INSTITUT FÜR INFORMATIK UND PRAKTISCHE MATHEMATIK

LEHRSTUHL FÜR SOFTWARETECHNOLOGIE

Composition and Verification of Condition/Event Systems

Ben Lukoschus

Bericht Nr. TR-ST-99-1

May 1999

CHRISTIAN-ALBRECHTS-UNIVERSITÄT ZU KIEL

Composition and Verification of Condition/Event Systems

Ben Lukoschus

(bls@informatik.uni-kiel.de)

Chair of Software Technology
Institute of Computer Science and Applied Mathematics
Christian Albrechts University of Kiel

May 1999

Abstract

We consider different ways of combining discrete and timed condition/event systems in a modular, compositional way. In addition to the interconnection operators for discrete condition/event systems found in the literature, we introduce a new, powerful operator, called *parallel interconnection* operator.

We introduce condition/event systems (CESs) as a model for continuous-time discrete event systems which communicate by exchanging two kinds of symbols, called *condition* and *event* symbols. Condition symbols are used to describe system states and can enable or disable state changes, whereas event symbols denote instantaneous actions and are used to trigger state changes.

A special class of CESs are *discrete condition/event systems* (DCESs), which use a finite transition system to control the input/output behavior of the system. CESs and DCESs can be represented as block diagrams using signal flows for the connection between the system. These models are widely used in system engineering and control theory.

We investigate the operators for connecting DCESs given in the literature, namely the *cascade interconnection* and the *feedback connection* operators. These operators are very restrictive in the way systems can be connected, and therefore, their usage is complicated and not very intuitive. We introduce the *loop interconnection* operator, which covers any connection between two systems, but this operator does not reach maximal flexibility for connections among more than two systems.

Thus, we define the *parallel interconnection* operator, which allows arbitrary connections among a set of DCESs. This operator allows easy and intuitive handling of interconnected systems and does not have the strong restrictions of the previously defined operators.

Quantitative timing, an important feature DCESs are not capable of, is added by introducing *timed condition/event systems* (TCESs), which are DCESs augmented with *timers.* We extend our parallel interconnection operator to TCESs, completing our goal of defining a flexible operator for composing DCESs and TCESs connected in an arbitrary way.

An extensive example shows various possibilities how to verify properties of a system consisting of some interconnected TCESs. Obviously, compositional techniques are most promising for an efficient verification.

We conclude this work by showing how a set of interconnected DCESs can be transformed into the input language of the model checking tool SMV. This offers another possibility to apply formal verification to condition/event systems.

Contents

1 Introduction **9**

2 Condition/Event Systems **17**
2.1 Basic Notions 17
2.2 Condition/Event Systems 19
2.3 Discrete Condition/Event Systems 20

3 Connecting Condition/Event Systems **25**
3.1 The Cascade Interconnection 26
3.1.1 The Cascade Interconnection of C/E Systems 26
3.1.2 The Cascade Interconnection of Discrete C/E Systems 27
3.2 The Feedback Connection 30
3.2.1 The Feedback Connection of C/E Systems 30
3.2.2 The Feedback Connection of Discrete C/E Systems . . 31
3.3 The Loop Interconnection 35
3.3.1 The Loop Interconnection of C/E Systems 36
3.3.2 The Loop Interconnection of Discrete C/E Systems . . 37
3.3.3 Properties of the Discrete Loop Interconnection 41
3.3.4 Special Cases of the Discrete Loop Interconnection . . 43

4 The Parallel Interconnection **49**
4.1 Adding Component Names to C/E Systems 50
4.1.1 Named C/E Systems 51
4.1.2 Named Discrete C/E Systems 52
4.2 The Parallel Interconnection of C/E Systems 53
4.3 The Parallel Interconnection of Discrete C/E Systems 56
4.4 Hiding Output Components in CESs and DCESs 61
4.5 Special Cases of the Discrete Parallel Interconnection 65
4.6 Weakening the Unique Solution Property 69

5 Introducing Quantitative Timing **81**
5.1 Timers 81
5.2 Timed Condition/Event Systems 84
5.3 The Parallel Interconnection of Timed C/E Systems 88

6 Example: The Evaporator System **93**
6.1 The Problem 93
6.2 The Model 95
6.2.1 The Modules 97
6.3 Verification 107
6.3.1 Using Timing Diagrams 107
6.3.2 Using Timed Automata 110
6.3.3 Using the Parallel Interconnection Operator 110

7 Verification of DCESs Using SMV **115**
7.1 Transforming a system of DCESs into the SMV framework . . 115
7.1.1 Conversion of Identifiers 116
7.1.2 Translating a DCES into an SMV module 116
7.1.3 Combining the SMV modules 118
7.1.4 Well-definedness 121
7.1.5 A Complete Example 125
7.2 Verification 126

8 Conclusion **129**

Bibliography **131**

List of Figures

1.1 Example of an algebraic loop. 11

2.1 Example of a condition signal. 18
2.2 Example of an event signal. 19
2.3 A block diagram of a condition/event system. 20

3.1 CES cascade interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$. 26
3.2 The feedback connection in [SK91]. 30
3.3 CES feedback connection of $\mathcal{S}$. 31
3.4 CES loop interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$. 36
3.5 Modeling feedback of $\mathcal{D}$ by loop using the system $\mathcal{D}_{\mathsf{id}}$. 46

4.1 Example of a parallel interconnection. 54
4.2 The composition of the system in Figure 4.1. 54
4.3 Mutually connected DCESs. 57
4.4 Modeling feedback by parallel using $\mathcal{D}_{\mathsf{id}}$. 69
4.5 A system with two solutions for the feedback value. 70
4.6 A parallel interconnection of DCESs with multiple solutions. . 78
4.7 Block diagram of $\mathcal{D} = \mathcal{D}_1 \,|||\, \mathcal{D}_2$. 79

5.1 Example of a timer function. 82
5.2 Block diagram of a timed condition/event system. 85

6.1 Diagram of the evaporator system. 94
6.2 Block diagram of the evaporator system model. 96
6.3 Block diagram of the TCES *Tank-1*. 98
6.4 Discrete part of the TCES *Tank-1*. 99
6.5 Block diagram of the TCES *Tank-2*. 100
6.6 Discrete part of the TCES *Tank-2*. 101
6.7 Block diagram of the TCES *Heater*. 101
6.8 Discrete part of the TCES *Heater*. 102
6.9 Block diagram of the DCES *Condenser*. 102
6.10 Discrete part of the DCES *Condenser*. 103

6.11 Block diagram of the TCES *Pressure*. 103
6.12 Discrete part of the TCES *Pressure*. 103
6.13 Block diagram of the TCES *Controller*. 104
6.14 Discrete part of the TCES *Controller*. 105
6.15 A timing diagram for the evaporator system. 108
6.16 Block diagram of the TCES *PressCond*. 112
6.17 Discrete part of the TCES *PressCond*. 113
6.18 The reachable discrete part of the TCES *PressCond*. 113

7.1 A DCES transition and its SMV representation. 119
7.2 Block diagram of *Controller* and *Tank-2*. 122
7.3 The SMV module `main`. 122
7.4 The SMV module `Controller`. 123
7.5 The SMV module `Tank2`. 124
7.6 Block diagram of the DCES *Switch*. 126
7.7 The transition system of the DCES *Switch*. 126
7.8 The SMV code for the DCES *Switch*. 127

Chapter 1

Introduction

When building a computer controlled system, one has to ensure that it works properly, that is, according to its specifications. Often it is considered that "good faith" in programmers, system builders, and advanced technology is enough to create error free systems. And often enough, this short-minded (or short-financed) attitude causes loss of money, time, or even lives, e.g., [Lev95].

In the design of critical systems such as nuclear power plants, planes, medical devices, or small robots moving on the surface of Mars, more than just "good faith" is needed to ensure compliance with the specifications. The first step into that direction is a *formal framework* where the system design and the system requirements can be embedded.

In this formal framework, large systems should be dividable into smaller ones, called *modules,* which are connected in some way to exchange information. This is called *modularity.* Among the advantages of modularity are the reusability of modules, the ability to split up work during system design, and compositional reasoning [HdR86, HdR90] in verification, model checking, and testing. A common method for representing modular structures in system engineering and control theory is the use of *block diagrams* and *signal flows.*

The framework used in this work is the model of *condition/event systems* (CESs), which are introduced in [SK91]. CESs are a class of continuous-time discrete event systems and can be represented graphically as block diagrams. Communication between CESs takes place by exchanging two different kinds of symbols, called *condition* and *event* symbols. A set of condition or event symbols is called condition or event *alphabet.* Condition symbols are used to describe system states and to enable or disable state changes, whereas event symbols denote certain actions at discrete points in time and can be used to trigger state changes. To describe the semantics of a CES, condition and event *signals* are used. A signal is a function which maps points in time to a condition or event alphabet, i.e., at each moment, one symbol of an alphabet is visible. The behavior of a CES is given by a function which associates with each pair of an input condition signal and an input event signal a nonempty set of possible outputs, which are pairs of output condition signals and output event signals.

This way of describing a system's behavior has some disadvantages. The behavior of a CES is given by relating signals, and one signal describes the current symbol visible at a CES's input or output component for *all* points in time. This is a very abstract view, since normally a dynamic system's output symbols at a certain moment depend only on its internal state and its input symbols at that very moment. Furthermore, we would like to have an *operational* description of the system using functions mapping input symbols to output symbols (and not relating signals like in CESs), since this makes the implementation and the reasoning about systems much easier. Therefore, we introduce *discrete condition/event systems* (DCESs), which communicate using the same symbols as CESs, but are described operationally using a finite set of internal states, a transition function, and output functions which compute output symbols using the current internal state and the current input symbols. So a DCES can be understood as a finite automaton having two output functions, one for conditions and one for events. The semantics of a DCES is described as a CES, but not every behavior of a CES is expressible using a DCES.

In the step from CESs to DCESs, however, we lose *quantitative timing,* i.e., in the DCES model, we have no means of measuring time. Since many critical systems are based on strong timing requirements, we introduce *timers.* A timer has an internal clock which increases linearly with the global time and can be reset by an external event. Semantically, timers are modeled as a class of CESs. Using these timers, *timed condition/event systems* (TCESs) are defined in [Kow95] in the spirit of timed automata [AD94]. A TCES is simply a DCES which is connected to a set of timers. With each timer we associate a list of *thresholds*, which are integers the timer value can be compared to. In addition to the reset event signal received from the DCES, each timer has a condition signal denoting in which interval between the thresholds the current timer value is. Furthermore, timers use an event signal to notify the DCES whenever the timer value equals exactly one of the thresholds.

For compositional reasoning about condition/event systems, the notions of *cascade* and *feedback* connections to compose CESs and DCESs are presented in [SK91]. These connections are very restrictive in the sense that they do not provide many ways of connecting systems, e.g., feedback allows no additional external input signals. Moreover, there is no formal framework for composing TCESs.

In this work we present a formal framework for composing DCESs and TCESs. Intuitively, systems are composed by connecting output components to input components, and the signals of the connected components have to be equal. We discuss the following two questions which arise when constructing a system consisting of some interconnected modules (DCESs in our framework):

1. Can we express the semantics of the system by using the semantics of its modules?

2. Can we compose the modules into one DCES having the same semantics as the compound system?

For answering these questions, we must of course have a formal definition of the semantics of the compound system. This must not necessarily be expressible in the way we express the semantics of the parts.

The answers to both questions depend on the way the systems are connected. If there are no interconnections at all, i.e., the parts do not exchange any information and are independent from each other, both answers are obviously yes. We simply can use Cartesian products of the parts' signals to form the signals of the compound system (this answers question 1), and we can use a Cartesian product of the parts' sets of states to gain the set of states for the DCES needed in question 2. No synchronization is needed, since the parts have no common signals.

If there are interconnections of the parts, which is usually the case for non-trivial systems, the answers are not that easy to find. Although question 1 might be positively answered, a DCES for question 2 may not exist due to so-called *algebraic loops*. For example, consider the two DCESs $\mathcal{D}_1$ and $\mathcal{D}_2$ in Figure 1.1. Both systems are mutually connected by condition signals that can take the values "a" or "b". $\mathcal{D}_1$ just copies its input (from $\mathcal{D}_2$) to its output (to $\mathcal{D}_2$), while $\mathcal{D}_2$ is an inverter, i.e., input "a" yields output "b" and input "b" yields output "a". Since we have no time delays during

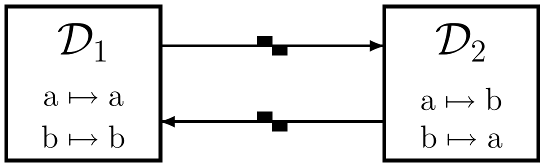


Figure 1.1: Example of an algebraic loop.

communication in our model, this connection leads to contradiction; there are no feasible values for the symbols exchanged between $\mathcal{D}_1$ and $\mathcal{D}_2$, and therefore, no DCES for the composed system can be given.

In synchronous languages, much research is done to examine such algebraic loops. One way of dealing with this is to demand *unique* solutions for the symbols between $\mathcal{D}_1$ and $\mathcal{D}_2$ (e.g. in [CPHP87, BG92, Hal93]). Systems which do not fulfill this property must not have that kind of mutual connection. Another way is to demand that there exists *at least one* solution (e.g. in [BG90, GGB87, Hal93]).

If the answer to question 2 is yes, one is interested in a *composition operator* which effectively constructs the DCES for the compound system, using the parts and information about the connections between them. For different kinds of interconnections, different composition operators have to be defined. In [SK91] two operators are given: The *cascade interconnection* operator defines the connection of a DCES's outputs with the inputs of another DCES, and the *feedback connection* operator defines the connection of a DCES's outputs with its own inputs. We additionally define the *loop interconnection* operator (where two DCESs are mutually connected like in Figure 1.1) and the *parallel interconnection* operator, which allows arbitrary connections among a set of DCESs. Two versions of the parallel interconnection operator are defined, one to handle systems with unique solutions and one for the case with multiple solutions. Both the loop and the parallel operator are also able to express feedback and cascade.

For verifying and model checking modularly composed systems, one can use the composition operators to combine all parts into one single system which then can be checked with respect to the desired properties. Such approaches often break space and time limits due to the state explosion problem. A more promising way is to find out which parts of the system are really needed to check a certain property; only these parts are then composed and checked.

In the untimed case we show how to transform a set of interconnected DCESs into the input language of the model checking tool SMV [McM92, McM93]. This gives us the possibility to apply verification by model checking to DCESs.

In [Huu98], TCESs are transformed into timed automata, which then can be model checked with tools like KRONOS [OY93], UPPAAL [BLL$^+$96], or HyTech [HHWT95, HH95]. This provides a wide range of possibilities to examine properties of TCESs by using existing tools for timed automata instead of building a new model checker specially designed for TCESs.

Structure of this Work

The main goal of this work is to provide useful operators for composing discrete condition/event systems as well as timed condition/event systems. We take a look at the existing operators of [SK91] (cascade interconnection and feedback connection) and reason about their applicability to large interconnected systems. We believe that these operators are not flexible enough to allow easy handling of complex interconnections. Therefore, we define the parallel interconnection operator which allows us to compose any finite number of systems with any kind of interconnection structure, provided a certain property forbidding algebraic loops holds.

We start in Chapter 2 with the basic notions like *condition symbols, event symbols,* and *signals. Condition/event systems* are defined as systems which relate input and output signals. Then we introduce *discrete condition/event systems* syntactically as finite transition systems with output functions for conditions and events. The semantics of a DCES is defined as a CES by using the notion of a *run* of the DCES over input and output signals, that is, a function mapping time to discrete states of the DCES such that the transition function and the output functions behave in accordance to the given signals.

Chapter 3 introduces the *cascade interconnection,* which is first defined for CESs in a *descriptive* way, that is, a CES $\mathcal{S}$ is said to be a cascade interconnection of two given CESs $\mathcal{S}_1$ and $\mathcal{S}_2$, if certain conditions hold. The nature of such a descriptive definition is that such a system $\mathcal{S}$ must not exist for all $\mathcal{S}_1$ and $\mathcal{S}_2$. In the discrete case however, we prove that for all DCESs $\mathcal{D}_1$ and $\mathcal{D}_2$ where $\mathcal{D}_1$'s output alphabets match $\mathcal{D}_2$'s input alphabets, a DCES $\mathcal{D}$ can be effectively constructed such that the semantics of $\mathcal{D}$ is a cascade interconnection of the semantics of $\mathcal{D}_1$ and $\mathcal{D}_2$. This effective construction defines the *cascade interconnection operator* for DCESs.

Then the *feedback connection* is defined, first in a descriptive way for CESs. We show that a feedback connection does not exist for all CESs. In the discrete case, we prove that for all DCESs $\mathcal{D}$ where the output alphabets match the input alphabets, a DCES $\mathcal{D}'$ can be effectively constructed such that the semantics of $\mathcal{D}'$ is a feedback connection of the semantics of $\mathcal{D}$, if a *unique solution property* is satisfied. This property is used to avoid algebraic loops in the feedback signals. The *feedback connection operator* for DCESs is defined by this effective construction.

We conclude Chapter 3 by defining the *loop interconnection* for CESs and the *loop interconnection operator* for DCESs. This describes a mutual connection between systems like in Figure 1.1. Again, a unique solution property is used in the discrete case to avoid algebraic loops. We show that the loop operator is able to express cascade as well as feedback.

