of the semantic structure. This section discusses how these properties and relations can be used in the primary application considered in this thesis: modelling multi-agent systems as compositional systems. In Section 8.3.1, the relation between control and autonomy is discussed. Section 8.3.2 presents some perspectives on control in multi-agents system. In Section 8.3.3, modelling choices with respect to control are investigated. In Section 8.3.3, the use of the commitments presented in Section 8.2 for modelling multi-agent systems is discussed.

8.3.1 Control and Autonomy

A multi-agent system consists of a number of agents, each of which is assumed to be autonomous. Even without a philosophically completely satisfactory definition of autonomy, it seems clear that control by one agent over another might conflict with the autonomy of the agents. The central idea put forward in (Castelfranchi, 1995) and (Luck & d’Inverno, 1995) is that to be autonomous, an agent must be able to generate (from its motivations) its own goals, so the agent is not only dependent on the goals of others. This notion of autonomy is called ‘goal autonomy’ in (Castelfranchi, 1995), who also distinguishes a less strict form of autonomy, called ‘executive autonomy’. (For executive autonomy, an agent should be able to accomplish its goals (either generated by itself or given to it by others) free of direct influence of the environment, free to choose among the means needed for the goal.)

Thus, it is impossible for an agent to directly and forcefully change the goal of another agent. However, this does not imply that it is impossible to control an autonomous agent. To control an autonomous agent, instead of forcefully resetting its goals or beliefs (against which it can only complain unsuccessfully), the agent is influenced in such a way that it adopts new goals by itself. This influence can be exercised by communication or by changing the environment of the controlled agent such that this agent changes its beliefs and goals based on new observations. Thus, control over an *autonomous* agent amounts to attempting to influence it such that it changes its state, specifically its goals. This is completely conform the characterisation of control presented at the beginning of Section 8.1. There is no conflict between control as characterised in this chapter and an agent’s autonomy. This is in correspondence with the observation that responses of many living systems are “neither caused by, nor independent of the external stimuli.” (Chomsky, 1988). There must be no external influences that directly dictate an agent’s responses. Under these circumstances, an agent is able to accomplish a goal (either given to it or generated by itself) by the means it chooses.

8.3.2 Some Perspectives on Control in Multi-Agent Systems

Control in a multi-agent system can be related to some other areas:

- In most multi-agent systems, the activities of individual agents in the shared environment introduce interdependencies between the agents and between individual agents and object in the environment. In general, management of these interdependencies by the agents is essential for the agents to achieve their goals. This management of interdependencies, or *co-ordination*, is thus essential in multi-agent systems. Malone and Crowston (1994) study co-ordination from an interdisciplinary point of view, surveying and comparing methods for co-ordination in Computer Science, in biological systems and in Management Science. Malone and Crowston define co-ordination as “management of interdependencies” and argue that co-ordination is a more general notion than co-operation, collaboration and competition. (I.e., co-operation, collaboration and competition are special forms of co-ordination. Co-operation usually is connoted with a shared goal that cannot be achieved individually. Collaboration implies a division of work, and competition usually has the connotation that interdependencies have to be managed in the presence of other agents with conflicting goals.) Control in multi-agent systems as sketched in Section 8.3.1 borders on co-ordination.
- Agents in a multi-agent system can be compared with concurrent objects or actors. Wooldridge (1999) and Briot and Gasser (1998) characterise agents as a special kind of concurrent objects that “decide for themselves whether or not to perform an action on request from another agent” (Wooldridge, 1999, p. 35). This view coincides with the view on autonomy presented in Section 8.3.1. Concurrent, or active, objects, however, do not possess such autonomy. A (concurrent) object makes a set of state operators (methods) available for other objects to invoke. Once a method is made available, the object has no control over invocations of this method: if another object invokes the method, it has to execute that method.
- Contrary to a concurrent object, an agent can decide whether or not to perform an action on request. Consequently, an agent requires a decision procedure that it can use upon receipt of a request to perform an action. In the generic agent model GAM presented in Section 3.2.2, such a decision procedure can be part of the component Own Process Control. A specific decision procedure that has drawn much attention is the BDI (Beliefs, Desires and Intention) architecture developed by Rao and Georgeff (1991). Receipt of a request to perform an action results in a new belief state of the agent. Based on its belief and desires and a theory on the relationships between beliefs, desires and intentions, an agent may adopt an intention to perform the action. From a control point of view, the BDI-approach can be

8.3: Control in a Multi-Agent System

compared with logic programming languages: an agent is equipped with application-specific, static knowledge that drives a general control mechanism specific for the BDI approach. This control mechanism plays the same role as an interpreter for a logic programming language. However, the control mechanism in the BDI approach handles knowledge represented in terms of beliefs, desires and intentions. The control mechanism in logic programming usually handles Horn clauses.

8.3.3 Modelling Choices for Control in Multi-Agent Systems

In addition to modelling choices for interaction presented in Chapter 3, one modelling choice is made for control. Options for this modelling choice are discussed in the next subsection.

8.3.3.1 Modelling Control

The primary application of the semantic structure developed in this thesis is to model multi-agent systems. Chapter 3 presented a guideline for modelling a multi-agent system as a compositional system, which can be represented by the semantic structure. The guideline assumes that a multi-agent system is viewed as a collection of processes, and that specific relations between processes are identified.

In almost every multi-agent system, some processes have to control other processes in the sense of Section 8.1: they have to influence other processes to achieve one of their own goals. As explained in Section 8.1, the essence of exercising control is information transmission. Thus, the control relationship between processes identified in the multi-agent system is modelled by information transmission in the compositional system that represents the multi-agent system. Modelling choices with respect to information processing are presented in Chapter 3.