In Chapter 4, we present an even more flexible way of connecting systems, the *parallel interconnection.* It allows arbitrary connections between systems, whereas all operators defined before have strong restrictions on the way systems can be connected. We obtain this new flexibility by introducing *names* to identify system components, and then consider components of different systems to be connected if they have the same name. Therefore, *named CESs* and *named DCESs* are introduced and used instead of CESs and DCESs. Our first version of the *parallel interconnection operator* again uses a unique solution property and is able to express cascade, feedback, and loop. After defining a *hiding operator* which removes output signals from CESs and DCESs, we present a second version of the parallel interconnection operator, which is able to handle *multiple solutions* for the signals connecting the subsystems.

Quantitative timing is considered in Chapter 5. We introduce *timed condition/event systems* as an extension of DCESs with features for quantitative timing. This is done by introducing *timers,* which are CESs that operate like a clock and can be reset by an external input event. The outputs of a timer are condition and event signals denoting where between a list of threshold values the current timer value is. A TCES is defined as a DCES which is connected to a set of timers. The DCES can reset timers via reset events and is able to use the information about the timer values to integrate quantitative timing into its transition and output functions.

Our multiple solution version of the parallel interconnection operator is augmented to systems consisting of TCESs instead of DCESs. Also the unique solution version can be expressed for TCESs. With these operators, we have reached our goal to present a powerful set of operators for composing timed condition/event systems.

An extensive example for the composition of TCESs is given in Chapter 6, where a part of the VHS Case Study CS1 (the chemical laboratory batch plant) is presented and modeled as a parallel interconnection of TCESs.

Chapter 7 shows how to transform a set of interconnected DCESs into the framework of the model checking tool SMV [McM92, McM93]. Each DCES is represented by one SMV module, and the modules communicate via shared variables. The result of the transformation is an SMV input file representing the parallel interconnection of the set of DCESs. We can formulate desired properties of the system in the logic CTL [CE81], and SMV can check the validity of these properties.

The last chapter contains conclusions and points out further work.

Acknowledgments

This work extends a Diplomarbeit (Master's thesis) at the University of Kiel [Luk98] and was partially supported by the European Community in the Esprit Long-Term Research Project 26270 – VHS (Verification of Hybrid Systems).

Notations

We present notational conventions used throughout this work. For a set X, we denote by 2^X the set of all subsets of X. Given a finite set D and a function $r : \mathbb{R}_{\geq 0} \to D$, we call r *finite-variable,* if in any bounded subinterval of $\mathbb{R}_{\geq 0}$, r changes its value only finitely often, i.e, r has only a finite number of discontinuity points. If r is finite-variable, r is furthermore called *right-continuous,* if for all $t \in \mathbb{R}_{\geq 0}$ there exists an $\varepsilon > 0$ such that $r(t + x) = r(t)$ for all $x \in (0, \varepsilon)$. And for any finite-variable r, we define for all $t \in \mathbb{R}_{>0}$ the limit from the left of r at t: $r(t^-) = \lim_{\varepsilon \to 0}^{\varepsilon > 0} r(t - \varepsilon)$.

For any n-tuple x, we denote by x_i the projection of x on its ith component. This notion is extended to Cartesian products of sets and to functions having a Cartesian product as range. The domain of a function f is denoted by $\mathrm{dom}(f)$, and its range by $\mathrm{ran}(f)$.

Chapter 2

Condition/Event Systems

Condition/event systems (CESs) are a class of continuous-time discrete event systems and have been introduced in [SK91]. The CES model is widely used to describe the behavior of discrete and hybrid systems. It allows to develop complex systems by means of block diagrams and signal flows, which is standard practice in system theory. A CES is defined as a relation between input and output *signals.* A signal can be understood as a function mapping points in time to a finite set, called *alphabet,* of symbols representing time-dependent information. A signal uses one of two different kinds of symbols, namely *condition symbols* and *event symbols. Condition signals* are piecewise constant functions and are used to represent time-dependent system states by condition symbols like "on", "off", or "waiting". Conditions symbols can be used to enable or disable transitions in systems with discrete state realizations. *Event signals* map points in time to a special *null symbol* except for some discrete points, in which so-called *proper events* occur. Proper events represent instantaneous actions like "start", "stop", or "reset"; they can be used to force transitions in a discrete state system in such a way that a transition is taken exactly at the moment the event occurs.

In this chapter, we formally define condition/event systems. We also introduce a model of condition/event systems with discrete state realizations, called *discrete* condition/event systems.

2.1 Basic Notions

Let D be a set of symbols. These will be the symbols in the range of condition and event signals.

Definition 2.1 (Condition alphabet, event alphabet) *A* condition alphabet *is a nonempty, finite set $U \subseteq D$, and its elements are called* condition symbols *or simply* conditions.[1]

An event alphabet *is a nonempty, finite set $V \subseteq D$ containing a special* null symbol $\mathbf{0}_V$. *Any $v \in V$ is called* event symbol, *or simply* event, *and if $v \neq \mathbf{0}_V$, it is also called* proper event. □

We write $\mathbf{0}$ for the null symbol $\mathbf{0}_V$, if the event alphabet V is known from the context. A Cartesian product $V = V_1 \times \ldots \times V_n$ of event alphabets is also an event alphabet, and we abbreviate its null symbol $(\mathbf{0}_{V_1}, \ldots, \mathbf{0}_{V_n})$ also with $\mathbf{0}_V$ or $\mathbf{0}$. In the following, we will use the letters U and Y for condition alphabets, and V and Z for event alphabets.

Condition and event alphabets serve as the range for *condition signals* and *event signals,* which are functions mapping points in time to such alphabets. In our framework, time is modeled by the set $\mathbb{R}_{\geq 0}$ of nonnegative real numbers.

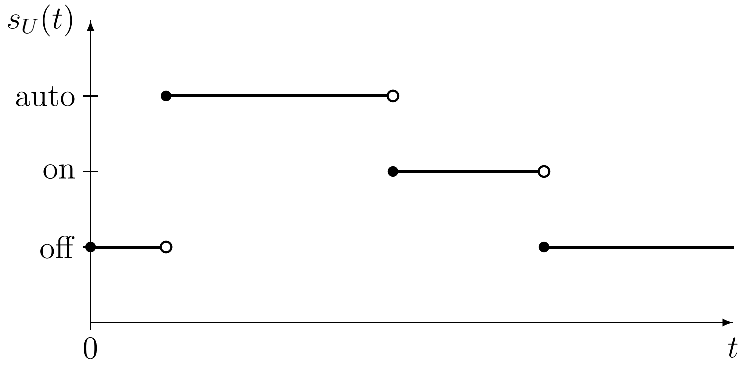


Figure 2.1: Example of a condition signal.

Definition 2.2 (Condition signal, event signal) *A* condition signal *over a condition alphabet U is a right-continuous, finite-variable function $s_U : \mathbb{R}_{\geq 0} \to U$. The set of all condition signals over U is denoted by $C(U)$.*

An event signal *over an event alphabet V is a function $s_V : \mathbb{R}_{\geq 0} \to V$, if $s_V(0) = \mathbf{0}$ and for any bounded subinterval I of $\mathbb{R}_{\geq 0}$, there are only finitely many $t \in I$ with $s_V(t) \neq \mathbf{0}$. The set of all event signals over V is denoted by $E(V)$.* □

[1] The reader is warned not to confuse the usage of "condition" as an abbreviation for "condition symbol" with the notion of "condition" for "property", "prerequisite", "Boolean guard", etc.

Figure 2.1 shows an example of a condition signal s_U over the condition alphabet $U = \{\text{on}, \text{off}, \text{auto}\}$.

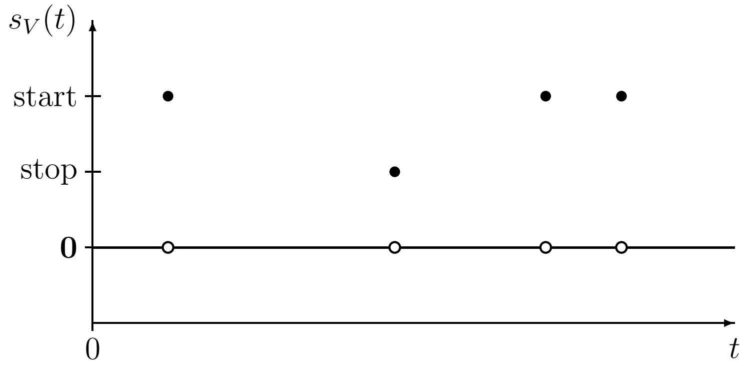


Figure 2.2: Example of an event signal.

Figure 2.2 shows an example of an event signal s_V over the event alphabet $V = \{\mathbf{0}, \text{start}, \text{stop}\}$.

2.2 Condition/Event Systems

Now we define condition/event systems. As said before, such a system is characterized by a relation between input and output signals. The relation can also be seen as a function mapping input signals to sets of output signals. So for each input there may be more than one possible output behavior, but there has to be at least one, since CESs are not allowed to refuse generating output (from a practical point of view, you can always measure some values at the CES's output "connectors").

Definition 2.3 (Condition/event system) *A quintuple*

$$\mathcal{S} = (U, V, Y, Z, S),$$

is called condition/event system *(CES), where*

- U *is the* input condition alphabet,
- V *is the* input event alphabet,
- Y *is the* output condition alphabet,
- Z *is the* output event alphabet, *and*

- $S : C(U) \times E(V) \to 2^{C(Y)\times E(Z)} \setminus \{\emptyset\}$ *is the* system behavior function.

We also use the symbol S *for the* system behavior relation $R \subseteq \mathrm{Beh}(U, V, Y, Z) = C(U) \times E(V) \times C(Y) \times E(Z)$ *with* $(s_U, s_V, s_Y, s_Z) \in R$ *if and only if* $(s_Y, s_Z) \in S(s_U, s_V)$. □

For the purpose of brevity, we will use the symbols s_U, s_V, s_Y, and s_Z to denote the four projections of a given behavior $s \in \mathrm{Beh}(U, V, Y, Z)$ on its components.

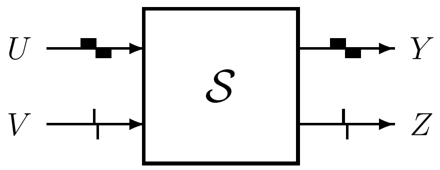


Figure 2.3: A block diagram of a condition/event system.

Figure 2.3 shows a block diagram of a condition/event system. We use an arrow like to denote the flow of a condition signal and for an event signal flow. If we need systems with more than these four signals, we use Cartesian products like $U = U_1 \times U_2$ as alphabets, and we draw signal flows for each U_i in the block diagrams (see, for example, Figure 3.4 on page 36). Formally, we access a certain alphabet U_i, its symbols $u_i \in U_i$, or its signals $s_{U_i} \in C(U_i)$ by using a projection on the ith component of U, $(u_1, u_2) \in U$, or $(s_{U_1}, s_{U_2}) \in C(U)$.

2.3 Discrete Condition/Event Systems

With the notion of condition/event systems we have introduced a model describing a relation of input and output signals. However, it is in practice very difficult to relate signals, since they range over the infinite time domain $\mathbb{R}_{\geq 0}$.

Moreover, changes in the *condition* input might force the CES to generate a proper output event at the time the condition change occurs. This behavior is not intended; only input *events* should be able to force output events. And since one specifies behaviors over the whole time domain, you can even build systems that are able to look into the future, e.g., a CES that outputs events one time unit *before* they occur as an input event.

Strange behaviors of such nature can be avoided by demanding certain properties like *causality, time-change invariance,* and *spontaneity* [SK91].

Another way of avoiding this, and, which is most important, a way of describing a CES in an *operational* way, is the model of *discrete* condition/event systems. These systems communicate with their environment via condition and event signals just like CESs, but an internal finite transition system is used to compute the current output symbols from the current input symbols at each point in time. The input conditions can enable or disable state changes in the transition system, and the input events may force transitions.

Definition 2.4 (Discrete condition/event system) *A* discrete condition/event system *(DCES)*

$$\mathcal{D} = (U, V, X, Y, Z, f, g, h, x_0)$$

consists of an input condition alphabet U, an input event alphabet V, an output condition alphabet Y, an output event alphabet Z, and

- *a finite set X of* states,
- *a* state transition function $f : X \times U \times V \to 2^X \setminus \{\emptyset\}$,
- *a* condition output function $g : X \times U \to Y$,
- *an* event output function $h : X \times X \times V \to Z$*, and*
- *an initial state $x_0 \in X$.* □

We give a short intuitive description of how DCESs work. The formal semantics can be found in Definition 2.7. The function f is used to determine the next state of the transition system. As parameters f has the current state and the current condition and event input symbols. A nonempty set of states is computed by f, and one of these is nondeterministically chosen as the next state. The condition output symbol is generated by g from the current state and input condition symbol. The function h produces the event output symbol; parameters are the current state, the next state (which was chosen from the set of possible ones), and the current event input symbol. Note that g and h allow no nondeterministic choice; this is possible in f only.

Now we introduce two properties which lead to an interesting subclass of DCESs, the *well-behaved* DCESs.

Definition 2.5 (Well-behaved DCES) *Let $\mathcal{D}$ be a DCES as in Definition 2.4. $\mathcal{D}$ is called* well-behaved, *if it satisfies*

- stuttering: $x \in f(x, u, \mathbf{0})$*, and*

- output triggering: $h(x, x, \mathbf{0}) = \mathbf{0}$,

for all $x \in X$ *and* $u \in U$. $\square$

Stuttering describes that the system is always able to stay in the same state as long as no proper input event occurs. This ensures that changes in the input condition *cannot* force any transitions. Output triggering means that the system does not generate a proper output event if it stays in the same state and no proper input event occurs. This prevents the system from generating infinitely many proper events in finite time, since in a finite time interval only finitely many state changes are allowed (this is required in Definition 2.6 below).

The operative nature of DCESs, well-behavedness, and its semantics (defined below) ensure the properties mentioned above (causality, time-change invariance, spontaneity), which avoid the "strange behaviors". In the following, we will therefore use well-behaved DCESs only, without explicitly mentioning it. However, when constructing a DCES by giving explicit definitions for f, g, and h, we do have to show that the DCES is well-behaved.

To define the semantics for DCESs, we define *runs* of DCESs. Intuitively, a run is a function mapping points in time to the state of the DCES's transition system at that time. A run r always starts at the initial state. At all moments $t \in \mathbb{R}_{\geq 0}$, the value of the condition output signal $s_Y(t)$ is computed by g using as parameters the current state $r(t)$ and the current value of the condition input signal $s_U(t)$. At each point in time $t \in \mathbb{R}_{>0}$, a set of states is generated by f using the last state $r(t^-)$, the input condition symbol $s_U(t^-)$ just before time t, and the current input event symbol $s_V(t)$. One of these states is nondeterministically chosen as the new state $r(t)$. The value of the event output signal $s_Z(t)$ is computed by h using the last state $r(t^-)$, the current state $r(t)$ and the current value of the event input signal $s_V(t)$.

Note that nothing is said about the event output symbol at time 0, since the definition of event signal implies $s_Z(0) = \mathbf{0}$.

The usage of $s_U(t^-)$ as parameter for f emphasizes that conditions cannot force transitions; if the condition input signal changes just at time t, still the old value will be used. So this change can only influence transitions *after* time t, but not the transition *at* time t.

Definition 2.6 (Run of a DCES) *Let $\mathcal{D}$ be a DCES as in Definition 2.4. We call a right-continuous and finite-variable function $r : \mathbb{R}_{\geq 0} \to X$* run *of $\mathcal{D}$ over $s \in \mathrm{Beh}(U, V, Y, Z)$, if and only the following conditions are satisfied by r:*

1. $r(t) \in f(r(t^-), s_U(t^-), s_V(t))$ *for all* $t \in \mathbb{R}_{>0}$,

2. $s_Y(t) = g(r(t), s_U(t))$ *for all* $t \in \mathbb{R}_{\geq 0}$,
3. $s_Z(t) = h(r(t^-), r(t), s_V(t))$ *for all* $t \in \mathbb{R}_{>0}$, *and*
4. $r(0) = x_0$. □

Now we can assign a semantics to DCESs. The semantics of a DCES $\mathcal{D}$ is a CES $\mathcal{S}_\mathcal{D}$ with a behavior relation containing all the behaviors matching to possible runs of $\mathcal{D}$.

Definition 2.7 (Semantics of a DCES) *Let $\mathcal{D}$ be a DCES as in Definition 2.4. The* semantics *of $\mathcal{D}$ is given by the CES $\mathcal{S}_\mathcal{D} = (U, V, Y, Z, S_\mathcal{D})$ with $s \in S_\mathcal{D}$ if and only if there exists a run r of $\mathcal{D}$ over s, for all $s \in$ $\mathrm{Beh}(U, V, Y, Z)$.* □

There is one crucial question which arises immediately after having defined the semantic relation between CESs and DCESs: Can every condition/event system be described by a discrete condition/event system? Or, speaking formally, can we find for every CES $\mathcal{S}$ a DCES $\mathcal{D}$ such that $\mathcal{S}_\mathcal{D} = \mathcal{S}$? The answer is obviously no, as shown in the following example.

Example 2.8 We define a simple CES which has only trivial inputs and outputs except for the event output. The only proper output event is a "ping" which can be observed at every time point $t \in \mathbb{N}$, i.e., the time difference between two "pings" is exactly one time unit. Let $\mathcal{S} = (U, V, Y, Z, S)$ be the CES with the trivial condition alphabets[2] $U = Y = \{0\}$, event alphabets $V = \{\mathbf{0}\}$, $Z = \{\mathbf{0}, \mathrm{ping}\}$, and, for all $s \in \mathrm{Beh}(U, V, Y, Z)$,

$$s \in S \quad \text{if and only if} \quad s_Z(t) = \begin{cases} \mathrm{ping}, & \text{if } t \in \mathbb{N} \\ \mathbf{0} & \text{, otherwise} \end{cases}.$$

Note that the trivial alphabets allow only one possibility for s_U, s_V, and s_Y.