To introduce the modelling choice presented in this section, the characterisation of control presented in this chapter is summarised. As stated in Section 8.1, control information should be separated, and as domain-independent as possible. Moreover, information transmission for control is special in the sense that it constrains the future of controlled components. As a consequence, only processes that process domain-independent control information should be modelled as control components, and only information exchange between processes in the multi-agent system that constrain the future of other processes in an inescapable way should be modelled as transmissions for control. Thus, for each control relationship between processes in a multi-agent system, it is determined whether the information used for control is domain-independent and whether the controlled processes cannot escape the future envisioned by the controlling process. Information for which this is the case is designated as control information, and the controlling process is modelled as a control component. Together with the three relations between processes presented in Chapter 3, a structure hierarchy is

found that describes the compositional system that represents the multi-agent system. If, during analysis of the multi-agent system, it is decided that specific processes process domain-independent control information and that they determine an inescapable future for the processes they control, the structure hierarchy that represents the multi-agent system should be a structure hierarchy with control as defined in Definition 8.1.

As stated in Chapter 1, the semantic structure developed in this thesis can be used to provide semantics for multi-agent modelling frameworks or specification languages. In such frameworks, the problem of modelling control in multi-agent systems is approached in a generic way. For instance, in the DESIRE modelling framework presented in Chapter 9, the framework provides a fixed control lexicon and fixed control components for each multi-agent system modelled within the framework. The precise behaviour of control components, however, is not fixed. The user of the DESIRE framework may represent control relationships between processes by specifying the behaviour of specific control components. Alternatively, the user may represent control relationships between processes by normal information transmission. The choice between these alternatives depends on the nature of the processes involved.

On the one hand, if the processes involved represent agents, the control representation is best represented as information transmission. As stated in Chapter 1, an agent is considered to be, among others, autonomous. Therefore, there is almost no information that constrains the future of those processes in an inescapable way. (The only exceptions are probably creation and death of agents.) On the other hand, most processes within agents are not autonomous at all: their future can be completely determined by control information. In this case, the best choice is to provide a description of the precise behaviour of a specific control component.

8.4 An Example

In this section, a comprehensive example is presented to illustrate how control in a multi-agent system is represented in a compositional system.

Example 8.4. Chapter 4 presented a scenario in which an information broker agent matches information provided by provider agents with requests received from user agents. In Chapter 4, two internal processes for the provider agents are distinguished: Own Process Control (OPC) and Agent Specific Processes (ASP). However, in Chapter 4, no further information on these processes was provided.

In this chapter, the scenario presented in Chapter 4 is extended to illustrate how control is represented in a separate, domain-independent way as follows. Upon receipt of a request from one of the user agent, the broker agent first determines whether it is willing to process the request. (This decision is for instance made on the basis of a payment scheme: only if the user's balance maintained by the broker is positive, the broker processes the request.) In the generic model presented in

8.4: An Example

Chapter 3, knowledge of whether to process incoming information is often modelled as a subprocess of Own Process Control. The actual processing of the request is considered a subprocess of Agent Specific Processes. (Many different choices are possible. The decision whether to process an incoming request can for instance also be considered a subprocess of Cooperation Management.)

The execution order of Own Process Control and Agent Specific Tasks sketched above is determined during design of the example system. As stated in Section 8.2.1, the question which process is responsible for exercising the control needed to ensure the proper execution order of Own Process Control and Agent Specific Tasks has to be answered. There are two possible alternatives: either OPC and ASP themselves are responsible, or the component of which they are subcomponents (in the example, this is the component called *broker*, which represents the broker agent). In this example, the second alternative is chosen. Thus, *broker* controls its subcomponents to ensure the proper execution order upon receipt of a request from one of the user agents.

As indicated by the description above, control within the broker can be expressed independently from the domain (information brokering): it is sufficient to use terms such as execution of processes and activation of components. Moreover, it is realistic to expect that the subprocesses of the broker agent are not autonomous at all and will comply exactly with control exercised by the broker agent. Consequently, the information needed to control ASP and OPC is designated to be control information. In conformance with the commitment presented in Section 8.2.2, a dedicated control component is added as a subcomponent of *broker*. Moreover, dedicated control links are added to *broker*. To describe the composition structure of *broker* including the added control component and links, a substructure of the structure hierarchy with control presented in Example 8.3 can be used:

$SH_{\text{broker}} = SS(\text{broker}, sh') = \langle \text{Comp}; \text{Lnk}; \prec; \text{dom}; \text{cdom} \rangle$, with:

- $\text{Comp} = \{\text{broker}, \text{OPC}, \text{ASP}, \text{broker_control}\};$
- $\text{Lnk} = \{\text{OPC_UCL}, \text{ASP_UCL}, \text{OPC_DCL}, \text{ASP_DCL}, \text{broker_ICL}, \text{broker_ECL}\};$
- $\prec = \{(\text{OPC}; \text{broker}), (\text{ASP}; \text{broker}), (\text{broker_control}; \text{broker})\} \cup \{I; \text{broker} \mid I \in \text{Lnk}\};$
- $\text{dom} = \{(\text{broker_ICL}; \text{broker}), (\text{broker_ECL}; \text{broker_control}), (\text{OPC_UCL}; \text{OPC}), (\text{OPC_DCL}; \text{broker_control}), (\text{ASP_UCL}; \text{ASP}), (\text{ASP_DCL}; \text{broker_control})\};$
- $\text{cdom} = \{(\text{broker_ICL}; \text{broker_control}), (\text{broker_ECL}; \text{broker}), (\text{OPC_UCL}; \text{broker_control}), (\text{OPC_DCL}; \text{OPC}), (\text{ASP_UCL}; \text{broker_control}), (\text{ASP_DCL}; \text{ASP})\}.$

This structure hierarchy is itself a structure hierarchy with control.