This CES cannot be modeled as a DCES, since DCESs have no means to do quantitative timing. In other words, one can build a DCES that outputs a "ping" from time to time, but there is no way to measure time, so the time difference between two "pings" cannot be forced to be exactly one time unit. ◇

In Chapter 5 we will introduce *timed condition/event systems*, which are DCESs augmented with timers capable of doing quantitative timing. Using these systems, the behavior above becomes expressible for discrete systems, as we will see in Example 5.9 on page 87.

[2] In our examples, we use $\{0\}$ as a trivial condition alphabet. This is quite arbitrary; one can use any singleton $U \subseteq D$.

Chapter 3

Connecting Condition/Event Systems

When dealing with a large and complex system in a formal framework like specification languages, programming languages, proof systems, timing diagrams, or, as in this work, condition/event systems, it is always a bad idea to treat this system as a whole, i.e., in one monolithic part. Complex systems tend to be unmanageable because of their size and should therefore always be dividable into smaller parts, called *modules.* This property, called *modularity,* allows to reuse previously defined modules, to split up work between several teams, and, which is most important, to use *compositional* reasoning in verification, model checking, and testing. *Compositionality* [HdR86, HdR90] means that properties of the complex system can be derived from properties of its modules, and that properties of a module are independent from the system it is part of. Compositionality is becoming more and more important, since the growing complexity of systems often makes it impossible to specify, design, or verify a system as a whole due to the state explosion problem. Among other techniques like the use of symbolic representations [HNSY94], compositionality is a promising way to handle large systems.

An important part of a compositional framework is a set of *composition operators* which describe in a formal way how modules can be connected and how the resulting system can be derived from the modules being parts of the connection. In our framework, we will use the following strategy to examine a connection between CESs or DCESs: First, we define the semantics of a connection among CESs in a *descriptive* way, that is, we say that a CES $\mathcal{S}$ describes the semantics of some connected CESs, if some properties hold. We do *not* check formally under which circumstances such a CES $\mathcal{S}$ exists. In a second step, we define the connection for DCESs in an *operational* way, that is, we introduce a composition operator for DCESs by saying under which circumstances a DCES $\mathcal{D}$ exists such that the semantics of $\mathcal{D}$ is the semantics of some connected DCESs, and by giving an effective construction for $\mathcal{D}$. Since the semantics of a DCES is a CES, the semantic reasoning in this second step is based on the descriptive definition in the first step.

In this chapter, we examine several ways of connecting CESs and DCESs. First we have a look at the *cascade interconnection* and the *feedback connection* of [SK91]. Then we introduce the *loop interconnection*, which combines the properties of cascade and feedback. In Chapter 4, we will introduce the *parallel interconnection*, which is augmented for composing timed condition/event systems in Chapter 5.

3.1 The Cascade Interconnection

The cascade interconnection describes the connection of one system's outputs with a second system's inputs as defined in [SK91]. Figure 3.1 shows a cascade interconnection of two CESs. The condition and event alphabets of the connected components have to contain the same symbols. If not all symbols in the input alphabets of $\mathcal{S}_2$ need to be used in the connecting signals, a less strict definition could allow $Y_1 \subseteq U_2$ and $Z_1 \subseteq V_2$.

3.1.1 The Cascade Interconnection of C/E Systems

We define a cascade interconnection of two CESs $\mathcal{S}_1$ and $\mathcal{S}_2$ as a CES $\mathcal{S}$ having the input alphabets of $\mathcal{S}_1$ and the output alphabets of $\mathcal{S}_2$. Intuitively, the behavior of $\mathcal{S}$ is given by using the input signals of $\mathcal{S}$ as input for $\mathcal{S}_1$, the output signals of $\mathcal{S}_1$ as input for $\mathcal{S}_2$, and the output signals of $\mathcal{S}_2$ as output for $\mathcal{S}$. The signals between $\mathcal{S}_1$ and $\mathcal{S}_2$ are not visible in $\mathcal{S}$.

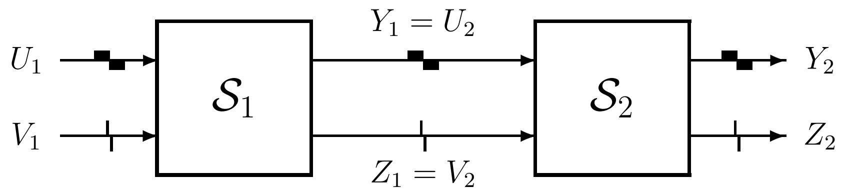


Figure 3.1: CES cascade interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$.

The following definition does not say if such a CES $\mathcal{S}$ exists; it only defines if a given CES $\mathcal{S}$ is a cascade interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$.

Definition 3.1 (Cascade interconnection) *Let $\mathcal{S}_i = (U_i, V_i, Y_i, Z_i, S_i)$ be a CES, $i \in \{1, 2\}$, with $Y_1 = U_2$ and $Z_1 = V_2$. A CES $\mathcal{S} = (U, V, Y, Z, S)$ is called* cascade interconnection *of $\mathcal{S}_1$ and $\mathcal{S}_2$, if and only if the following conditions hold:*

1. *$U = U_1$, $V = V_1$, $Y = Y_2$, $Z = Z_2$,*
2. *$S = S_1 \circ S_2$, i.e., for any $s \in \mathrm{Beh}(U, V, Y, Z)$, we have $s \in S$ if and only if there exists $(s'_U, s'_V) \in S_1(s_U, s_V)$ such that $(s_Y, s_Z) \in S_2(s'_U, s'_V)$.* □

The following lemma states that such a CES $\mathcal{S}$ can always be constructed from the given CESs $\mathcal{S}_1$ and $\mathcal{S}_2$.

Lemma 3.2 *Given two CESs $\mathcal{S}_1$ and $\mathcal{S}_1$ as in Definition 3.1, a CES $\mathcal{S}$ can be constructed such that $\mathcal{S}$ is a cascade interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$.* □

Proof Let $\mathcal{S}_1$ and $\mathcal{S}_1$ be CESs as in Definition 3.1. We define $\mathcal{S} = (U_1, V_1, Y_2, Z_2, S)$ with $s \in S$ if and only if there exists $(s'_U, s'_V) \in S_1(s_U, s_V)$ such that $(s_Y, s_Z) \in S_2(s'_U, s'_V)$, for all $s \in \mathrm{Beh}(U_1, V_1, Y_2, Z_2)$.

Both conditions of Definition 3.1 are obviously fulfilled. It remains to check that $\mathcal{S}$ is well-defined, i.e., that for every pair $(s_U, s_V) \in C(U_1) \times E(V_1)$ of input signals there exists at least one pair $(s_Y, s_Z) \in C(Y_2) \times E(Z_2)$ of output signals such that $(s_U, s_V, s_Y, s_Z) \in S$. Let $(s_U, s_V) \in C(U_1) \times E(V_1)$. Since $\mathcal{S}_1$ is well-defined, we find at least one $(s'_U, s'_V) \in S_1(s_U, s_V)$. And since $\mathcal{S}_2$ is also well-defined, we find at least one $(s_Y, s_Z) \in S_2(s'_U, s'_V)$. By definition of S, this leads to $(s_U, s_V, s_Y, s_Z) \in S$. ■

3.1.2 The Cascade Interconnection of Discrete C/E Systems

The observation of Lemma 3.2 also holds for DCESs; note that the following theorem makes use of the DCESs' semantics, since the notion of cascade interconnection in Definition 3.1 is given for CESs only.

Theorem 3.3 *Given two DCES $\mathcal{D}_i = (U_i, V_i, X_i, Y_i, Z_i, f_i, g_i, h_i, x_{0,i})$, $i \in \{1, 2\}$, with $Y_1 = U_2$ and $Z_1 = V_2$, a DCES $\mathcal{D}$ can be effectively constructed such that $\mathcal{S}_{\mathcal{D}}$ is a cascade interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$.* □

The proof of this theorem shows the effective construction of $\mathcal{D}$, which is done as follows: The set of states of $\mathcal{D}$ is the Cartesian product of the state sets of $\mathcal{D}_1$ and $\mathcal{D}_2$. The state transitions in the first component of $\mathcal{D}$'s states are the same as in $\mathcal{D}_1$ using f_1. The state transitions in the second component are done like in $\mathcal{D}_2$, but the condition and event parameters for f_2 are the outputs of g_1 and h_1. The condition and event output of $\mathcal{D}$ is computed by g_2 and h_2, also using the outputs of g_1 and h_1.

Proof Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be given as above. We construct a DCES $\mathcal{D}$ such that $\mathcal{S}_\mathcal{D}$ is a cascade interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$. Let $\mathcal{D} = (U, V, X, Y, Z, f, g, h, x_0)$ be the DCES with $U = U_1$, $V = V_1$, $Y = Y_2$, $Z = Z_2$, $X = X_1 \times X_2$, $x_0 = (x_{0,1}, x_{0,2})$, and, for all $x = (x_1, x_2) \in X$, $x' = (x'_1, x'_2) \in X$, $u \in U$, $v \in V$,

- $x' \in f(x, u, v)$ if and only if $x'_1 \in f_1(x_1, u, v)$ and $x'_2 \in f_2(x_2, g_1(x_1, u), h_1(x_1, x'_1, v))$,
- $g(x, u) = g_2(x_2, g_1(x_1, u))$,
- $h(x, x', v) = h_2(x_2, x'_2, h_1(x_1, x'_1, v))$.

It can easily be checked that $\mathcal{D}$ is well-behaved, since $\mathcal{D}_1$ and $\mathcal{D}_2$ are well-behaved. Let $x = (x_1, x_2) \in X$ and $u \in U$. *Output triggering:* $h(x, x, \mathbf{0}) = h_2(x_2, x_2, h_1(x_1, x_1, \mathbf{0})) = h_2(x_2, x_2, \mathbf{0}) = \mathbf{0}$. *Stuttering:* We have $x_1 \in f_1(x_1, u, \mathbf{0})$ and $x_2 \in f_2(x_2, g_1(x_1, u), \mathbf{0}) = f_2(x_2, g_1(x_1, u), h_1(x_1, x_1, \mathbf{0}))$, so $x \in f(x, u, \mathbf{0})$.

Now we check if $\mathcal{S}_\mathcal{D}$ is a cascade interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$. Condition 1 of Definition 3.1 holds obviously. For proving condition 2, let $s \in \mathrm{Beh}(U, V, Y, Z)$. We have to show that $s \in S_\mathcal{D}$ if and only if there exists $(s'_U, s'_V) \in S_{\mathcal{D}_1}(s_U, s_V)$ such that $(s_Y, s_Z) \in S_{\mathcal{D}_2}(s'_U, s'_V)$. We have

$$s \in S_\mathcal{D}$$

if and only if (by Definition 2.7)

$$\begin{aligned}
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} : \\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f(r(t^-), s_U(t^-), s_V(t)) \\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : s_Y(t) = g(r(t), s_U(t)) \\
&3.\ \forall t \in \mathbb{R}_{>0} : s_Z(t) = h(r(t^-), r(t), s_V(t)) \\
&4.\ r(0) = x_0
\end{aligned}$$

if and only if (by construction of f, g, h, and x_0 of the DCES $\mathcal{D}$)

$$
\begin{array}{ll}
\multicolumn{2}{l}{\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} :} \\
1.(a) & \forall t \in \mathbb{R}_{>0} : r(t)_1 \in f_1(r(t^-)_1, s_U(t^-), s_V(t)) \\
(b) & \forall t \in \mathbb{R}_{>0} : r(t)_2 \in f_2(r(t^-)_2, s'_U(t^-), s'_V(t)) \\
2. & \forall t \in \mathbb{R}_{\geq 0} : s_Y(t) = g_2(r(t)_2, s'_U(t)) \\
3. & \forall t \in \mathbb{R}_{>0} : s_Z(t) = h_2(r(t^-)_2, r(t)_2, s'_V(t)) \\
4.(a) & r(0)_1 = x_{0,1} \\
(b) & r(0)_2 = x_{0,2} \\
\multicolumn{2}{l}{\text{where}} \\
\multicolumn{2}{l}{s'_U(t) = g_1(r(t)_1, s_U(t))} \\
\multicolumn{2}{l}{s'_V(t) = h_1(r(t^-)_1, r(t)_1, s_V(t))}
\end{array}
$$

if and only if (we introduce existential quantification for s'_U and s'_V, and we split r into r_1 and r_2)

$$
\begin{array}{l}
\exists s'_U \in C(Y_1), s'_V \in E(Z_1) : \\
\text{(a) } \exists r_1 : \mathbb{R}_{\geq 0} \to X_1 \text{ (right-continuous, finite-variable)} : \\
\quad 1.\ \forall t \in \mathbb{R}_{>0} : r_1(t) \in f_1(r_1(t^-), s_U(t^-), s_V(t)) \\
\quad 2.\ \forall t \in \mathbb{R}_{\geq 0} : s'_U(t) = g_1(r_1(t), s_U(t)) \\
\quad 3.\ \forall t \in \mathbb{R}_{>0} : s'_V(t) = h_1(r_1(t^-), r_1(t), s_V(t)) \\
\quad 4.\ r_1(0) = x_{0,1} \\
\text{(b) } \exists r_2 : \mathbb{R}_{\geq 0} \to X_2 \text{ (right-continuous, finite-variable)} : \\
\quad 1.\ \forall t \in \mathbb{R}_{>0} : r_2(t) \in f_2(r_2(t^-), s'_U(t^-), s'_V(t)) \\
\quad 2.\ \forall t \in \mathbb{R}_{\geq 0} : s_Y(t) = g_2(r_2(t), s'_U(t)) \\
\quad 3.\ \forall t \in \mathbb{R}_{>0} : s_Z(t) = h_2(r_2(t^-), r_2(t), s'_V(t)) \\
\quad 4.\ r_2(0) = x_{0,2}
\end{array}
$$

if and only if (by Definition 2.7)

$$\exists (s'_U, s'_V) \in S_{\mathcal{D}_1}(s_U, s_V) : (s_Y, s_Z) \in S_{\mathcal{D}_2}(s'_U, s'_V). \qquad \blacksquare$$

Finally, we can define the cascade interconnection operator for DCESs by using the construction in the previous proof.

Definition 3.4 (Discrete cascade interconnection) *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two DCESs as in Theorem 3.3. The DCES $\mathcal{D}$ gained from the construction above is called the* discrete cascade interconnection *of $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathcal{D}_1 \to \mathcal{D}_2$.* □

3.2 The Feedback Connection

In [SK91], the feedback connection describes the connection of one system with itself, i.e., its condition output component is connected to its condition input component, and its event output component is connected to its event input component. The resulting system has no inputs, and the output signals remain visible. Figure 3.2 shows such a feedback connection.

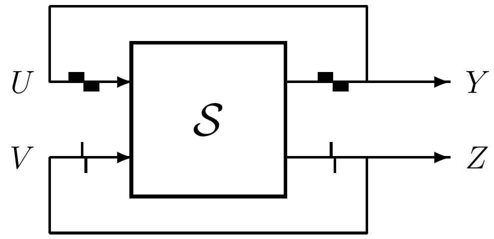


Figure 3.2: The feedback connection in [SK91].

Our notion of the feedback connection is slightly different. We have additional input and output components, and the signals in the feedback paths are not visible as output components of the resulting system. They can however be made visible by defining the system in such a way that the feedback signals are copied to the non-feedback components. Figure 3.3 shows a feedback connection of a CES. Like in the cascade interconnection, we demand matching alphabets $Y_2 = U_2$ and $Z_2 = V_2$. Again, this can be formulated less strict as $Y_2 \subseteq U_2$ and $Z_2 \subseteq V_2$.

3.2.1 The Feedback Connection of C/E Systems

Given a CES $\mathcal{S}$ with all alphabets being a Cartesian product of two single alphabets, we define a feedback connection of $\mathcal{S}$ as a CES $\mathcal{S}'$ having as alphabets only the first components of the alphabets of $\mathcal{S}$. Intuitively, the behaviors of $\mathcal{S}'$ are the behaviors s of $\mathcal{S}$ restricted to the first components, where we allow only the behaviors s with a feedback solution, i.e., where in the condition, respectively event, signal in the second components of s we have the same signal for both the input and the output part.

As in Definition 3.1, we do not say if such a CES $\mathcal{S}'$ exists; we only define if a given CES $\mathcal{S}'$ is a feedback connection of $\mathcal{S}$.

Definition 3.5 (Feedback connection) *Let* $\mathcal{S} = (U_1 \times U_2, V_1 \times V_2, Y_1 \times Y_2, Z_1 \times Z_2, S)$ *be a C/E system with* $Y_2 = U_2$ *and* $Z_2 = V_2$*. A C/E system*

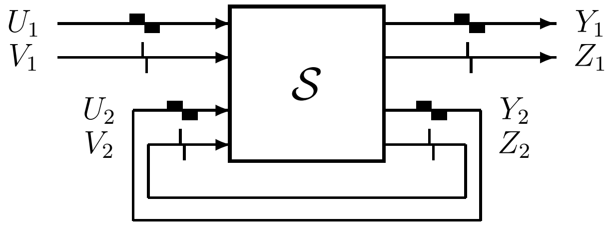


Figure 3.3: CES feedback connection of $\mathcal{S}$.

$\mathcal{S}' = (U, V, Y, Z, S')$ is called feedback connection *of $\mathcal{S}$, if and only if the following conditions hold:*

1. *$U = U_1$, $V = V_1$, $Y = Y_1$, $Z = Z_1$,*
2. *For any $s \in \mathrm{Beh}(U, V, Y, Z)$, we have $s \in S'$ if and only if there exists $s_{Y_2} \in C(Y_2)$ and $s_{Z_2} \in E(Z_2)$ such that $((s_U, s_{Y_2}), (s_V, s_{Z_2}), (s_Y, s_{Y_2}), (s_Z, s_{Z_2})) \in S$.* □

In contrast to the cascade interconnection, we cannot find for every CES $\mathcal{S}$ with alphabets as in Definition 3.5 a CES $\mathcal{S}'$ which is a feedback connection of $\mathcal{S}$. This is caused by algebraic loops, as shown in the following lemma.

Lemma 3.6 *There exists a CES $\mathcal{S}$ fulfilling the syntactic requirements of Definition 3.5 having no feedback connection.* □

Proof Let $\mathcal{S} = (U_1 \times U_2, V_1 \times V_2, Y_1 \times Y_2, Z_1 \times Z_2, S)$ be the C/E system with $U_1 = Y_1 = \{0\}$, $V_1 = V_2 = Z_1 = Z_2 = \{\mathbf{0}\}$, $U_2 = Y_2 = \{\mathrm{a}, \mathrm{b}\}$ and $S = \{s \in \mathrm{Beh}(U_1 \times U_2, V_1 \times V_2, Y_1 \times Y_2, Z_1 \times Z_2) \mid s_{Y_2}(t) \neq s_{U_2}(t) \text{ for all } t \in \mathbb{R}_{\geq 0}\}$. Thus, $\mathcal{S}$ has only two non-trivial components, the input and output condition in the feedback part. Since S is defined in such a way that the signals of these components never match (if $s_{Y_2}(t) = \mathrm{a}$, then $s_{U_2}(t) = \mathrm{b}$ and vice versa), we cannot find any $s_{Y_2} \in C(Y_2)$ which meets the requirements in condition 2 of Definition 3.5. So a feedback connection of $\mathcal{S}$ must have an empty set of behaviors, which is not allowed for well-defined CESs, see Definition 2.3. ■

3.2.2 The Feedback Connection of Discrete C/E Systems

Since the CES $\mathcal{S}$ given in the proof above can also be modeled as a DCES (see upcoming Example 3.8), it is clear that Lemma 3.6 also holds for DCESs.

So not for every DCES $\mathcal{D}$ with alphabets as in Definition 3.5 there exists a DCES $\mathcal{D}'$ such that $\mathcal{S}_{\mathcal{D}'}$ is a feedback connection of $\mathcal{S}_{\mathcal{D}}$.

If we want to define a feedback connection operator for DCESs, we need to have a property that defines for a given DCES $\mathcal{D}$ if there exists a DCES $\mathcal{D}'$ with $\mathcal{S}_{\mathcal{D}'}$ being a feedback connection of $\mathcal{S}_{\mathcal{D}}$. Such a property must forbid the existence of algebraic loops in the given DCES $\mathcal{D}$.

In [SK91], where the feedback connection has no additional inputs (see Figure 3.2), this can be done easily be demanding that there exists a solution y for the condition feedback equation $y = g(x_0, y)$. Then such a solution y can be used to generate the output condition of $\mathcal{D}'$ by defining $g'(x_0) = y$ (remember that $\mathcal{D}'$ has no inputs). Since we have no external input events, stuttering allows us to remain in the initial state x_0, and by output triggering, the null event $\mathbf{0}$ is always a solution for the event feedback. So we have at least one run (always at the initial state x_0) of $\mathcal{D}'$ over a behavior that can be regarded as a feedback behavior. However, if there is more than one solution y for the equation $y = g(x_0, y)$, only one solution can be used for the initial state, and $S_{\mathcal{D}'}$ will only contain a subset of the behaviors of a feedback connection of $\mathcal{S}_{\mathcal{D}}$ (this is discussed in detail in Example 4.26 on page 69).

Our notion of a feedback connection makes the definition of such a property even more difficult. Since we have additional external inputs, the existence of a solution for the feedback signals may depend on these external input signals. Therefore, we demand for each state of $x \in X$ and each external input condition symbol $u_1 \in U_1$ that there exists a condition symbol $y_2 \in Y_2$ which is a solution for the condition feedback equation, and we demand for each pair of pre- and post-states $x, x' \in X$ and each external input event symbol $v_1 \in V_1$ that there exists an event symbol $z_2 \in Z_2$ which is a solution for the event feedback equation.

Furthermore, we demand *unique* solutions, since this makes the construction in the following proof much easier. An operator handling *multiple* solutions will be presented in Section 4.6.

Theorem 3.7 *Given a DCES* $\mathcal{D} = (U_1 \times U_2, V_1 \times V_2, X, Y_1 \times Y_2, Z_1 \times Z_2, f, g, h, x_0)$*, with* $Y_2 = U_2$ *and* $Z_2 = V_2$*, a DCES* $\mathcal{D}'$ *can be effectively constructed such that* $\mathcal{S}_{\mathcal{D}'}$ *is a feedback connection of* $\mathcal{S}_{\mathcal{D}}$*, if the following* unique solution property *holds for* $\mathcal{D}$*:*

- *For all* $x \in X$ *and* $u_1 \in U_1$*, there exists a unique* $y_2 \in Y_2$ *such that* $g(x, (u_1, y_2))_2 = y_2$*.*
- *For all* $x, x' \in X$ *and* $v_1 \in V_1$*, there exists a unique* $z_2 \in Z_2$ *such that* $h(x, x', (v_1, z_2))_2 = z_2$*.* □

Before we prove Theorem 3.7, we present a DCES where the unique solution property does not hold.

Example 3.8 Reconsider the CES $\mathcal{S}$ of Lemma 3.6. We define a DCES $\mathcal{D}$ which has the CES $\mathcal{S}$ as semantics. Let $\mathcal{D} = (U_1 \times U_2, V_1 \times V_2, \{x\}, Y_1 \times Y_2, Z_1 \times Z_2, f, g, h, x)$ be the DCES with $U_1 = Y_1 = \{0\}$, $V_1 = V_2 = Z_1 = Z_2 = \{\mathbf{0}\}$, $U_2 = Y_2 = \{\mathrm{a}, \mathrm{b}\}$ and, for all $(u_1, u_2) \in U_1 \times U_2$ and $(v_1, v_2) \in V_1 \times V_2$,

- $f(x, (u_1, u_2), (v_1, v_2)) = \{x\}$,
- $g(x, (u_1, u_2)) = \begin{cases} (0, \mathrm{b}), \text{ if } u_2 = \mathrm{a} \\ (0, \mathrm{a}), \text{ if } u_2 = \mathrm{b} \end{cases}$, and
- $h(x, x, (v_1, v_2)) = (\mathbf{0}, \mathbf{0})$.

As semantics of $\mathcal{D}$, we have $\mathcal{S}_{\mathcal{D}} = \mathcal{S}$. Together with Lemma 3.6, this shows that not for every DCES $\mathcal{D}$ with alphabets as in Definition 3.5 there exists a DCES $\mathcal{D}'$ such that $\mathcal{S}_{\mathcal{D}'}$ is a feedback connection of $\mathcal{S}_{\mathcal{D}}$.

And since $g(x, (0, u_2))_2 \neq u_2$ follows directly from the definition of g, the unique solution property does not hold. $\diamond$

Now we prove Theorem 3.7. This is done by giving the effective construction of $\mathcal{D}'$, which is done as follows: The set of states of $\mathcal{D}'$ is the set of states of $\mathcal{D}$. The functions f', g', and h' for the state transitions and the condition and event symbol outputs are defined using the functions f, g, and h of $\mathcal{D}$ and the unique solutions that have been found while checking the unique solution property.

Proof Let $\mathcal{D}$ be given as above, let the unique solution property hold. Let $G(x, u_1)$ and $H(x, x', v_1)$ be the unique solutions y_2 and z_2, respectively.

We construct a DCES $\mathcal{D}'$ such that $\mathcal{S}_{\mathcal{D}'}$ is a feedback connection of $\mathcal{S}_{\mathcal{D}}$. Let $\mathcal{D}' = (U_1, V_1, X, Y_1, Z_1, f', g', h', x_0)$ be the DCES with

- $x' \in f'(x, u_1, v_1)$ if and only if
  $x' \in f\Big(x, (u_1, G(x, u_1)), (v_1, H(x, x', v_1))\Big)$,
- $g'(x, u_1) = g\Big(x, (u_1, G(x, u_1))\Big)_1$,
- $h'(x, x', v_1) = h\Big(x, x', (v_1, H(x, x', v_1))\Big)_1$,

for all $x, x' \in X$, $u_1 \in U_1$, $v_1 \in V_1$. The DCES $\mathcal{D}'$ is well-behaved, since $\mathcal{D}$ is well-behaved, and therefore, $H(x, x, \mathbf{0}_{V_1}) = \mathbf{0}_{Z_2}$ for all $x \in X$.

Now we check if $\mathcal{S}_{\mathcal{D}'}$ is a feedback connection of $\mathcal{S}_{\mathcal{D}}$. Condition 1 of Definition 3.5 holds obviously. For proving condition 2, let $(s_U, s_V, s_Y, s_Z) \in \mathrm{Beh}(U_1, V_1, Y_1, Z_1)$. We have to show that $(s_U, s_V, s_Y, s_Z) \in S_{\mathcal{D}'}$ if and only if there exist $s_{Y_2} \in C(Y_2)$ and $s_{Z_2} \in E(Z_2)$ such that $((s_U, s_{Y_2}), (s_V, s_{Z_2}), (s_Y, s_{Y_2}), (s_Z, s_{Z_2})) \in S_{\mathcal{D}}$.

We have

$$(s_U, s_V, s_Y, s_Z) \in S_{\mathcal{D}'}$$

if and only if (by Definition 2.7)

$$\begin{aligned}
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} : \\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f'(r(t^-), s_U(t^-), s_V(t)) \\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : s_Y(t) = g'(r(t), s_U(t)) \\
&3.\ \forall t \in \mathbb{R}_{>0} : s_Z(t) = h'(r(t^-), r(t), s_V(t)) \\
&4.\ r(0) = x_0
\end{aligned}$$

if and only if (by construction of the DCES $\mathcal{D}'$)

$$\begin{aligned}
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} : \\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f\Big(r(t^-), \Big(s_U(t^-), G(r(t^-), s_U(t^-))\Big), \\
&\qquad\qquad\qquad\qquad\quad \Big(s_V(t), H(r(t^-), r(t), s_V(t))\Big)\Big) \\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : s_Y(t) = g\Big(r(t), \Big(s_U(t), G(r(t), s_U(t))\Big)\Big)_1 \\
&3.\ \forall t \in \mathbb{R}_{>0} : s_Z(t) = h\Big(r(t^-), r(t), \Big(s_V(t), H(r(t^-), r(t), s_V(t))\Big)\Big)_1 \\
&4.\ r(0) = x_0
\end{aligned}$$

if and only if (since G and H describe unique signals, they can be replaced by the existentially quantified signals s_{Y_2} and s_{Z_2})

$$\begin{aligned}
&\exists s_{Y_2} \in C(Y_2), s_{Z_2} \in E(Z_2) : \\
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} : \\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f\Big(r(t^-), (s_U(t^-), s_{Y_2}(t^-))), (s_V(t), s_{Z_2}(t))\Big) \\
&2.\text{a. } \forall t \in \mathbb{R}_{\geq 0} : s_Y(t) = g\Big(r(t), (s_U(t), s_{Y_2}(t))\Big)_1 \\
&2.\text{b. } \forall t \in \mathbb{R}_{\geq 0} : s_{Y_2}(t) = g\Big(r(t), (s_U(t), s_{Y_2}(t))\Big)_2 \\
&3.\text{a. } \forall t \in \mathbb{R}_{>0} : s_Z(t) = h\Big(r(t^-), r(t), (s_V(t), s_{Z_2}(t))\Big)_1 \\
&3.\text{b. } \forall t \in \mathbb{R}_{>0} : s_{Z_2}(t) = h\Big(r(t^-), r(t), (s_V(t), s_{Z_2}(t))\Big)_2 \\
&4.\ r(0) = x_0
\end{aligned}$$

if and only if

$$\begin{aligned}
&\exists s_{Y_2} \in C(Y_2), s_{Z_2} \in E(Z_2) : \\
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} : \\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f\Big(r(t^-), (s_U(t^-), s_{Y_2}(t^-))), (s_V(t), s_{Z_2}(t))\Big) \\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : (s_Y(t), s_{Y_2}(t)) = g\Big(r(t), (s_U(t), s_{Y_2}(t))\Big) \\
&3.\ \forall t \in \mathbb{R}_{>0} : (s_Z(t), s_{Z_2}(t)) = h\Big(r(t^-), r(t), (s_V(t), s_{Z_2}(t))\Big) \\
&4.\ r(0) = x_0
\end{aligned}$$

if and only if (by Definition 2.7)

$$\begin{aligned}
&\exists s_{Y_2} \in C(Y_2), s_{Z_2} \in E(Z_2) : \\
&((s_U, s_{Y_2}), (s_V, s_{Z_2}), (s_Y, s_{Y_2}), (s_Z, s_{Z_2})) \in S_{\mathcal{D}}.
\end{aligned}$$

Therefore, condition 2 of Definition 3.5 holds. ■

Finally, we can define the feedback connection operator for DCESs by using the construction in the previous proof.

Definition 3.9 (Discrete feedback connection) *Let $\mathcal{D}$ be a DCES as in Theorem 3.7, and let the unique solution property hold. The DCES $\mathcal{D}'$ gained from the construction above is called the* discrete feedback connection *of $\mathcal{D}$, denoted by $\mathcal{D} \hookleftarrow$.* □

3.3 The Loop Interconnection

The goal of this section is to introduce an interconnection operator which is able to express both the cascade interconnection and the feedback connection. The basic idea is to describe a mutual connection of two systems. Furthermore, each system has its own inputs and outputs for conditions and events. A block diagram of this connection, called *loop interconnection,* is shown in Figure 3.4. In the connections between the two systems, the condition and event alphabets of the connected components have to be equal, so we demand $U_{1,2} = Y_{2,2}$, $V_{1,2} = Z_{2,2}$, $U_{2,2} = Y_{1,2}$, and $V_{2,2} = Z_{1,2}$. In a less strict variant, "$\supseteq$" instead of "$=$" can be allowed in all four equations.

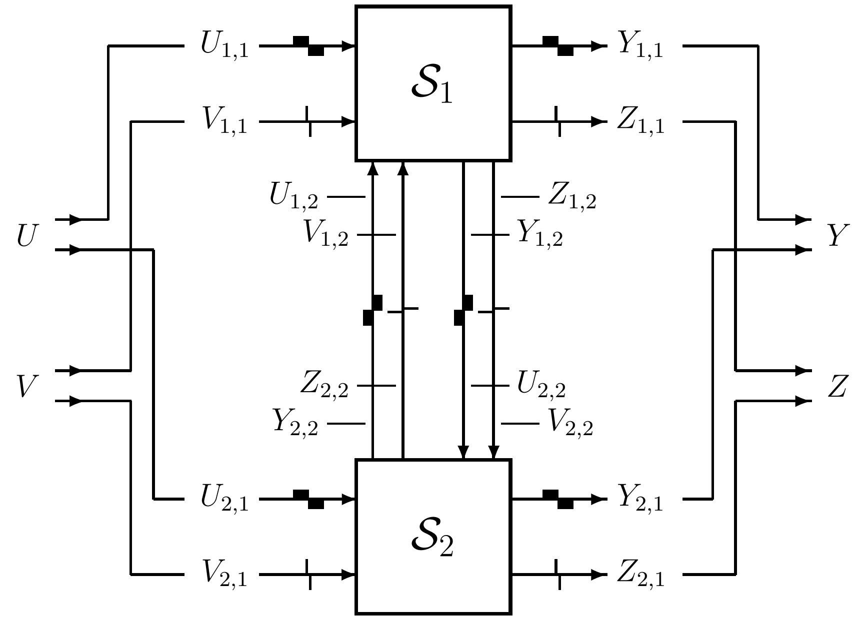


Figure 3.4: CES loop interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$.

3.3.1 The Loop Interconnection of C/E Systems

Given two CESs $\mathcal{S}_1$ and $\mathcal{S}_2$ where every alphabet is a Cartesian product of two alphabets, a cascade interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$ is a CES $\mathcal{S}$ having as input and output alphabets all the first components of $\mathcal{S}_1$'s and $\mathcal{S}_2$'s alphabets. The behaviors of $\mathcal{S}$ are composed from the first components of the signals in the behaviors of $\mathcal{S}_1$ and $\mathcal{S}_2$, and we demand that the signals of the connections between $\mathcal{S}_1$ and $\mathcal{S}_2$ (these are the second components) match. The signals between $\mathcal{S}_1$ and $\mathcal{S}_2$ are not visible in $\mathcal{S}$.