In conformance with the commitment presented in Section 8.2.3, control states are distinguished as a part of the state of *each* component. For controlled components, the control state is part of the input and output substates. The

following sets of control state names are used for broker, OPC and ASP (broker is a controlled component of toplevel, as indicated in Example 8.3):

- $S_{in,csub}$ = {own_state(active),own_state(idle)};
- $S_{out,csub}$ = {own_evaluation(succeeded),own_evaluation(failed)}.

The domain part of the input and output substates of OPC and ASP are not specified in this example. The domain part of the input and output substates of broker is given in Example 5.2.

The state of the control component broker_control only consists of control information. Its set of component states is therefore fully specified as follows:

- $S_{broker_control,out}$ = {component_state(OPC,active),
component_state(ASP,active),
own_state(idle)};
- $S_{broker_control,in}$ = {own_state(active),
evaluation(OPC,succeeded),evaluation(OPC,failed),
evaluation(ASP,succeeded),evaluation(ASP,failed)};

Note that the elements of these sets are names of states that do not have an internal structure. Similar to controlled components, the state of a control component is determined by its information contents and changes as a result of information processing. This is illustrated by the set of local component traces $Beh_{loc}(broker_control)$ for the control component:

- $It_{broker_control,1}$ = own_state(active)| \emptyset | \emptyset →
own_state(active)| \emptyset |component_state(OPC,active) →
evaluation(OPC,succeeded)| \emptyset | \emptyset →
own_state(active)| \emptyset |component_state(ASP,active) →
evaluation(ASP,succeeded)| \emptyset | \emptyset →
 \emptyset | \emptyset |own_state(idle);
- $It_{broker_control,2}$ = own_state(active)| \emptyset | \emptyset →
own_state(active)| \emptyset |component_state(OPC,active) →
evaluation(OPC,failed)| \emptyset | \emptyset →
 \emptyset | \emptyset |own_state(idle).

The two local component traces in $Beh_{loc}(broker_control)$ represent two different situations. The first state of both traces results from information transmission via broker_ICL, which imports control information from the input interface of broker. (It is assumed that receipt of a request from a user agent triggered the occurrence of an input state with own_state(active) as the control part for the input interface of broker.) The output substate of the second state of both traces indicates the envisioned future for OPC: it has to become active. The input substate of the third state of $It_{broker_control,1}$ indicates that feedback from OPC is received. As explained below,

8.4: An Example

evaluation(OPC,succeeded) indicates that OPC has succeeded in establishing that the balance of the user is positive. Consequently, the request received has to be processed by ASP, and therefore, the envisioned future of ASP is that it becomes active, as indicated by the fourth state in $It_{\text{broker_control},1}$. Assuming that there are no other requests to be processed, following receipt of feedback information from ASP, the state of broker_control becomes idle. In the second trace, the user's balance is not positive. The input substate of the third state of $It_{\text{broker_control},2}$ also indicates that feedback from OPC has been received. However, evaluation(OPC,failed) indicates that OPC has not succeeded in establishing that the balance of the user is positive (possibly because the balance is negative or zero, or OPC could not determine the state of the account for another reason). Consequently, the request received is not processed by ASP. Assuming that there are no other requests to be processed, the state of broker_control becomes idle. In both situations, the information that broker_control becomes idle is transmitted to the output interface of broker by the export control link broker_ECL. This information may then serve as feedback information for a component that controls broker.

The control component broker_control receives control information from the components it controls and transmits control info to these controlled components via dedicated control links. Apart from their dedication, control links are normal links. Information link mappings as defined in Definition 5.6 are used to specify which states are linked. According to Definition 5.6, an information link mapping is an octet of states. To keep this example concise, the state of the links is not specified. Consequently, the third, fourth, fifth and sixth states of octets in an information link mapping, which are states of the link itself, need not be taken into account. Moreover, it is assumed that the result of information transmission is not recorded by the domain of the link, and no enabling condition for the co-domain is taken into account. Therefore, the second and seventh state in an information link mapping are not necessary. Thus, the information link mappings are presented as sets of pairs of states: the first state is a state of the domain of the link, and the second state is a state of the co-domain. Such pairs specify that if the first state occurs in a local trace of the domain of the link, then the second state has to occur in a trace of the co-domain of the link. The information link mappings used are:

- $\lambda_{\text{broker_ICL}} = \lambda_{\text{broker_ECL}} = \{ \langle \text{own_state}(\text{active}); \text{own_state}(\text{active}) \rangle, \langle \text{own_state}(\text{idle}); \text{own_state}(\text{idle}) \rangle \};$
- $\lambda_{\text{OPC_DCL}} = \{ \langle \text{component_state}(\text{OPC}, \text{active}); \text{own_state}(\text{active}) \rangle, \langle \text{component_state}(\text{ASP}, \text{active}); \text{own_state}(\text{idle}) \rangle \};$
- $\lambda_{\text{OPC_UCL}} = \{ \langle \text{own_evaluation}(\text{succeeded}); \text{evaluation}(\text{OPC}, \text{succeeded}) \rangle, \langle \text{own_evaluation}(\text{failed}); \text{evaluation}(\text{OPC}, \text{failed}) \rangle \};$
- $\lambda_{\text{ASP_DCL}} = \{ \langle \text{component_state}(\text{ASP}, \text{active}); \text{own_state}(\text{active}) \rangle, \langle \text{component_state}(\text{OPC}, \text{active}); \text{own_state}(\text{idle}) \rangle \};$

- $\lambda_{ASP_UCL} = \{ \langle \text{own_evaluation}(\text{succeeded}); \text{evaluation}(\text{ASP}, \text{succeeded}) \rangle, \langle \text{own_evaluation}(\text{failed}); \text{evaluation}(\text{ASP}, \text{failed}) \rangle \};$

The envisioned future of OPC and ASP as determined by `broker_control` is transmitted to the respective components, where the control substates are affected by the receipt of this information. The commitment presented in Section 8.2.6 requires that a relation between the envisioned future as determined by `broker_control` and the behaviour of the controlled components are specified. This relation is indicated by the set of local component traces $Beh_{loc}(OPC)$ for OPC. In these traces, only the control part of the input and output substates is shown.

- $It_{OPC,1} = \emptyset | \emptyset | \text{own_state}(\text{idle}) \rightarrow \text{own_state}(\text{active}) | \emptyset | \emptyset \rightarrow \dots \rightarrow \text{own_state}(\text{active}) | \text{user_balance_positive} | \emptyset \rightarrow \text{own_state}(\text{idle}) | \emptyset | \text{own_evaluation}(\text{succeeded});$
- $It_{OPC,2} = \emptyset | \emptyset | \text{own_state}(\text{idle}) \rightarrow \text{own_state}(\text{active}) | \emptyset | \emptyset \rightarrow \dots \rightarrow \text{own_state}(\text{active}) | \text{user_balance_not_positive} | \emptyset \rightarrow \text{own_state}(\text{idle}) | \emptyset | \text{own_evaluation}(\text{failed});$

The starting state for OPC is a state in which it is idle. The second state in both traces indicates that OPC receives control information from `broker_control` (via the downward control link `OPC_DCL`, which is added to the structure hierarchy representing the running example system to form a structure hierarchy with control conform Definition 8.2. The downward control link `OPC_DCL` is depicted in Figure 8.2, together with other control links). As a result, OPC processes the request (indicated by the dots). After a while, the assessment of the user's balance is finished. In the first trace, the balance is positive, which results in the occurrence in the output state `own_evaluation(succeeded)`. In the second trace, the balance is not positive, which results in the occurrence in the output state `own_evaluation(failed)`. Both evaluations are transmitted to `broker_control` via the upward control link `OPC_UCL`. The fact that in both traces in $Beh_{loc}(OPC)$, the occurrence of `own_state(active)` (caused by receipt of information via the downward control link `OPC_DCL`) is followed by activity indicates that the information transmitted via the downward control link fully determines the future of OPC. Moreover, the traces also show how domain dependent information (represented by the internal substates `user_balance_positive` and `user_balance_not_positive`) is related to domain-independent control information (`own_evaluation(succeeded)` and `own_evaluation(failed)`).

In this example, for ASP the following local component traces are of interest:

- $It_{ASP,1} = \emptyset | \emptyset | \text{own_state}(\text{idle}) \rightarrow \text{own_state}(\text{active}) | \emptyset | \emptyset;$

8.5: The Relation between Control Components and Controlled Components

- $It_{ASP,2} = \emptyset|\emptyset|own_state(idle)$.

The starting state for ASP is a state in which it is idle. The second state in the first trace indicates that ASP receives control information from broker_control (via the downward control link ASP_DCL). After receipt of this information, ASP starts processing the request. The second trace indicates the situation in which ASP stays idle.

The behaviour of the broker agent can be analysed using the white box view on the behaviour of broker with respect to SH_{broker} . (As all subcomponents of broker are primitive, the white box view equals the glass box view.) The white box view on the behaviour of broker with respect to SH_{broker} is a set of compatible multitraces for SH_{broker} . A multitrace that represents the situation in which the user's balance is positive is as follows:

$$mt_{broker,1} = \{ \langle broker; It_{broker,1} \rangle, \langle broker_control; It_{broker_control,1} \rangle, \langle OPC; It_{OPC,1} \rangle, \langle ASP; It_{ASP,1} \rangle, \\ \langle broker_ICL; \dots \rangle, \langle broker_ECL; \dots \rangle, \langle OPC_UCL; \dots \rangle, \langle OPC_DCL; \dots \rangle, \\ \langle ASP_UCL; \dots \rangle, \langle ASP_DCL; \dots \rangle \}.$$

It is straightforward to check that this multitrace is compatible. A multitrace that represents the situation in which the user's balance is positive is as follows:

$$mt_{broker,2} = \{ \langle broker; It_{broker,1} \rangle, \langle broker_control; It_{broker_control,2} \rangle, \langle OPC; It_{OPC,2} \rangle, \langle ASP; It_{ASP,2} \rangle, \\ \langle broker_ICL; \dots \rangle, \langle broker_ECL; \dots \rangle, \langle OPC_UCL; \dots \rangle, \langle OPC_DCL; \dots \rangle, \\ \langle ASP_UCL; \dots \rangle, \langle ASP_DCL; \dots \rangle \}.$$

In fact, there are only two pairwise compatible combinations of local component traces for the subcomponents of broker: (1) $It_{broker_control,1}$, $It_{OPC,1}$ and $It_{ASP,1}$, and (2) $It_{broker_control,2}$, $It_{OPC,2}$ and $It_{ASP,2}$. The traces $It_{broker_control,1}$ and $It_{OPC,2}$ are not compatible: in $It_{OPC,1}$ a state $own_evaluation(succeeded)$ occurs, but in $It_{broker_control,1}$ a state $evaluation(OPC,succeeded)$ does not occur, which violates λ_{OPC_UCL} .

8.5 The Relation between Control Components and Controlled Components

Section 8.1 states that information transmission for control is special in the sense that it constrains the future of the component to which control information is transmitted: this component cannot escape the future specified by the control information. Section 8.5.1 explains which commitment of the semantic structure supports this property of information transmission for control. Section 8.5.2 presents some relations between control components and controlled components associated with these commitments.

8.5.1 Constraints Imposed by Control Information

The commitment of the semantic structure with respect to information transmission for control is that the model of the behaviour of a controlled component maintained by a control component must match the actual behaviour of the controlled component. As described in Section 8.1, according to Chandrasekaran (1994), each control processes can be described as a process in which a model of the past and current behaviour of the controlled component is made, after which this model is extended to include an envisioned future for the controlled component. (As the semantic structure abstracts from the structure of information processed in components, the model maintained by a control component cannot be specified. However, similar to all information maintained by a component, such a model determines a part of the state of the control component. Note that the model maintained by a control component is *not* a construct provided by the semantic structure.)