Definition 3.10 (Loop interconnection) *Let $\mathcal{S}_i = (U_i, V_i, Y_i, Z_i, S_i)$ be two C/E systems with inputs $U_i = U_{i,1} \times U_{i,2}$, $V_i = V_{i,1} \times V_{i,2}$, and outputs $Y_i = Y_{i,1} \times Y_{i,2}$, $Z_i = Z_{i,1} \times Z_{i,2}$, $i \in \{1,2\}$. Let $U_{1,2} = Y_{2,2}$, $V_{1,2} = Z_{2,2}$, $U_{2,2} = Y_{1,2}$, $V_{2,2} = Z_{1,2}$. A C/E system $\mathcal{S} = (U, V, Y, Z, S)$ is called* loop interconnection *of $\mathcal{S}_1$ and $\mathcal{S}_2$, if and only if the following conditions hold:*

1. $U = U_{1,1} \times U_{2,1}$, $V = V_{1,1} \times V_{2,1}$, $Y = Y_{1,1} \times Y_{2,1}$, $Z = Z_{1,1} \times Z_{2,1}$.
2. *For all $(s_U, s_V, s_Y, s_Z) \in \mathrm{Beh}(U, V, Y, Z)$, we have $(s_U, s_V, s_Y, s_Z) \in S$ if and only if there exist $\bar{s}_{U_i} \in C(U_{i,2})$, $\bar{s}_{V_i} \in E(V_{i,2})$, for $i \in \{1,2\}$, such that $\Big(((s_U)_i, \bar{s}_{U_i}), ((s_V)_i, \bar{s}_{V_i}), ((s_Y)_i, \bar{s}_{U_{3-i}}), ((s_Z)_i, \bar{s}_{V_{3-i}})\Big) \in S_i$, for all $i \in \{1,2\}$.* $\square$

Without proof we state in the following lemma that we cannot find for every two CESs $\mathcal{S}_1$ and $\mathcal{S}_2$ with alphabets as in Definition 3.10 a CES $\mathcal{S}$ which is a loop interconnection of $\mathcal{S}_1$ and $\mathcal{S}_2$. The reason for this lies again in algebraic loops. An example for this is Figure 1.1 in the introduction.

Lemma 3.11 *There exist CESs $\mathcal{S}_1$ and $\mathcal{S}_2$ fulfilling the syntactic requirements of Definition 3.10 having no loop interconnection.* □

3.3.2 The Loop Interconnection of Discrete C/E Systems

As for the feedback connection, we need a property that defines for two given DCESs $\mathcal{D}_1$ and $\mathcal{D}_2$ if there exists a DCES $\mathcal{D}$ with $\mathcal{S}_\mathcal{D}$ being a loop interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$. We use again a *unique solution property* to ensure that there exists a solution for the connections between $\mathcal{D}_1$ and $\mathcal{D}_2$. This time we need as solutions two condition symbols ($u_{2,2}$ for the condition connection from $\mathcal{D}_1$ to $\mathcal{D}_2$ and $u_{1,2}$ for the other direction), which depend on the states x_1 and x_2 of $\mathcal{D}_1$ and $\mathcal{D}_2$ and their external condition input symbols $u_{1,1}$ and $u_{2,1}$, and two event symbols $v_{2,2}$ and $v_{1,2}$ (one for each event connection between $\mathcal{D}_1$ and $\mathcal{D}_2$), which depend on the pre- and post-states x_1, x_1' and x_2, x_2' of $\mathcal{D}_1$ and $\mathcal{D}_2$ and their external event input symbols $v_{1,1}$ and $v_{2,1}$.

Theorem 3.12 *Given two DCES $\mathcal{D}_i = (U_i, V_i, X_i, Y_i, Z_i, f_i, g_i, h_i, x_{0,i})$ with inputs $U_i = U_{i,1} \times U_{i,2}$, $V_i = V_{i,1} \times V_{i,2}$, outputs $Y_i = Y_{i,1} \times Y_{i,2}$, $Z_i = Z_{i,1} \times Z_{i,2}$, for $i \in \{1, 2\}$, and matching alphabets $U_{1,2} = Y_{2,2}$, $V_{1,2} = Z_{2,2}$, $U_{2,2} = Y_{1,2}$, $V_{2,2} = Z_{1,2}$, a DCES $\mathcal{D}$ can be effectively constructed such that $\mathcal{S}_\mathcal{D}$ is a loop interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$, if the following* unique solution property *holds for $\mathcal{D}_1$ and $\mathcal{D}_2$:*

- *For all $x_1 \in X_1$, $x_2 \in X_2$, $u_{1,1} \in U_{1,1}$, $u_{2,1} \in U_{2,1}$, there exist unique $u_{1,2} \in U_{1,2}$, $u_{2,2} \in U_{2,2}$, such that $g_1(x_1, (u_{1,1}, u_{1,2}))_2 = u_{2,2}$ and $g_2(x_2, (u_{2,1}, u_{2,2}))_2 = u_{1,2}$.*
- *For all $x_1, x_1' \in X_1$, $x_2, x_2' \in X_2$, $v_{1,1} \in V_{1,1}$, $v_{2,1} \in V_{2,1}$, there exist unique $v_{1,2} \in V_{1,2}$, $v_{2,2} \in V_{2,2}$, such that $h_1(x_1, x_1', (v_{1,1}, v_{1,2}))_2 = v_{2,2}$ and $h_2(x_2, x_2', (v_{2,1}, v_{2,2}))_2 = v_{1,2}$.* □

In the following proof, the effective construction of $\mathcal{D}$ is given. The set of states of $\mathcal{D}$ is the Cartesian product of the state sets of $\mathcal{D}_1$ and $\mathcal{D}_2$. The functions f, g, and h of $\mathcal{D}$ are combined from f_i, g_i, and h_i, $i \in \{1, 2\}$. The parameters for these functions are taken from the parameters of f, g, and h, and from the unique solutions that have been found while checking the unique solution property.

Proof Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be given as above, let the unique solution property hold. Let $G_1(x_1, x_2, u_{1,1}, u_{2,1})$, $G_2(x_1, x_2, u_{1,1}, u_{2,1})$, $H_1(x_1, x_1', x_2, x_2', v_{1,1}, v_{2,1})$, $H_2(x_1, x_1', x_2, x_2', v_{1,1}, v_{2,1})$ be the unique solutions $u_{2,2}$, $u_{1,2}$, $v_{2,2}$, $v_{1,2}$, respectively.

We construct a DCES $\mathcal{D}$ such that $\mathcal{S}_\mathcal{D}$ is a loop interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$. Let $\mathcal{D} = (U, V, X, Y, Z, f, g, h, x_0)$ be the DCES with inputs $U = U_{1,1} \times U_{2,1}$, $V = V_{1,1} \times V_{2,1}$, outputs $Y = Y_{1,1} \times Y_{2,1}$, $Z = Z_{1,1} \times Z_{2,1}$, states $X = X_1 \times X_2$, $x_0 = (x_{0,1}, x_{0,2})$, and, for all $x = (x_1, x_2) \in X$, $x' = (x_1', x_2') \in X$, $u = (u_{1,1}, u_{2,1}) \in U$, $v = (v_{1,1}, v_{2,1}) \in V$,

- $x' \in f(x, u, v)$ if and only if $x_i' \in f_i(x_i, (u_{i,1}, u_{i,2}), (v_{i,1}, v_{i,2}))$, $i \in \{1, 2\}$,
- $g(x, u) = \Big(g_1(x_1, (u_{1,1}, u_{1,2}))_1, g_2(x_2, (u_{2,1}, u_{2,2}))_1\Big)$,
- $h(x, x', v) = \Big(h_1(x_1, x_1', (v_{1,1}, v_{1,2}))_1, h_2(x_2, x_2', (v_{2,1}, v_{2,2}))_1\Big)$,

with

- $u_{1,2} = G_2(x_1, x_2, u_{1,1}, u_{2,1})$, $u_{2,2} = G_1(x_1, x_2, u_{1,1}, u_{2,1})$,
- $v_{1,2} = H_2(x_1, x_1', x_2, x_2', v_{1,1}, v_{2,1})$, $v_{2,2} = H_1(x_1, x_1', x_2, x_2', v_{1,1}, v_{2,1})$.

It can easily be checked that $\mathcal{D}$ is well-behaved, since $\mathcal{D}_1$ and $\mathcal{D}_2$ are well-behaved and therefore, $H_1(x_1, x_1, x_2, x_2, \mathbf{0}_{V_{1,1}}, \mathbf{0}_{V_{2,1}}) = \mathbf{0}_{V_{2,2}}$ and $H_2(x_1, x_1, x_2, x_2, \mathbf{0}_{V_{1,1}}, \mathbf{0}_{V_{2,1}}) = \mathbf{0}_{V_{1,2}}$ for all $x_1 \in X_1$ and $x_2 \in X_2$.

Now we check if $\mathcal{S}_\mathcal{D}$ is a loop interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$. Condition 1 of Definition 3.10 holds obviously. For proving condition 2, let $(s_U, s_V, s_Y, s_Z) \in \mathrm{Beh}(U, V, Y, Z)$. We have to show:

$$\begin{aligned}
&(s_U, s_V, s_Y, s_Z) \in S_\mathcal{D} \text{ if and only if} \\
&\exists \bar{s}_{U_i} \in C(U_{i,2}), \bar{s}_{V_i} \in E(V_{i,2}), \text{ for } i \in \{1, 2\} : \\
&\Big(((s_U)_i, \bar{s}_{U_i}), ((s_V)_i, \bar{s}_{V_i}), ((s_Y)_i, \bar{s}_{U_{3-i}}), ((s_Z)_i, \bar{s}_{V_{3-i}})\Big) \in S_{\mathcal{D}_i},\ i \in \{1, 2\}.
\end{aligned}$$

We have

$$(s_U, s_V, s_Y, s_Z) \in S_\mathcal{D}$$

if and only if (by Definition 2.7)

$\exists r : \mathbb{R}_{\geq 0} \to X$ (right-continuous, finite-variable) :
1. $\forall t \in \mathbb{R}_{>0} : r(t) \in f(r(t^-), s_U(t^-), s_V(t))$
2. $\forall t \in \mathbb{R}_{\geq 0} : s_Y(t) = g(r(t), s_U(t))$
3. $\forall t \in \mathbb{R}_{>0} : s_Z(t) = h(r(t^-), r(t), s_V(t))$
4. $r(0) = x_0$

if and only if (by construction of the DCES $\mathcal{D}$)

$\exists r : \mathbb{R}_{\geq 0} \to X$ (right-continuous, finite-variable) :
$\forall i \in \{1, 2\}$:
1. $\forall t \in \mathbb{R}_{>0} : r(t)_i \in f_i\Big(r(t^-)_i, (s_U(t^-)_i, u_{i,2}(t^-)), (s_V(t)_i, v_{i,2}(t))\Big)$
2. $\forall t \in \mathbb{R}_{\geq 0} : s_Y(t)_i = g_i\Big(r(t)_i, (s_U(t)_i, u_{i,2}(t))\Big)_1$
3. $\forall t \in \mathbb{R}_{>0} : s_Z(t)_i = h_i\Big(r(t^-)_i, r(t)_i, (s_V(t)_i, v_{i,2}(t))\Big)_1$
4. $r(0)_i = x_{0,i}$

with the unique solutions
$u_{1,2}(t) = G_2(r(t)_1, r(t)_2, s_U(t)_1, s_U(t)_2)$
$u_{2,2}(t) = G_1(r(t)_1, r(t)_2, s_U(t)_1, s_U(t)_2)$
$v_{1,2}(t) = H_2(r(t^-)_1, r(t)_1, r(t^-)_2, r(t)_2, s_V(t)_1, s_V(t)_2)$
$v_{2,2}(t) = H_1(r(t^-)_1, r(t)_1, r(t^-)_2, r(t)_2, s_V(t)_1, s_V(t)_2)$

if and only if (since $u_{1,2}$, $u_{2,2}$, $v_{1,2}$, $v_{2,2}$ are unique, they can be replaced by the existentially quantified $\bar{s}_{U_1}$, $\bar{s}_{U_2}$, $\bar{s}_{V_1}$, $\bar{s}_{V_2}$, respectively)

$\exists \bar{s}_{U_i} \in C(U_{i,2}), \bar{s}_{V_i} \in E(V_{i,2}),$ for all $i \in \{1, 2\}$:
$\exists r : \mathbb{R}_{\geq 0} \to X$ (right-continuous, finite-variable) :
$\forall i \in \{1, 2\}$:
1. $\forall t \in \mathbb{R}_{>0} : r(t)_i \in f_i\Big(r(t^-)_i, (s_U(t^-)_i, \bar{s}_{U_i}(t^-)), (s_V(t)_i, \bar{s}_{V_i}(t))\Big)$
2. $\forall t \in \mathbb{R}_{\geq 0} : s_Y(t)_i = g_i\Big(r(t)_i, (s_U(t)_i, \bar{s}_{U_i}(t))\Big)_1$
3. $\forall t \in \mathbb{R}_{>0} : s_Z(t)_i = h_i\Big(r(t^-)_i, r(t)_i, (s_V(t)_i, \bar{s}_{V_i}(t))\Big)_1$
4. $r(0)_i = x_{0,i}$

with
$\bar{s}_{U_1}(t) = G_2(r(t)_1, r(t)_2, s_U(t)_1, s_U(t)_2)$
$\bar{s}_{U_2}(t) = G_1(r(t)_1, r(t)_2, s_U(t)_1, s_U(t)_2)$
$\bar{s}_{V_1}(t) = H_2(r(t^-)_1, r(t)_1, r(t^-)_2, r(t)_2, s_V(t)_1, s_V(t)_2)$
$\bar{s}_{V_2}(t) = H_1(r(t^-)_1, r(t)_1, r(t^-)_2, r(t)_2, s_V(t)_1, s_V(t)_2)$

if and only if (the unique solution property completely determines the behavior of the second components of g_i and h_i)

$$\begin{aligned}
&\exists \bar{s}_{U_i} \in C(U_{i,2}), \bar{s}_{V_i} \in E(V_{i,2}), \text{ for all } i \in \{1,2\}:\\
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)}:\\
&\forall i \in \{1,2\}:\\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t)_i \in f_i\Big(r(t^-)_i, (s_U(t^-)_i, \bar{s}_{U_i}(t^-)), (s_V(t)_i, \bar{s}_{V_i}(t))\Big)\\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : (s_Y(t)_i, \bar{s}_{U_{3-i}}(t)) = g_i\Big(r(t)_i, (s_U(t)_i, \bar{s}_{U_i}(t))\Big)\\
&3.\ \forall t \in \mathbb{R}_{>0} : (s_Z(t)_i, \bar{s}_{V_{3-i}}(t)) = h_i\Big(r(t^-)_i, r(t)_i, (s_V(t)_i, \bar{s}_{V_i}(t))\Big)\\
&4.\ r(0)_i = x_{0,i}
\end{aligned}$$

if and only if (splitting r into r_1 and r_2)

$$\begin{aligned}
&\exists \bar{s}_{U_i} \in C(U_{i,2}), \bar{s}_{V_i} \in E(V_{i,2}), \text{ for all } i \in \{1,2\}:\\
&\forall i \in \{1,2\} : \exists r_i : \mathbb{R}_{\geq 0} \to X_i \text{ (right-continuous, finite-variable)}:\\
&1.\ \forall t \in \mathbb{R}_{>0} : r_i(t) \in f_i\Big(r_i(t^-), (s_U(t^-)_i, \bar{s}_{U_i}(t^-)), (s_V(t)_i, \bar{s}_{V_i}(t))\Big)\\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : (s_Y(t)_i, \bar{s}_{U_{3-i}}(t)) = g_i\Big(r_i(t), (s_U(t)_i, \bar{s}_{U_i}(t))\Big)\\
&3.\ \forall t \in \mathbb{R}_{>0} : (s_Z(t)_i, \bar{s}_{V_{3-i}}(t)) = h_i\Big(r_i(t^-), r_i(t), (s_V(t)_i, \bar{s}_{V_i}(t))\Big)\\
&4.\ r_i(0) = x_{0,i}
\end{aligned}$$

if and only if (by Definition 2.7)

$$\begin{aligned}
&\exists \bar{s}_{U_i} \in C(U_{i,2}), \bar{s}_{V_i} \in E(V_{i,2}), \text{ for } i \in \{1,2\}:\\
&\Big(((s_U)_i, \bar{s}_{U_i}), ((s_V)_i, \bar{s}_{V_i}), ((s_Y)_i, \bar{s}_{U_{3-i}}), ((s_Z)_i, \bar{s}_{V_{3-i}})\Big) \in S_{\mathcal{D}_i},\ i \in \{1,2\}.
\end{aligned}$$

Therefore, condition 2 of Definition 3.10 holds. ■

Now we can define the loop interconnection operator for DCESs by using the construction in the proof above.

Definition 3.13 (Discrete loop interconnection) *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two DCESs as in Theorem 3.12, and let the unique solution property hold. The DCES $\mathcal{D}$ gained from the construction above is called the* discrete loop interconnection *of $\mathcal{D}_1$ and $\mathcal{D}_2$, denoted by $\mathcal{D}_1 \leftrightarrow \mathcal{D}_2$.* □

3.3.3 Properties of the Discrete Loop Interconnection

An interesting observation is that the unique solution property of Theorem 3.12 is independent from the transition functions of $\mathcal{D}_1$ and $\mathcal{D}_2$, and that the two output functions that are constructed for the discrete loop interconnection of $\mathcal{D}_1$ and $\mathcal{D}_2$ also does not need the transition functions.

Remark 3.14 The unique solution property for the discrete loop interconnection of the DCESs $\mathcal{D}_1$ and $\mathcal{D}_2$ only depends on the condition and event output functions g_i and h_i of $\mathcal{D}_i$, $i \in \{1,2\}$, and these functions are sufficient to effectively construct the condition/event output functions of $\mathcal{D}_1 \leftrightarrow \mathcal{D}_2$. $\diamond$

This observation is useful when checking a large system consisting of several discrete loop interconnections for well-definedness. The following lemma shows this for a connection of three DCESs.

Lemma 3.15 *Given three DCESs $\mathcal{D}_i = (U_i, V_i, X_i, Y_i, Z_i, f_i, g_i, h_i, x_{0,i})$ with inputs $U_i = U_{i,1} \times U_{i,2}$, $V_i = V_{i,1} \times V_{i,2}$, outputs $Y_i = Y_{i,1} \times Y_{i,2}$, $Z_i = Z_{i,1} \times Z_{i,2}$, for $i \in \{1,2,3\}$, and matching alphabets $U_{1,2} = Y_{2,2}$, $V_{1,2} = Z_{2,2}$, $U_{2,2} = Y_{1,2}$, $V_{2,2} = Z_{1,2}$, and $U_{2,1} = Y_{3,2}$, $V_{2,1} = Z_{3,2}$, $U_{3,2} = Y_{2,1}$, $V_{3,2} = Z_{2,1}$, the well-definedness of $(\mathcal{D}_1 \leftrightarrow \mathcal{D}_2) \leftrightarrow \mathcal{D}_3$ only depends on g_i and h_i, $i \in \{1,2,3\}$, and these functions are sufficient to effectively construct the condition/event output functions of $(\mathcal{D}_1 \leftrightarrow \mathcal{D}_2) \leftrightarrow \mathcal{D}_3$.* $\square$

Proof Let $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$ be given as above. Note that the well-definedness reduces to the unique solution property, since the alphabets are chosen to fit to the loop constructions. For checking the well-definedness of $(\mathcal{D}_1 \leftrightarrow \mathcal{D}_2) \leftrightarrow \mathcal{D}_3$, we first check the well-definedness of $\mathcal{D}_1 \leftrightarrow \mathcal{D}_2$. By Remark 3.14, this can be done using only g_1, h_1, g_2, h_2. If $\mathcal{D}_1 \leftrightarrow \mathcal{D}_2$ is well-defined, we also get from Remark 3.14 that the output functions g_{12} and h_{12} of $\mathcal{D}_{12} = \mathcal{D}_1 \leftrightarrow \mathcal{D}_2$ can be effectively constructed from g_1, h_1, g_2, h_2. Using Remark 3.14 again, checking the well-definedness of $\mathcal{D}_{12} \leftrightarrow \mathcal{D}_3$ can be done by using the functions g_{12}, h_{12}, g_3, h_3 only.

Now, let $(\mathcal{D}_1 \leftrightarrow \mathcal{D}_2) \leftrightarrow \mathcal{D}_3$ be well-defined. We give a direct construction for condition/event output functions g_{123} and h_{123} identical to those of $\mathcal{D}_{123} = (\mathcal{D}_1 \leftrightarrow \mathcal{D}_2) \leftrightarrow \mathcal{D}_3$ which only uses g_i and h_i, $i \in \{1,2,3\}$.

Since the unique solution property holds for $\mathcal{D}_1$ and $\mathcal{D}_2$, we find for all $x = (x_1, x_2, x_3) \in X_1 \times X_2 \times X_3$, $x' = (x'_1, x'_2, x'_3) \in X_1 \times X_2 \times X_3$, $u_1 \in U_{1,1}$, $v_1 \in V_{1,1}$, $u_2 \in U_{3,1}$, $v_2 \in V_{3,1}$ unique solutions $G_1(x, u_1, u_2)$, $H_1(x, x', v_1, v_2)$, and, for all $j \in \{1,2\}$, $G_{2,j}(x, u_1, u_2)$, $H_{2,j}(x, x', v_1, v_2)$, and $G_3(x, u_1, u_2)$, $H_3(x, x', v_1, v_2)$, such that

- $G_1(x, u_1, u_2) = g_1\Big(x_1, (u_1, G_{2,2}(x, u_1, u_2))\Big)_2$,
- $H_1(x, x', v_1, v_2) = h_1\Big(x_1, x'_1, (v_1, H_{2,2}(x, x', v_1, v_2))\Big)_2$,
- $G_{2,j}(x, u_1, u_2) = g_2\Big(x_2, (G_3(x, u_1, u_2), G_1(x, u_1, u_2))\Big)_j$,
- $H_{2,j}(x, x', v_1, v_2) = h_2\Big(x_2, x'_2, (H_3(x, x', v_1, v_2), H_1(x, x', v_1, v_2))\Big)_j$,
- $G_3(x, u_1, u_2) = g_3\Big(x_3, (u_2, G_{2,1}(x, u_1, u_2))\Big)_2$,
- $H_3(x, x', v_1, v_2) = h_3\Big(x_3, x'_3, (v_2, H_{2,1}(x, x', v_1, v_2))\Big)_2$.

In the following, we do not distinguish between $X_1 \times X_2 \times X_3$ and $(X_1 \times X_2) \times X_3$ for the state space of $\mathcal{D}_{123}$. Let g_{123} and h_{123} be defined by

- $g_{123}(x, (u_1, u_2))_1 = g_1\Big(x_1, (u_1, G_{2,2}(x, u_1, u_2))\Big)_1$,
- $g_{123}(x, (u_1, u_2))_2 = g_3\Big(x_3, (u_2, G_{2,1}(x, u_1, u_2))\Big)_1$,
- $h_{123}(x, (u_1, u_2))_1 = h_1\Big(x_1, x'_1, (v_1, H_{2,2}(x, x', v_1, v_2))\Big)_1$,
- $h_{123}(x, (u_1, u_2))_2 = h_3\Big(x_3, x'_3, (v_2, H_{2,1}(x, x', v_1, v_2))\Big)_1$,

for all $x = (x_1, x_2, x_3) \in X_1 \times X_2 \times X_3$, $x' = (x'_1, x'_2, x'_3) \in X_1 \times X_2 \times X_3$, $u_1 \in U_{1,1}$, $v_1 \in V_{1,1}$, $u_2 \in U_{3,1}$, $v_2 \in V_{3,1}$.

Let g and h be the condition/event output functions of $\mathcal{D}_{123}$. We prove $g_{123} = g$ and $h_{123} = h$. Following Definition 3.13, and using G_1, H_1, $G_{2,1}$, $G_{2,2}$, $H_{2,1}$, $H_{2,2}$, G_3, and H_3 as the unique solutions in the loop construction, we get for the condition output function

$$
\begin{aligned}
g(x, (u_1, u_2))_1 &= g_{12}\Big((x_1, x_2), (u_1, G_3(x, u_1, u_2))\Big)_1 \\
&= g_1\Big(x_1, (u_1, G_{2,2}(x, u_1, u_2))\Big)_1 \\
&= g_{123}(x, (u_1, u_2))_1, \text{ and} \\
g(x, (u_1, u_2))_2 &= g_3\Big(x_3, (u_2, G_{2,1}(x, u_1, u_2))\Big)_1 \\
&= g_{123}(x, (u_1, u_2))_2,
\end{aligned}
$$

and for the event output function

$$\begin{aligned}
h(x, x', (v_1, v_2))_1 &= h_{12}\Big((x_1, x_2), (x'_1, x'_2), (v_1, G_3(x, x', v_1, v_2))\Big)_1 \\
&= h_1\Big(x_1, x'_1, (v_1, G_{2,2}(x, x', v_1, v_2))\Big)_1 \\
&= h_{123}(x, (v_1, v_2))_1, \text{ and} \\
h(x, x', (v_1, v_2))_2 &= h_3\Big(x_3, x'_3, (v_2, G_{2,1}(x, x', v_1, v_2))\Big)_1 \\
&= h_{123}(x, x', (v_1, v_2))_2,
\end{aligned}$$

for all $x = (x_1, x_2, x_3) \in X_1 \times X_2 \times X_3$, $x' = (x'_1, x'_2, x'_3) \in X_1 \times X_2 \times X_3$, $u_1 \in U_{1,1}$, $v_1 \in V_{1,1}$, $u_2 \in U_{3,1}$, $v_2 \in V_{3,1}$. ■

3.3.4 Special Cases of the Discrete Loop Interconnection

Now we show that the discrete loop interconnection operator is able to express the discrete cascade interconnection and the discrete feedback connection.

Theorem 3.16 *The discrete cascade interconnection is a special case (modulo some syntactic modifications) of the discrete loop interconnection.* □

Proof Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be given as in Theorem 3.3. First, we add some trivial input/output components to $\mathcal{D}_1$ and $\mathcal{D}_2$, making them satisfy the syntactic requirements of Theorem 3.12: For $i \in \{1, 2\}$, let $\bar{\mathcal{D}}_i = (\bar{U}_i, \bar{V}_i, X_i, \bar{Y}_i, \bar{Z}_i, \bar{f}_i, \bar{g}_i, \bar{h}_i, x_{0,i})$ be the DCES with

- $\bar{U}_1 = U_1 \times C_t$, $\bar{V}_1 = V_1 \times E_t$, $\bar{Y}_1 = C_t \times Y_1$, $\bar{Z}_1 = E_t \times Z_1$, and $\bar{U}_2 = C_t \times U_2$, $\bar{V}_2 = E_t \times V_2$, $\bar{Y}_2 = Y_2 \times C_t$, $\bar{Z}_2 = Z_2 \times E_t$, where $C_t = \{0\}$ is a trivial condition alphabet, and $E_t = \{\mathbf{0}\}$ is the trivial event alphabet.
- For all $x_1, x'_1 \in X_1$, $x_2, x'_2 \in X_2$, $u_1 \in U_1$, $u_2 \in U_2$, $v_1 \in V_1$, $v_2 \in V_2$,
  - $x'_1 \in \bar{f}_1(x_1, (u_1, 0), (v_1, \mathbf{0}))$ if and only if $x'_1 \in f_1(x_1, u_1, v_1)$,
  - $x'_2 \in \bar{f}_2(x_2, (0, u_2), (\mathbf{0}, v_2))$ if and only if $x'_2 \in f_2(x_2, u_2, v_2)$,
  - $\bar{g}_1(x_1, (u_1, 0)) = (0, g_1(x_1, u_1))$,
  - $\bar{g}_2(x_2, (0, u_2)) = (g_2(x_2, u_2), 0)$,
  - $\bar{h}_1(x_1, x'_1, (v_1, \mathbf{0})) = (\mathbf{0}, h_1(x_1, x'_1, v_1))$,
  - $\bar{h}_2(x_2, x'_2, (\mathbf{0}, v_2)) = (h_2(x_2, x'_2, v_2), \mathbf{0})$.

We check the unique solution property for $\bar{\mathcal{D}}_1$ and $\bar{\mathcal{D}}_2$: The second components of $\bar{g}_2$ and $\bar{h}_2$ are constant (i. e. there is only trivial feedback from $\bar{\mathcal{D}}_2$ to $\bar{\mathcal{D}}_1$), so it is easy to see that $G_1(x_1, x_2, u_1, 0) = g_1(x_1, u_1)$, $G_2(x_1, x_2, u_1, 0) = 0$ and $H_1(x_1, x_1', x_2, x_2', v_1, \mathbf{0}) = h_1(x_1, x_1', v_1)$, $H_2(x_1, x_1', x_2, x_2', v_1, \mathbf{0}) = \mathbf{0}$ are the unique solutions.

By using Theorem 3.12, we can effectively construct the discrete loop interconnection $\bar{\mathcal{D}} = \bar{\mathcal{D}}_1 \leftrightarrow \bar{\mathcal{D}}_2$. By projecting the input and output alphabets of $\bar{\mathcal{D}} = (U_1 \times C_t, V_1 \times E_t, X, C_t \times Y_2, E_t \times Z_2, \bar{f}, \bar{g}, \bar{h}, (x_{0,1}, x_{0,2}))$ to its non-trivial components, we get the DCES $\mathcal{D} = (U_1, V_1, X, Y_2, Z_2, f, g, h, (x_{0,1}, x_{0,2}))$.

Now we check if $\mathcal{S}_{\mathcal{D}}$ is a cascade interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$. Condition 1 of Definition 3.1 holds obviously. For proving condition 2, let $s \in \mathrm{Beh}(U, V, Y, Z)$. We have to show that $s \in S_{\mathcal{D}}$ if and only if there exists $(s_U', s_V') \in S_{\mathcal{D}_1}(s_U, s_V)$ such that $(s_Y, s_Z) \in S_{\mathcal{D}_2}(s_U', s_V')$. Let $s_{U_0} \in C(C_t)$, $s_{V_0} \in E(E_t)$. Obviously, s_{U_0} and s_{V_0} are unique and constant. We have

$$(s_U, s_V, s_Y, s_Z) \in S_{\mathcal{D}}$$

if and only if ($S_{\mathcal{D}}$ is the projection of $S_{\bar{\mathcal{D}}}$ to its non-trivial components)

$$((s_U, s_{U_0}), (s_V, s_{V_0}), (s_{U_0}, s_Y), (s_{V_0}, s_Z)) \in S_{\bar{\mathcal{D}}}$$

if and only if (by Theorem 3.12 and Definition 3.10)

$$\begin{aligned}
&\exists \bar{s}_{U_1} \in C(C_t), \bar{s}_{V_1} \in E(E_t), \bar{s}_{U_2} \in C(U_2), \bar{s}_{V_2} \in E(V_2): \\
&((s_U, \bar{s}_{U_1}), (s_V, \bar{s}_{V_1}), (s_{U_0}, \bar{s}_{U_2}), (s_{V_0}, \bar{s}_{V_2})) \in S_{\bar{\mathcal{D}}_1}, \\
&((s_{U_0}, \bar{s}_{U_2}), (s_{V_0}, \bar{s}_{V_2}), (s_Y, \bar{s}_{U_1}), (s_Z, \bar{s}_{V_1})) \in S_{\bar{\mathcal{D}}_2}
\end{aligned}$$

if and only if ($\bar{s}_{U_1}$ and $\bar{s}_{V_1}$ are unique and constant, and $\bar{s}_{U_1} = s_{U_0}$, $\bar{s}_{V_1} = s_{V_0}$)

$$\begin{aligned}
&\exists \bar{s}_{U_2} \in C(U_2), \bar{s}_{V_2} \in E(V_2): \\
&((s_U, s_{U_0}), (s_V, s_{V_0}), (s_{U_0}, \bar{s}_{U_2}), (s_{V_0}, \bar{s}_{V_2})) \in S_{\bar{\mathcal{D}}_1}, \\
&((s_{U_0}, \bar{s}_{U_2}), (s_{V_0}, \bar{s}_{V_2}), (s_Y, s_{U_0}), (s_Z, s_{V_0})) \in S_{\bar{\mathcal{D}}_2}
\end{aligned}$$

if and only if ($S_{\mathcal{D}_1}$ and $S_{\mathcal{D}_2}$ are the projections of $S_{\bar{\mathcal{D}}_1}$ and $S_{\bar{\mathcal{D}}_2}$ to their non-trivial components)

$$\begin{aligned}
&\exists \bar{s}_{U_2} \in C(U_2), \bar{s}_{V_2} \in E(V_2): \\
&(s_U, s_V, \bar{s}_{U_2}, \bar{s}_{V_2}) \in S_{\bar{\mathcal{D}}_1}, \\
&(\bar{s}_{U_2}, \bar{s}_{V_2}, s_Y, s_Z) \in S_{\bar{\mathcal{D}}_2}
\end{aligned}$$

if and only if

$$\exists (s_u', s_v') \in S_{\bar{\mathcal{D}}_1}(s_U, s_V) : (s_Y, s_Z) \in S_{\bar{\mathcal{D}}_2}(s_u', s_v'). \qquad \blacksquare$$

Also the discrete feedback connection is a special case of the discrete loop interconnection. Since feedback involves only one system, we have to add a second system in the proof of the following theorem.

Theorem 3.17 *The discrete feedback connection is a special case (modulo some syntactic modifications) of the discrete loop interconnection.* □

Proof Let $\mathcal{D}$ be given as in Theorem 3.7, and let its unique solution property hold. To model the feedback connection as a loop interconnection, we introduce a second DCES $\mathcal{D}_{\mathsf{id}}$.

Let $\mathcal{D}_{\mathsf{id}} = (C_t \times U_2, E_t \times V_2, \{x_{\mathsf{id}}\}, C_t \times U_2, E_t \times V_2, f_{\mathsf{id}}, g_{\mathsf{id}}, h_{\mathsf{id}}, x_{\mathsf{id}})$ be the DCES with

- $f_{\mathsf{id}}(x_{\mathsf{id}}, u, v) = \{x_{\mathsf{id}}\}$,
- $g_{\mathsf{id}}(x_{\mathsf{id}}, u) = u$, and
- $h_{\mathsf{id}}(x_{\mathsf{id}}, x_{\mathsf{id}}, v) = v$,

for all $u \in C_t \times U_2, v \in E_t \times V_2$, and $C_t = \{0\}, E_t = \{\mathbf{0}\}$ being trivial condition/event alphabets. Obviously, $\mathcal{D}_{\mathsf{id}}$ just copies its input signals to its output signals. So the semantics of $\mathcal{D}_{\mathsf{id}}$ is

$$S_{\mathcal{D}_{\mathsf{id}}} = \{(s_U, s_V, s_U, s_V) \mid s_U \in C(C_t \times U_2), s_V \in E(E_t \times V_2)\}.$$

We check the well-definedness of the loop interconnection $\mathcal{D} \leftrightarrow \mathcal{D}_{\mathsf{id}}$, depicted in Figure 3.5 (compare this with Figures 3.3 and 3.4). It is easy to see that the alphabets match correctly (note that $Y_2 = U_2$ and $Z_2 = V_2$). The unique solution property for $\mathcal{D}$ and $\mathcal{D}_{\mathsf{id}}$ also holds: Since $\mathcal{D}_{\mathsf{id}}$ copies from input to output, the unique solutions G_1, G_2, H_1, H_2 for $\mathcal{D}$ and $\mathcal{D}_{\mathsf{id}}$ are the same as the unique solutions G and H we get from the feedback connection of $\mathcal{D}$.