Although there is a commitment to the existence of a relation between the model maintained by a control component and the controlled components, no commitment to a specific relation is made, for the following reasons:

- The precise definition of such a relation depends on how a control component models its controlled component. However, this differs for different applications of the semantic structure. Thus, as the semantic structure abstracts from the structure of the information maintained within a component, a precise definition fixed for the semantic structure of such a relation cannot be given;
- The extent to which a control component is able to determine the future of a controlled component varies from application to application. A commitment to a specific commitment would probably be too restrictive for a number of applications, and not restrictive enough for other applications. If the commitment is too restrictive for an application, almost no information transmission can be designated as control information. If the commitment is not restrictive enough, almost all information transmission can be designated as control information. In both cases, control cannot be separated to such an extent that it leads to a better, reusable structure of the model.
- The precise definition of such a relation determines, to a large extent, a number of other dynamic properties of a compositional system. Applications of the semantic structure differ with respect to these properties. Consequently, if one specific relation is defined for the semantic structure, this relation is most likely too weak for some applications, and too restrictive for others. The way in which such relations affect dynamic properties is discussed in Section 8.5.1. For the application of the semantic structure to DESIRE presented in Chapter 9, a precise definition is given.

8.5: The Relation between Control Components and Controlled Components

Thus, in this chapter, no commitment to a specific relation is made. However, the problem of how to formalise such a relation is addressed in general.

As stated earlier in this chapter, the essence of control is information transmission, and therefore, the constructs presented in Chapter 5 and Chapter 6 suffice for control in the semantic structure. In fact, the constructs presented in Chapter 5 and Chapter 6 suffice to express the relation between a control component and the component it controls in a basic form, using the sets $Beh_{loc}(S)$ for controlled components and links S and information link mappings as follows.

As stated in Section 8.2, the state of controlled components and links consist of control and domain parts. Information in the control parts is exchanged via control links with the control component that controls the controlled components. The state of control components only have a control part. Information link mappings can be used to specify which control information produced by the control component is to be transmitted to which controlled component or link. This control information is meant to influence the behaviour of the controlled component. For example, assume that a state called $component_state(A, active)$ of the control component is linked to a state with a control part called $own_state(active)$ of a controlled component A . Compatibility ensures that if the state $component_state(A, active)$ occurs in a local component trace of the control component, then a state with a control part $own_state(active)$ occurs in the local component trace for A in a compatible multitrace. As the names of these states suggest, A should become active, which, in this example means that the next state of A contains specific domain information, e.g., the next state should be the state called $results_available$.

Control information is special in the sense that applications of the semantic framework have to ensure that the future of a controlled component as envisioned by a control component is inescapable. In the example sketched in the previous paragraph, this can be accomplished by restricting the set $Beh_{loc}(A)$ to traces in which each state with control part $own_state(active)$ is followed by state $results_available$. This requirement is a local requirement: it only restricts the set of local component traces of A . (Example 8.4 in Section 8.4 specified the relation between the envisioned future of OPC as determined by $broker_control$ and the local component traces of OPC in a similar way.)

Thus, the relation between a control component and the components it controls can be expressed using the constructs presented in Chapter 5 and Chapter 6: specific states of the control component are linked to specific states of the controlled components and links, and requirements on the sets of local component traces of the controlled components and links further determine how the behaviour of the controlled components is related to behaviour of the control component. However, although in this way, the future of a controlled component is related to the behaviour of the control component, the key point is that it is not clear when this future begins from the point of view of the control component. When the future begins depends on the properties of information transmission (if information transmission is not lossless, it may not begin at all). For instance, in

8.5: The Relation between Control Components and Controlled Components

Example 8.4, it is not clear how the states of `broker_control` relate to the states of OPC and ASP in time. In local component trace $It_{\text{broker_control},1}$, the state in which feedback information from OPC is received (third state) immediately follows the state in which OPC is made active (second state). However, as trace $It_{\text{OPC},1}$ indicates, there are several local states of OPC between the receipt of the information that OPC has to become active and the state `own_evaluation(succeeded)`.

For some applications of the semantic structure, more detail than provided by the constructs presented in Chapter 5 and Chapter 6 is not needed. However, for many applications more detail is needed for the following reason. Evaluation of the design of a multi-agent system includes analysis of dynamic properties of the system. In general, the three views on the behaviour of a compositional system can be used to assess dynamic properties by comparing occurrences of state transitions in the individual traces within compatible multitraces. For sets of components and links that consist of a dedicated control component and the components and links it controls, a second alternative is available. As stated in Section 8.2.6, the control component determines the behaviour of the components and links it controls, to a large extent. Therefore, it is possible to assess properties of the controlled components and links by evaluating the behaviour of the control component.

As global time is not assumed to be available, it is not possible to directly express relations between time points in different traces using only the constructs presented in Chapter 5 and Chapter 6. However, with the help of global states as defined in Chapter 7, the relation between a control component and the components and links it controls can be specified more precisely.

As an example of the use of global states, it is possible to request that the envisioned future of controlled components and links in a control component occurs in the same global states as the actual future of the controlled components and links. The envisioned future of controlled components and links is represented by specific states of the control components. These states are linked to states of the controlled components and links by downward task control links, as specified by the information link mappings for these links. The suggested requirement can thus be stated more precisely as follows: given a structure hierarchy with control and a compatible multitrace for this structure hierarchy, for each global state σ of the multitrace, if the output part of a control component C in the global state occurs as the first state in a tuple in the information link mapping of a downward control link from C , then in the same global state, the input part for the co-domain of this link must be the eighth state in that tuple. Formally: let $SH = \langle \text{Comp} \cup \text{Contr}; \text{Lnk} \cup \text{UCLnk} \cup \text{DCLnk}; \prec; \text{dom}; \text{cdom} \rangle$ be a structure hierarchy with control, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ . Then:

$$\forall \sigma \in \text{SGS}(SH, \mu, \gamma), \forall C \in \text{Contr}, \forall l \in \text{DCLnk} \text{ and } \forall v \in \mathcal{S}_C: \\ \text{if } \text{dom}(l) = C \text{ and } \sigma(C) = v \text{ and}$$

8.5: The Relation between Control Components and Controlled Components

$\langle\langle\text{out}(v); \text{out}(v_{C,j}); \langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l} \rangle; \langle \text{in}(v_{\text{cdom}(I),i}); \text{in}(v_{\text{cdom}(I),j}) \rangle \rangle \rangle \in \lambda_I$
then $\sigma(\text{cdom}(I)) = v_{\text{cdom}(I),j}$ for any j, i'', j'', k, l, i' and j' .

This requirement, however, is too strict: there are no global states for which this requirement holds. (A global state for which this requirement holds cannot be a global state for the following reason. On the one hand, for each global state σ and for each component C and link I , if $\langle\langle\text{out}(v); \text{out}(v_{C,j}); \langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l} \rangle; \langle \text{in}(v_{\text{cdom}(I),i}); \text{in}(v_{\text{cdom}(I),j}) \rangle \rangle \rangle \in \lambda_I$, with $\sigma(C) = v$ then $\sigma(C) \rightarrow_{sd} \sigma(\text{cdom}(I))$ by the second clause of the definition of strict dependence. On the other hand, by the definition of a global state σ , for each pair of components C and D , $\sigma(C) \rightarrow_{sd} \sigma(D)$, and therefore, for each global state σ and for each component C and link I , $\sigma(C) \rightarrow_{sd} \sigma(\text{cdom}(I))$.)

As an alternative, the future of controlled component A may be required to begin as soon as possible. More precisely: each global state in which the local state of the control component is `component_state(A,active)` must be followed by a global state in which the local state of A is `results_available`. Formally: let $SH = \langle \text{Comp} \cup \text{Contr}; \text{Lnk} \cup \text{UCLnk} \cup \text{DCLnk}; \prec; \text{dom}; \text{cdom} \rangle$ be a structure hierarchy with control, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ . Then:

$\forall \sigma \in \text{SGS}(SH, \mu, \gamma), \forall C \in \text{Contr}, \forall I \in \text{DCLnk}$ and $\forall v \in \mathcal{S}_C$:
if $\text{dom}(I) = C$ and $\sigma(C) = v$ and
 $\langle\langle\text{out}(v); \text{out}(v_{C,j}); \langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l} \rangle; \langle \text{in}(v_{\text{cdom}(I),i}); \text{in}(v_{\text{cdom}(I),j}) \rangle \rangle \rangle \in \lambda_I$
then there is a strict global state $\sigma' \in \text{next}_{\text{SGS}}(SH, \gamma, \mu)(\sigma)$ such that
 $\sigma'(\text{cdom}(I)) = v_{\text{cdom}(I),j}$ for any j, i'', j'', k, l, i' and j' .

Contrary to the requirement presented previously, this requirement can be met. However, two issues remain. First, this requirement is not very strong, as it does not constrain the global states between the global state with substate `component_state(A,active)` for the control component the global state with substate `results_available` for the controlled component. The second issue is related to the fact that the order of global states is a partial order. This issue is discussed with the help of some additional notions, which are presented in the rest of this chapter. Chapter 9 presents an application of the semantic structure in which the principles presented in this chapter are applied.

8.5.2 Common Global States

Chapter 7 defined a notion of global state relative to a multitrace for a structure hierarchy SH . As explained in Chapter 7, global states are partially ordered, even if all local component and link traces in a multitrace are linear. The partiality of the global state order represents the fact that different observers may observe different behaviour of the compositional multitrace represented by SH , due to the absence of

global time or fixed duration of information transmission. Each observation corresponds with a different path through the partial order of states.

Most global states for a specific multitrace are only observed by a subset of all possible observers. However, some global states occur on each path in the partial order. In other words, such a state is observed by each possible observer. To formally define common global states, first the notion of an observation is formally defined as a linear extension of the partial order of global states.

Definition 8.5. (Observation). *Let SH be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ . An observation of μ is a sequence of strict global states $\sigma_1, \sigma_2, \dots \in SGS(SH, \mu, \gamma)$ such that:*

- *if $\sigma_i \in \text{next}_{SGS(SH, \mu, \gamma)}(\sigma_j)$, then $i > j$, and*
- *σ_{i+1} is an immediate successor of σ_i , i.e, there is no $\sigma' \in SGS(SH, \mu, \gamma)$ such that $\sigma' \in \text{next}_{SGS(SH, \mu, \gamma)}(\sigma_i)$ and $\sigma_{i+1} \in \text{next}_{SGS(SH, \mu, \gamma)}(\sigma')$.*

The set of all observations of μ is denoted $OBS(SH, \mu, \gamma)$.