By using Theorem 3.12, we get the DCES $\bar{\mathcal{D}} = \mathcal{D} \leftrightarrow \mathcal{D}_{\mathsf{id}}$. Projecting the input and output alphabets of $\bar{\mathcal{D}}$ to its non-trivial first components, we get the DCES $\mathcal{D}' = (U_1, V_1, X', Y_1, Z_1, f', g', h', x'_0)$.

We check if $\mathcal{S}_{\mathcal{D}'}$ is a feedback connection of $\mathcal{S}_{\mathcal{D}}$. Condition 1 of Definition 3.5 holds obviously. For proving condition 2, let $(s_U, s_V, s_Y, s_Z) \in \mathrm{Beh}(U_1, V_1, Y_1, Z_1)$. We have to show that $(s_U, s_V, s_Y, s_Z) \in S_{\mathcal{D}'}$ if and only if there exist $s_{Y_2} \in C(Y_2)$ and $s_{Z_2} \in E(Z_2)$ such that $((s_U, s_{Y_2}), (s_V, s_{Z_2}), (s_Y, s_{Y_2}), (s_Z, s_{Z_2})) \in S_{\mathcal{D}}$.

Let $s_{U_0} \in C(C_t)$, $s_{V_0} \in E(E_t)$. Obviously, s_{U_0} and s_{V_0} are unique and constant. We have

$$(s_U, s_V, s_Y, s_Z) \in S_{\mathcal{D}'}$$

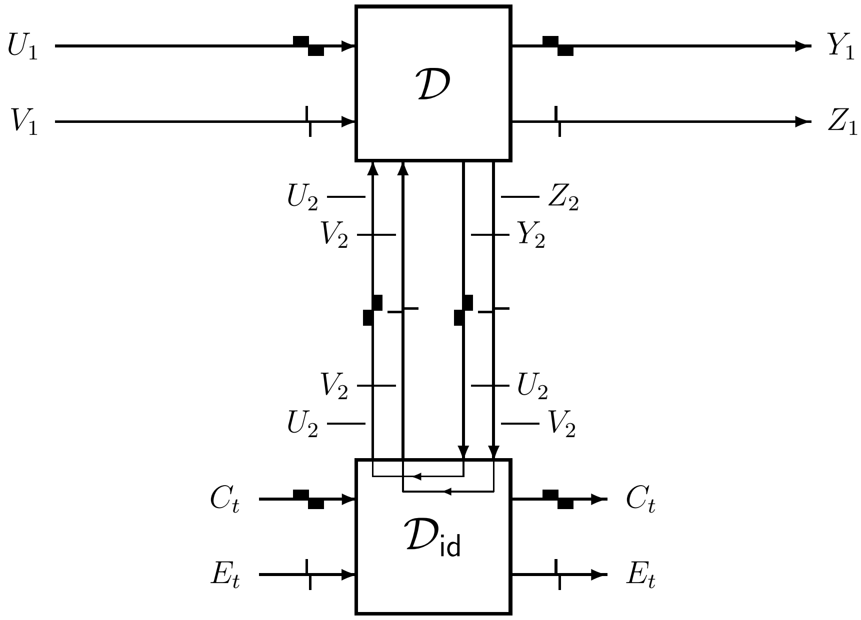


Figure 3.5: Modeling feedback of $\mathcal{D}$ by loop using the system $\mathcal{D}_{\mathsf{id}}$.

if and only if ($S_{\mathcal{D}'}$ is the projection of $S_{\bar{\mathcal{D}}}$ to its non-trivial components)

$$((s_U, s_{U_0}), (s_V, s_{V_0}), (s_Y, s_{U_0}), (s_Z, s_{V_0})) \in S_{\bar{\mathcal{D}}}$$

if and only if (by Theorem 3.12 and Definition 3.10)

$$\begin{aligned}
&\exists \bar{s}_{U_1} \in C(U_2), \bar{s}_{V_1} \in E(V_2), \bar{s}_{U_2} \in C(U_2), \bar{s}_{V_2} \in E(V_2) : \\
&((s_U, \bar{s}_{U_1}), (s_V, \bar{s}_{V_1}), (s_Y, \bar{s}_{U_2}), (s_Z, \bar{s}_{V_2})) \in S_{\mathcal{D}}, \\
&((s_{U_0}, \bar{s}_{U_2}), (s_{V_0}, \bar{s}_{V_2}), (s_{U_0}, \bar{s}_{U_1}), (s_{V_0}, \bar{s}_{V_1})) \in S_{\mathcal{D}_{\mathsf{id}}}
\end{aligned}$$

if and only if (from $S_{\mathcal{D}_{\mathsf{id}}}$, we conclude $\bar{s}_{U_2} = \bar{s}_{U_1}$ and $\bar{s}_{V_2} = \bar{s}_{V_1}$; we replace them by s_{Y_2} and s_{Z_2})

$$\begin{aligned}
&\exists s_{Y_2} \in C(U_2), s_{Z_2} \in E(V_2) : \\
&((s_U, s_{Y_2}), (s_V, s_{Z_2}), (s_Y, s_{Y_2}), (s_Z, s_{Z_2})) \in S_{\mathcal{D}}, \\
&((s_{U_0}, s_{Y_2}), (s_{V_0}, s_{Z_2}), (s_{U_0}, s_{Y_2}), (s_{V_0}, s_{Z_2})) \in S_{\mathcal{D}_{\mathsf{id}}}
\end{aligned}$$

if and only if (the last line applies for all s_{Y_2} and s_{Z_2} and can therefore be removed)

$$\begin{aligned}
&\exists s_{Y_2} \in C(U_2), s_{Z_2} \in E(V_2) : \\
&((s_U, s_{Y_2}), (s_V, s_{Z_2}), (s_Y, s_{Y_2}), (s_Z, s_{Z_2})) \in S_{\mathcal{D}}.
\end{aligned}$$

■

Now we have reached our goal to present an operator for composing DCESs which is able to express both the cascade interconnection and the feedback connection.

We have seen, however, that the use of these operators involves much work in making different systems syntactically compatible, i.e., we had to introduce new signals over the trivial alphabets $C_t = \{0\}$ and $E_t = \{\mathbf{0}\}$; even new systems like $\mathcal{D}_{\mathsf{id}}$ had to be created. The following chapter shows how to get rid of this "syntactic sugar".

Chapter 4

The Parallel Interconnection

The loop interconnection is the most universal operator for combining condition/event systems we have seen so far. But it has some disadvantages: First, the syntactic requirements for the systems are very strong, as we must have pairs of matching input and output components, even if in some directions we do not have any information to be passed. In that case, we have to add some trivial things, as we can see in the proofs of Theorem 3.16 and Theorem 3.17.

Another disadvantage is that for DCESs $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, the two terms $(\mathcal{D}_1 \leftrightarrow \mathcal{D}_2) \leftrightarrow \mathcal{D}_3$ and $\mathcal{D}_1 \leftrightarrow (\mathcal{D}_2 \leftrightarrow \mathcal{D}_3)$ describe completely different systems. Even if we make adequate syntactic changes to the systems to combine the same components in each expression, one of them may be well-defined, and the other may not, due to different unique solution properties that have to be fulfilled.

This shows that we need a different, more flexible way to describe connections between condition/event systems, and an interconnection operator that is "more distributive" than the loop interconnection. The solution we present is the *parallel interconnection,* which describes an almost arbitrary connection of CESs, and, by defining the *parallel interconnection operator,* we are able to do arbitrary connections among DCESs (there are some sensible syntactic restrictions, though).

Connections are described by assigning *names* to the input and output components of all systems. Components which share the same name are connected. An intuitive semantics for a system of CESs connected in such a way is obvious; For every name occurring in the system, one signal has to be provided. If a name is the name of an output component of a CES, this CES provides the signal for that name, and this signal is used for all input components with that name. If a name is used for input components only, the signal for that name must be an external input signal for the system.

The output signals of the system are the output signals of all CESs. This parallel interconnection of CESs will be given formally in Definition 4.10.

The notion of names makes new definitions for syntax and semantics of CESs and DCESs necessary, but the flexibility we gain for constructing interconnected systems is worth the effort.

4.1 Adding Component Names to C/E Systems

In the previous sections, we described a condition/event system's inputs and outputs by Cartesian products of condition or event alphabets. We accessed single components by projecting elements of these products on a certain index. To make dealing with many components easier and to provide a natural way of connecting systems, we change the definition of CESs in the following way: All input and output components of a CES are identified by names. Each name is associated with one condition or event alphabet. Each alphabet may be associated with more than one name. Components with the same names are considered to be connected later on.

The following definitions put these considerations on a formal basis. From now on, let $\mathcal{N}$ be a global set of identifiers, called *names.* We associate with each name $\mathsf{N} \in \mathcal{N}$ either a condition or an event alphabet, denoted by $\alpha(\mathsf{N})$. We call N *condition name* (respectively *event name*), if $\alpha(\mathsf{N})$ is a condition alphabet (respectively event alphabet).

In Chapter 3, we used Cartesian products of alphabets like $U_1 \times U_2$ if a CES or DCES needed to have two independent alphabets in one of its signals. With the new notion of names, we simply use a set of names for such purposes, like for the case above the set $\bar{\mathsf{U}} = \{\mathsf{U}_1, \mathsf{U}_2\}$ with $\alpha(\mathsf{U}_1) = U_1$ and $\alpha(\mathsf{U}_2) = U_2$. We use symbols like U, V, Y, Z for names, and symbols like $\bar{\mathsf{U}}$, $\bar{\mathsf{V}}$, $\bar{\mathsf{Y}}$, $\bar{\mathsf{Z}}$ for sets of names.

For describing the current value of a system's component named with the set $\bar{\mathsf{A}}$, we introduce the notion of an *evaluation,* which is a function mapping the names of $\bar{\mathsf{A}}$ to the condition or event symbols in the alphabets associated with the names in $\bar{\mathsf{A}}$.

Definition 4.1 (Evaluation) *For a set $\bar{\mathsf{A}} \subseteq \mathcal{N}$, we define*

$$\Sigma_{\bar{\mathsf{A}}} = [\bar{\mathsf{A}} \xrightarrow{\text{type}} \bigcup_{\mathsf{N} \in \bar{\mathsf{A}}} \alpha(\mathsf{N})]$$

as the set of all type-preserving functions φ mapping names to conditions or *events, where type-preserving in this case means* $\varphi(\mathsf{N}) \in \alpha(\mathsf{N})$ *for all* $\mathsf{N} \in \bar{\mathsf{A}}$. *Such a function* φ *is called* evaluation *of* $\bar{\mathsf{A}}$. □

Note that there is only one evaluation for the empty set of names, and this is the function $\emptyset$ having an empty domain, so we have $\Sigma_\emptyset = \{\emptyset\}$. If $\varphi(\mathsf{N}) = \mathbf{0}_{\alpha(\mathsf{N})}$ for all $\mathsf{N} \in \bar{\mathsf{A}}$, we use the symbol $\mathbf{0}_{\bar{\mathsf{A}}}$ (or simply $\mathbf{0}$) for φ. For $\mathsf{N} \in \mathcal{N}$, we define Σ_N as a shortcut for $\Sigma_{\{\mathsf{N}\}}$. Furthermore, we define some operations on evaluations.

Definition 4.2 (Operations on evaluations) *For evaluations* φ *and* ψ, *we define the following operations:*

- union: *If* φ *and* ψ *coincide on the intersection of their domains,* $\varphi \cup \psi$ *denotes their union.*
- restriction: *For* $\bar{\mathsf{A}} \subseteq \mathrm{dom}(\varphi)$, *the restriction of* φ *on* $\bar{\mathsf{A}}$ *is denoted by* $\varphi|_{\bar{\mathsf{A}}}$. *For the sake of convenient notation, we allow* $\bar{\mathsf{A}} = \emptyset$, *resulting in* $\varphi|_{\bar{\mathsf{A}}} = \emptyset$. *For* $\mathsf{N} \in \mathcal{N}$, $\varphi|_\mathsf{N}$ *is an abbreviation for* $\varphi|_{\{\mathsf{N}\}}$. □

These operations are extended in the obvious way to named signals and named behaviors, which are defined below.

4.1.1 Named C/E Systems

We use the evaluations introduced above to define *named condition/event systems*, which operate on *named condition/event signals.*

Definition 4.3 (Named condition/event signal) *Let* $\bar{\mathsf{U}} \subseteq \mathcal{N}$ *be a finite set of condition names. A function* $s_{\bar{\mathsf{U}}} : \mathbb{R}_{\geq 0} \to \Sigma_{\bar{\mathsf{U}}}$ *is called* named condition signal *over* $\bar{\mathsf{U}}$, *if it is right-continuous and finite-variable. The set of all named condition signals over* $\bar{\mathsf{U}}$ *is denoted by* $C(\bar{\mathsf{U}})$.

Let $\bar{\mathsf{V}} \subseteq \mathcal{N}$ *be a finite set of event names. A function* $s_{\bar{\mathsf{V}}} : \mathbb{R}_{\geq 0} \to \Sigma_{\bar{\mathsf{V}}}$ *is called* named event signal *over* $\bar{\mathsf{V}}$, *if* $s_{\bar{\mathsf{V}}}(0) = \mathbf{0}$ *and in every bounded subinterval* I *of* $\mathbb{R}_{\geq 0}$ *there are only finitely many points* $t \in I$ *with* $s_{\bar{\mathsf{V}}}(t) \neq \mathbf{0}$. *The set of all named event signals over* $\bar{\mathsf{V}}$ *is denoted by* $E(\bar{\mathsf{V}})$. □

For the signal over an empty set of names we use the symbol $\emptyset$, denoting the function $s_\emptyset : \mathbb{R}_{\geq 0} \to \Sigma_\emptyset$ with constant value $\emptyset$.

Now we are ready to define named condition/event systems. We ensure that each name in $\mathcal{N}$ is used at most once in one system, since we do not want a system to become connected to itself (feedback will only be allowed by using a second system, as we will see later on).

Definition 4.4 (Named condition/event system) *Let* $\bar{\mathsf{U}}$ *and* $\bar{\mathsf{Y}}$ *be finite sets of condition names, and let* $\bar{\mathsf{V}}$ *and* $\bar{\mathsf{Z}}$ *be finite sets of event names. Let all four sets be pairwise disjoint. The quintuple* $\mathcal{S} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, S)$ *is called* named condition/event system, *where* $S : C(\bar{\mathsf{U}}) \times E(\bar{\mathsf{V}}) \to 2^{C(\bar{\mathsf{Y}}) \times E(\bar{\mathsf{Z}})} \setminus \{\emptyset\}$ *is the* system behavior function. *We also use the symbol* S *for the* system behavior relation $R \subseteq \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}) = C(\bar{\mathsf{U}}) \times E(\bar{\mathsf{V}}) \times C(\bar{\mathsf{Y}}) \times E(\bar{\mathsf{Z}})$ *with* $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in R$ *if and only if* $(s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}})$. □

Sometimes, $S_{\mathcal{S}}$ is used to denote the behavior function/relation of the named CES $\mathcal{S}$. For the sake of brevity, we will use the symbols $s_{\bar{\mathsf{U}}}$, $s_{\bar{\mathsf{V}}}$, $s_{\bar{\mathsf{Y}}}$, and $s_{\bar{\mathsf{Z}}}$ to denote the four projections of a given $s \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$ on its components.

4.1.2 Named Discrete C/E Systems

Now we give the definition of *named discrete condition/event systems.* Note that besides the introduction of names, we now also have a set of initial states instead of only one initial state, because we want to have more than one possibility for the condition output evaluation at time 0. This is discussed in detail in Example 4.26 on page 69.

Definition 4.5 (Named DCES, well-behaved) *A* named discrete condition/event system $\mathcal{D} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, X, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, f, g, h, X_0)$ *consists of*

- *a finite set* $\bar{\mathsf{U}}$ *of* condition input names,
- *a finite set* $\bar{\mathsf{V}}$ *of* event input names,
- *a finite set* $\bar{\mathsf{Y}}$ *of* condition output names,
- *a finite set* $\bar{\mathsf{Z}}$ *of* event output names,
- *a finite set* X *of* states,
- *a* state transition function $f : X \times \Sigma_{\bar{\mathsf{U}}} \times \Sigma_{\bar{\mathsf{V}}} \to 2^X \setminus \{\emptyset\}$,
- *a* condition output function $g : X \times \Sigma_{\bar{\mathsf{U}}} \to \Sigma_{\bar{\mathsf{Y}}}$,
- *an* event output function $h : X \times X \times \Sigma_{\bar{\mathsf{V}}} \to \Sigma_{\bar{\mathsf{Z}}}$, *and*
- *a nonempty set* $X_0 \subseteq X$ *of initial states,*

where $\bar{\mathsf{U}}$, $\bar{\mathsf{V}}$, $\bar{\mathsf{Y}}$, *and* $\bar{\mathsf{Z}}$ *are pairwise disjoint. The named DCES* $\mathcal{D}$ *is called* well-behaved, *if it satisfies*

- stuttering: $x \in f(x, u, \mathbf{0})$ *and*
- output triggering: $h(x, x, \mathbf{0}) = \mathbf{0}$,

for all $x \in X$ *and* $u \in \Sigma_{\bar{\mathsf{U}}}$. □

In the following we will use well-behaved named DCESs only. To define the semantics for a named DCES, we define *runs* of named DCESs. Notice that in contrast to Definition 2.6, we have $r(0) \in X_0$ as start of the run.