The definition of a common global state is now straightforward:

Definition 8.6. (Common global state). *Let SH be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for c . A common global state is a strict global state $\sigma \in SGS(SH, \mu, \gamma)$ such that for all sequences $(\sigma_i)_{i \in \mathbb{N}}$ in $OBS(SH, \mu, \gamma)$, there is an i such that $\sigma = \sigma_i$. The set of all common global states for a structure hierarchy SH , a multitrace μ , and a collection of compatibility relations γ is denoted $CSGS(SH, \mu, \gamma)$.*

In (Fromentin & Raynal, 1995), a necessary and sufficient criterion for global states to be common is presented. In terms of the semantic structure developed in this thesis, this criterion is as follows:

Proposition 8.7. *Let $SH = \langle \text{Comp}; \text{Lnk}; \prec; \text{dom}; \text{cdom} \rangle$ be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ . A global state $\sigma \in SGS(SH, \mu, \gamma)$ is a common global state iff:*

$$\forall S_1, S_2 \in \text{Comp} \cup \text{Lnk}: \text{prev}_{\mu(S_2)}(\sigma(S_1)) \rightarrow_{sd} \text{next}_{\mu(S_2)}(\sigma(S_2)). \quad (1)$$

The formal proof of this proposition can be found in (Fromentin & Raynal, 1995). Informally, the proposition is most easily understood for the case where $\text{Comp} \cup \text{Lnk}$ has precisely two elements, say A and B . Assume that (1) holds, i.e., for a specific strict global state σ , $\text{prev}_{\mu(A)}(\sigma(A)) \rightarrow_{sd} \text{next}_{\mu(B)}(\sigma(B))$ and $\text{prev}_{\mu(B)}(\sigma(B)) \rightarrow_{sd} \text{next}_{\mu(A)}(\sigma(A))$ (note that these two dependencies are symmetric). This situation is depicted in Figure 8.3. The configuration of symmetrically crossing dependencies between state transitions (the thick arrows) prevents specific global states from occurring. For instance, global state σ' cannot occur as for σ' , the transition from $\sigma(A)$ to $\text{next}_{\mu(A)}(\sigma(A))$ occurred, while the transition from

8.5: The Relation between Control Components and Controlled Components

$prev_{\mu(B)}(\sigma(B))$ to $\sigma(B)$ did not occur. This is prohibited because the transition from $prev_{\mu(B)}(\sigma(B))$ to $\sigma(B)$ precedes the transition from $\sigma(A)$ to $next_{\mu(A)}(\sigma(A))$. Likewise, σ'' cannot occur. Starting from σ_0 , in all sequences of global states, σ has to occur before any global state containing $next_{\mu(A)}(\sigma(A))$ or $next_{\mu(B)}(\sigma(B))$ can occur. Therefore, σ is a common global state.

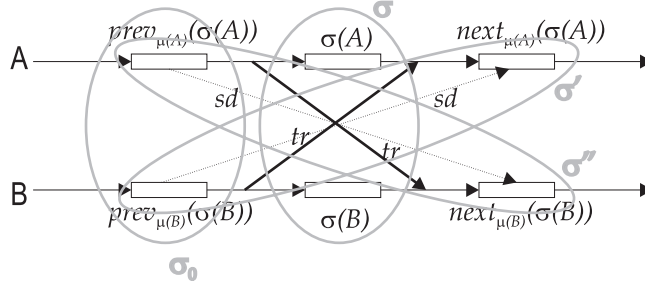


Figure 8.3: Illustration of Proposition 8.7.

Given a partial order of global states $\langle SGS(SH, \mu, \gamma); next_{SGS(SH, \mu, \gamma)} \rangle$, the restriction of this partial order to the subset of common global states is a total order. (If it were not, then some of the common global states would not be in all observations.) The total order of common global states can be used to formulate requirements on the relation between a control component and the components it controls. In the rest of this section, first such a requirement is formally defined. This requirement is then discussed.

8.5.2.1 Co-ordinated Control Requirement

The co-ordinated control requirement expresses a relation between a control component and the components it controls. According to this requirement, control is co-ordinated if control information made available in a specific common global state, is received in the immediate successor common global state of this specific state. (The immediate successor relation is the restriction of the partial order on global states to common global states. As this restriction is a total order, it is possible to require that control information be received in *the* immediate successor common global state.) The co-ordinated control requirement is defined formally as follows:

Definition 8.8. (Co-ordinated Control Requirement). *Let $SH = \langle Comp \cup Contr; Lnk \cup UCLnk \cup DCLnk; \prec; dom; cdom \rangle$ be a structure hierarchy with control, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ . Control in the compositional system represented by SH is co-ordinated if:*

8.5: The Relation between Control Components and Controlled Components

- $\forall \sigma \in \text{CSGS}(SH, \mu, \gamma), \forall C \in \text{Contr}, \forall I \in \text{DCLnk}$: if $\text{dom}(I) = C$ and $\langle \langle \text{out}(\sigma(C)); \text{out}(v_{C,j}) \rangle; \langle v_{L,i}; v_{L,j}; v_{L,k}; v_{L,l} \rangle; \langle \text{in}(v_{\text{cdom}(I),i}); \text{in}(v_{\text{cdom}(I),j}) \rangle \rangle \in \lambda_I$, then there is a common global state $\sigma' \in \text{next}_{\text{SGS}}(SH, \gamma, \mu)(\sigma)$ such that:
 - $\sigma'(\text{cdom}(I)) = v_{\text{cdom}(I),j'}$ for any j, i, j', k, l, i' and j' ;
 - there is no σ'' such that $\sigma'' \in \text{next}_{\text{SGS}}(SH, \gamma, \mu)(\sigma)$ and $\sigma' \in \text{next}_{\text{SGS}}(SH, \gamma, \mu)(\sigma'')$.
- and $\forall \sigma \in \text{CSGS}(SH, \mu, \gamma), \forall C \in \text{Contr}, \forall I \in \text{UCLnk}$: if $\text{cdom}(I) = C$ and $\langle \langle \text{out}(\sigma(\text{dom}(I))); \text{out}(v_{\text{dom}(I),j}) \rangle; \langle v_{L,i}; v_{L,j}; v_{L,k}; v_{L,l} \rangle; \langle \text{in}(v_{C,i}); \text{in}(v_{C,j}) \rangle \rangle \in \lambda_I$, then there is a common global state $\sigma' \in \text{next}_{\text{SGS}}(SH, \gamma, \mu)(\sigma)$ such that:
 - $\sigma'(\text{cdom}(I)) = v_{C,j'}$ for any j, i, j', k, l, i' and j' ;
 - there is no σ'' such that $\sigma'' \in \text{next}_{\text{SGS}}(SH, \gamma, \mu)(\sigma)$ and $\sigma' \in \text{next}_{\text{SGS}}(SH, \gamma, \mu)(\sigma'')$.

The first part of this requirement resembles the second requirement put forward in the previous subsection. There are two differences. First, the co-ordinated control requirement only formulates a requirement on those global states that are common. (The first universal quantifier ranges over the set of common global states.) Second, the result of control information transmission must be present in the *first* common global state that follows the common global state in which it was available in the control component. The second part of the co-ordinated control requirement co-ordinates feedback of control information from controlled components and links to the control component. To keep the example requirements in Section 8.5.1 concise, co-ordination of feedback information is not included in these requirements.

Common global states and the co-ordinated control requirement thus only involve a subset of all global states associated with a multitrace. It is indeed possible to formally introduce a restriction of the partial order of global states as an abstract perspective on a multitrace, in which only specific global states are represented. This can be appreciated as follows. Suppose that a compositional system is represented by a structure hierarchy with control SH and that for multitraces for SH , the co-ordinated control requirement must hold. At an early stage in the design, it is sufficient to specify that this requirement must hold. However, at a later stage, mechanisms must be incorporated in the design to ensure that the requirements hold. As an example, exchange of extra information may be needed to ensure that control information is received at the first common global state that follows a common global state in which the control information is made available. Figure 8.4 below depicts (part of) a partial order of eight global states (represented by ovals). Two of the global states (depicted in black) are common global states. The black arrows indicate the total order of these common global states. The small circles inside the global states depict the local component states of a control component. The leftmost common global state is a state in which control is co-ordinated. In the local component state of the control component, new control information is made available. This control information is transmitted to a number of controlled components. This information is not received by all

8.5: The Relation between Control Components and Controlled Components

controlled components at the same time, and moreover, the controlled components possibly exchange information with each other as well. The resulting behaviour is depicted by the grey states in Figure 8.4. The behaviour of the individual components (as represented by sets of local component traces $Beh_{loc}(S)$), however, is designed such that eventually there is a new common global state in which the control information is available to all controlled components. In this new common global state, control is co-ordinated.

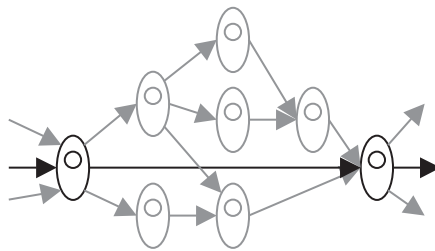


Figure 8.4: Common global states.

8.5.2.2 Some perspectives on the co-ordinated control requirement

Some advantages and disadvantages of the co-ordinated control requirement can be distinguished.

- The set of common global states is totally ordered. If the local component traces used to describe the behaviour of the control component are linear, the 'actual behaviour' of the controlled components as described by the linear order of common global states has the same structure as the behaviour of the control component. Thus, the envisioned future behaviour of controlled components as represented in a control component can be compared to their actual behaviour. (E.g., by viewing the combination of the local component state of a control component and of a controlled component in the same common global state as a bisimulation relation.)
- As stated before, the partial order of global states reflects different observations of the behaviour of a compositional system. Consequently, some properties of the behaviour of a compositional system, such as safety properties, hold for all observations, while other properties do not hold for all observations. However, under specific circumstances, the latter case suffices, as is explained in (Katz & Peled, 1990).
- As Proposition 8.7 indicates, a global state is a common global state iff there is a mutual dependency between local states of each pair of components and links in a compositional system. A mutual dependency between states in essence means that the possible next local state transitions are mutually

8.5: The Relation between Control Components and Controlled Components

constrained. Components in a common global state are thus synchronised: from a global time point of view, they are all in this state at the same time. Consequently, if common global states are required to exist, as is the case for the co-ordinated control requirement, controlled components are synchronised with their control component and with one another. Thus, the co-ordinated control requirement influences to a large extent the possible behaviour of controlled components. In an implementation, common global states are a source of concurrency bottlenecks. A possible solution is to apply the co-ordinated control requirement to a subset of all controlled components in a structure hierarchy. Fromentin and Raynal (1995) offer a similar solution, which they call *partial common global states*.

The paper by Fromentin and Raynal (1995) from which Proposition 8.7 was taken, introduces abstraction of specific states in a different way. The starting point for Fromentin and Raynal is a partial order on a set of events, which is assumed to be given. The set of events is partitioned in 'primitive level' events and 'user level' events. A primitive level event is not a 'sub-event' of a user level event. (This would not be possible, as events are atomic and do not have any internal structure.) The partial order of events is then restricted to the set of user level events. The resulting partial order on (user level) events is then used to define global states in a similar way as presented in Chapter 7 and common global states as defined in Section 8.5.2. The restriction of the partial order on global states to common global states presented in Section 8.5.2.1 is not used by Fromentin and Raynal (1995).

As an aside, in some approaches, events are not atomic. A hierarchical structure of events is used to model levels of event abstraction. Lamport (1986) introduces hierarchies of event abstraction to represent levels of detail of the behaviour of a system. Kshemkalyani (1998) describes all possible precedence relations between events that may have sub-events.

In the semantic structure presented in this thesis, control restricts the future of controlled components and links. The co-ordinated control requirement presented in Section 8.5.2.1 provides a precise expression of the restriction relation between a control component and the components and links it controls. It is advised for applications of the semantic structure to adopt the co-ordinated control requirement. However, applications are free to adopt alternative requirements to express how control restricts the future of controlled components and links.

8.5: The Relation between Control Components and Controlled Components