Definition 4.6 (Run of a named DCES) *Let $\mathcal{D}$ be as in Definition 4.5. We call a right-continuous and finite-variable function $r : \mathbb{R}_{\geq 0} \to X$* run *of $\mathcal{D}$ over $s \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$, if and only if*

1. $r(t) \in f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))$ *for all* $t \in \mathbb{R}_{>0}$,
2. $s_{\bar{\mathsf{Y}}}(t) = g(r(t), s_{\bar{\mathsf{U}}}(t))$ *for all* $t \in \mathbb{R}_{\geq 0}$,
3. $s_{\bar{\mathsf{Z}}}(t) = h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))$ *for all* $t \in \mathbb{R}_{>0}$, *and*
4. $r(0) \in X_0$. □

Now we can assign a semantics to named DCESs. Like in Definition 2.7, it is given via the notion of runs.

Definition 4.7 (Semantics of a named DCES) *Let $\mathcal{D}$ be as in Definition 4.5. The* semantics of $\mathcal{D}$ *is given by the named CES $\mathcal{S}_{\mathcal{D}} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, S_{\mathcal{D}})$ with $s \in S_{\mathcal{D}}$ if and only if there exists a run r of $\mathcal{D}$ over s, for all $s \in$* $\mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$. □

4.2 The Parallel Interconnection of C/E Systems

Having introduced component names for single condition/event systems, we are now interested in connecting some systems. As mentioned before, all components sharing the same name will be connected to each other.

Figure 4.1 shows an example with three systems (we do not distinguish between conditions and events in this example). The component names of $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ completely determine the connections between them. All outputs of the single systems will become outputs of the composed system, in this case A, D, C, and F. The inputs of the composed system are all inputs of the single systems for which there exist no output of the same name. This results in the input names B and E. Figure 4.2 shows a block diagram of the composed system.

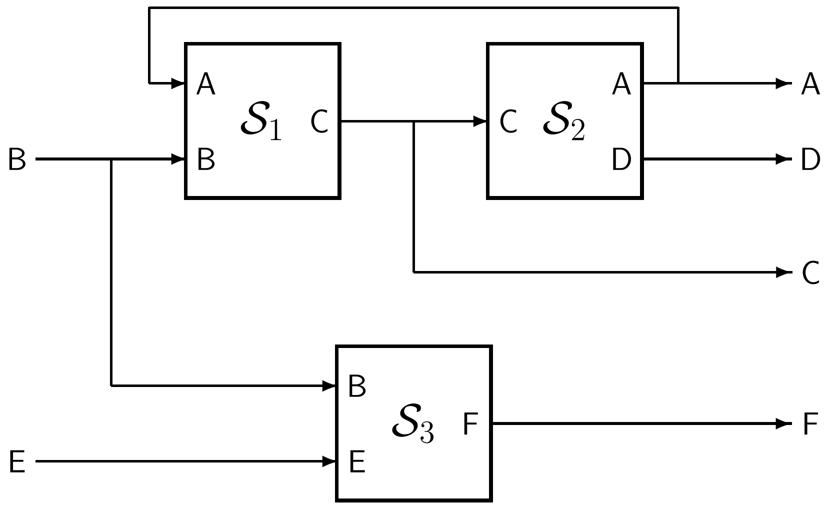


Figure 4.1: Example of a parallel interconnection.

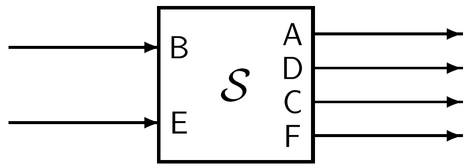


Figure 4.2: The composition of the system in Figure 4.1.

Remark 4.8 In control theory, there is at least one more possibility to describe connections between systems: One can use a function that describes for each component of a system how this component is connected to other systems. Such functions have some advantages. For example, one can use one name for two different components even if they are *not* connected. This supports the reuse of modules from a library of systems without having to rename the components.

Though we do not use this method, it is clear that both methods are equally expressive and can be transformed into each other. $\diamond$

In Definition 4.4, we assured that a name is used at most once within one system. For a set of systems to be connected to each other, we also have to ensure that in the whole set of systems all output components must have different names, since outputs must not be connected to each other. This requirement is formalized in the following definition.

Definition 4.9 (Consistent set of named CESs) *For $i \in \{1, \ldots, n\}$, let $\mathcal{S}_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, S_i)$ be a named C/E system. The set $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ is called* consistent, *if*

$$\left(\bar{\mathsf{Y}}_i \cup \bar{\mathsf{Z}}_i\right) \cap \left(\bar{\mathsf{Y}}_j \cup \bar{\mathsf{Z}}_j\right) = \emptyset$$

holds for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$. □

The parallel interconnection of C/E systems can now be defined. This is again only a descriptive definition; we do not reason about the existence of a parallel interconnection for any given set of C/E systems.

Definition 4.10 (Parallel interconnection) *For $i \in \{1, \ldots, n\}$, let $\mathcal{S}_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, S_i)$ be a named C/E system. Let $\{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ be* consistent. *A named C/E system $\mathcal{S} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, S)$ is called* parallel interconnection *of $\mathcal{S}_1, \ldots, \mathcal{S}_n$, if the following conditions hold:*

1. *The names of the components of $\mathcal{S}$ are those of the components of the single systems, except the inputs for which there exists an output with the same name:*

$$\begin{aligned} \bar{\mathsf{U}} &= \left(\textstyle\bigcup_{i=1}^n \bar{\mathsf{U}}_i\right) \setminus \left(\textstyle\bigcup_{i=1}^n \bar{\mathsf{Y}}_i\right), \quad \bar{\mathsf{Y}} = \textstyle\bigcup_{i=1}^n \bar{\mathsf{Y}}_i, \\ \bar{\mathsf{V}} &= \left(\textstyle\bigcup_{i=1}^n \bar{\mathsf{V}}_i\right) \setminus \left(\textstyle\bigcup_{i=1}^n \bar{\mathsf{Z}}_i\right), \quad \bar{\mathsf{Z}} = \textstyle\bigcup_{i=1}^n \bar{\mathsf{Z}}_i. \end{aligned}$$

2. *For any $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$, we have $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S$ if and only if there exist $(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_i$, for all $i \in \{1, \ldots, n\}$, such that the following two conditions are satisfied:*

   (a) *Connected components of the single systems share the same signals: For all $i, j \in \{1, \ldots, n\}$ we have*

$$\begin{aligned} &s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j}, \text{ and} \\ &s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j} = s_{\bar{\mathsf{Z}}_j}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j}. \end{aligned}$$

   (b) *The signals of the single systems match those of $\mathcal{S}$: For all $i \in \{1, \ldots, n\}$ we have*

$$\begin{aligned} &s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} = s_{\bar{\mathsf{U}}}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}}, \quad s_{\bar{\mathsf{Y}}_i} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i}, \text{ and} \\ &s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} = s_{\bar{\mathsf{V}}}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}}, \quad s_{\bar{\mathsf{Z}}_i} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i}. \end{aligned}$$

□

4.3 The Parallel Interconnection of Discrete C/E Systems

In Definition 4.9, we had to ensure that in a set of named CESs which are connected to each other, all output names are different. For discrete systems, this is done analogously in the following definition.

Definition 4.11 (Consistent set of named DCESs) *Let $\mathcal{D}_i$ be a named DCES with component name sets $\bar{\mathsf{U}}_i$, $\bar{\mathsf{V}}_i$, $\bar{\mathsf{Y}}_i$, and $\bar{\mathsf{Z}}_i$, for $i \in \{1, \ldots, n\}$. The set $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ is called* consistent*, if*

$$\left(\bar{\mathsf{Y}}_i \cup \bar{\mathsf{Z}}_i\right) \cap \left(\bar{\mathsf{Y}}_j \cup \bar{\mathsf{Z}}_j\right) = \emptyset$$

holds for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$. □

For the remainder of this section, let $n \in \mathbb{N}$, and, for all $i \in \{1, \ldots, n\}$, let $\mathcal{D}_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, X_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, f_i, g_i, h_i, X_{0,i})$ be a named DCES.

The following definition introduces a classification of component names which distinguishes external and internal names.

Definition 4.12 (Component name sets) *We define some sets of input and output component names of $\mathcal{D}_1, \ldots, \mathcal{D}_n$:*

$$\begin{aligned}
\mathsf{C}^{\mathsf{in}} &= \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{U}}_i\right) \setminus \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{Y}}_i\right), & \mathsf{E}^{\mathsf{in}} &= \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{V}}_i\right) \setminus \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{Z}}_i\right), \\
\mathsf{C}^{\mathsf{io}} &= \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{U}}_i\right) \cap \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{Y}}_i\right), & \mathsf{E}^{\mathsf{io}} &= \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{V}}_i\right) \cap \left(\textstyle\bigcup_{i=1}^{n} \bar{\mathsf{Z}}_i\right), \\
\mathsf{C}^{\mathsf{out}} &= \textstyle\bigcup_{i=1}^{n} \bar{\mathsf{Y}}_i\,, & \mathsf{E}^{\mathsf{out}} &= \textstyle\bigcup_{i=1}^{n} \bar{\mathsf{Z}}_i\,.
\end{aligned}$$ □

The set C^{in} contains all condition input component names for which there exists no output component of that name. These are the names of the condition signals which are not produced within the system; thus, these signals are *external* and have to be provided by the environment of the compound system. The set C^{io} contains all condition component names for which there exist input and output components of that name. These are the names for the *internal* condition signals which are exchanged by the systems. The set $\mathsf{C}^{\mathsf{out}}$ contains all output condition component names (it is a superset of C^{io}). The sets E^{in}, E^{io}, and $\mathsf{E}^{\mathsf{out}}$ have analog meanings for events. Using the notation of Definition 4.10, we have $\mathsf{C}^{\mathsf{in}} = \bar{\mathsf{U}}$, $\mathsf{E}^{\mathsf{in}} = \bar{\mathsf{V}}$, $\mathsf{C}^{\mathsf{out}} = \bar{\mathsf{Y}}$, and $\mathsf{E}^{\mathsf{out}} = \bar{\mathsf{Z}}$.

Looking at the example in Figures 4.1 and 4.2, this definition leads to $\mathsf{C}^{\mathsf{in}} = \{\mathsf{B}, \mathsf{E}\}$, $\mathsf{C}^{\mathsf{io}} = \{\mathsf{A}, \mathsf{C}\}$, and $\mathsf{C}^{\mathsf{out}} = \{\mathsf{A}, \mathsf{C}, \mathsf{D}, \mathsf{F}\}$ (provided we only have conditions).

As for the feedback connection and the loop interconnection of DCESs, we use a *unique solution property* to ensure that there exists a discrete system with the same semantics as a parallel interconnection of the single systems' semantics.

Theorem 4.13 *Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be consistent. A named DCES $\mathcal{D}$ can be effectively constructed such that $\mathcal{S}_{\mathcal{D}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}, \ldots, \mathcal{S}_{\mathcal{D}_n}$, if the following* unique solution property *holds for $\mathcal{D}_1, \ldots, \mathcal{D}_n$:*

- *For all $x_1 \in X_1, \ldots, x_n \in X_n$, $u \in \Sigma^{\mathsf{in}}_{\mathsf{C}}$, there exists a unique $y \in \Sigma_{\mathsf{C}^{\mathsf{io}}}$ such that for all $i \in \{1, \ldots, n\}$, we have*

$$g_i\left(x_i, (u \cup y)|_{\bar{\mathsf{U}}_i}\right)|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}} = y|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}}.$$

- *For all $x_1, x_1' \in X_1, \ldots, x_n, x_n' \in X_n$, $v \in \Sigma^{\mathsf{in}}_{\mathsf{E}}$, there exists a unique $z \in \Sigma_{\mathsf{E}^{\mathsf{io}}}$ such that for all $i \in \{1, \ldots, n\}$, we have*

$$h_i\left(x_i, x_i', (v \cup z)|_{\bar{\mathsf{V}}_i}\right)|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}} = z|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}}. \qquad \square$$

Before we prove this theorem, we take a look at the possible number of solutions for the evaluations $y \in \mathsf{C}^{\mathsf{io}}$ and $z \in \mathsf{E}^{\mathsf{io}}$.

Example 4.14 Reconsider the system in Figure 1.1. We define the DCESs $\mathcal{D}_1$ and $\mathcal{D}_2$ formally, this time in a parameterized version (cf. Figure 4.3).

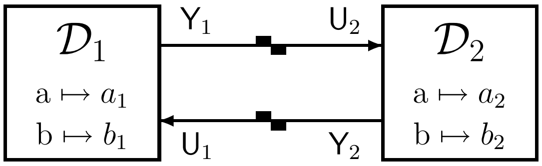


Figure 4.3: Mutually connected DCESs.

For $i \in \{1, 2\}$, let $\mathcal{D}_i = (\{\mathsf{U}_i\}, \emptyset, \{x\}, \{\mathsf{Y}_i\}, \emptyset, f_i, g_i, h_i, \{x\})$ be a named DCES with $\alpha(\mathsf{U}_i) = \alpha(\mathsf{Y}_i) = \{\mathrm{a}, \mathrm{b}\}$, and

- $f_i(x, u, \emptyset) = \{x\}$ for all $u \in \Sigma_{\mathsf{U}_i}$,
- $g_i(x, u)(\mathsf{Y}_i) = \begin{cases} a_i, \text{ if } u(\mathsf{U}_i) = \mathrm{a} \\ b_i, \text{ if } u(\mathsf{U}_i) = \mathrm{b} \end{cases}$, and

- $h_i(x, x, \emptyset) = \emptyset$.

We have only one state x and empty sets of event names, so the functions f_i and h_i are constant, and well-definedness is trivially fulfilled.[1] We characterize the connections between $\mathcal{D}_1$ and $\mathcal{D}_2$ by demanding $\mathsf{U}_1 = \mathsf{Y}_2 \neq \mathsf{Y}_1 = \mathsf{U}_2$.

Since $\mathcal{D}_1$ and $\mathcal{D}_2$ do not have any event names in common (there are no event names and alphabets at all), we have $\mathsf{E}^{\mathsf{io}} = \emptyset$, and the second condition of unique solution property is trivially fulfilled; the solution is always $z = \emptyset \in \Sigma_{\mathsf{E}^{\mathsf{io}}}$. Since there are no external condition inputs, we have $\mathsf{C}^{\mathsf{in}} = \emptyset$, and the first condition reduces to finding a unique $y \in \Sigma_{\{\mathsf{U}_1,\mathsf{U}_2\}}$ such that

$$g_1(x, y|_{\mathsf{U}_1}) = y|_{\mathsf{U}_2} \quad \text{and} \quad g_2(x, y|_{\mathsf{U}_2}) = y|_{\mathsf{U}_1}.$$

Now consider the following three setups:

1. $a_1 = \mathrm{a}$, $b_1 = \mathrm{b}$, and $a_2 = \mathrm{b}$, $b_2 = \mathrm{a}$.
2. $a_1 = \mathrm{a}$, $b_1 = \mathrm{a}$, and $a_2 = \mathrm{a}$, $b_2 = \mathrm{a}$.
3. $a_1 = \mathrm{a}$, $b_1 = \mathrm{b}$, and $a_2 = \mathrm{a}$, $b_2 = \mathrm{b}$.

For each of the three cases, we take a look at the solutions for the two equations above.

Case 1 has no solutions for y, since the two equations of the unique solution property are contradictory. No DCES $\mathcal{D}$ can be given such that $\mathcal{S}_\mathcal{D}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$. We have the existence of an algebraic loop; this is the case discussed in the introduction using Figure 1.1.

Case 2 has the unique solution $y = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}$, and, if Theorem 4.13 is sound, a DCES for the overall system can be effectively constructed, see upcoming Example 4.16.

In case 3 we have two solutions: $y = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}$ and $y = \{\mathsf{U}_1 \mapsto \mathrm{b}, \mathsf{U}_2 \mapsto \mathrm{b}\}$. Theorem 4.13 does not cover this case, and up to this point we do not know if a suitable DCES exists. At the end of Section 4.6, we will show in Example 4.30 that this is possible. ◇

The previous example could have been formulated using events instead of conditions. But this is a bit more complex, since well-behavedness (output triggering, to be precise) would force us to use at least two states in each DCES, or, if we have only one state, to use an external trigger event. Otherwise, we would not be able to generate any proper output event.

Now we prove Theorem 4.13.

[1]For output triggering, note that $\mathbf{0}_\emptyset = \emptyset$ (the evaluation with the empty domain).

Proof Let $\mathcal{D}_1, \ldots, \mathcal{D}_n$ be consistent, let the unique solution property hold, and let $G(x_1, \ldots, x_n, u) \in \Sigma_{\mathsf{C}^{\mathsf{io}}}$, respectively, $H(x_1, \ldots, x_n, x_1', \ldots, x_n', v) \in \Sigma_{\mathsf{E}^{\mathsf{io}}}$ be the unique solution y, respectively, z, for $x_1, x_1' \in X_1, \ldots, x_n, x_n' \in X_n$, $u \in \Sigma_{\mathsf{C}}^{\mathsf{in}}$, and $v \in \Sigma_{\mathsf{E}}^{\mathsf{in}}$.

We construct a named DCES $\mathcal{D}$ such that $\mathcal{S}_{\mathcal{D}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}, \ldots, \mathcal{S}_{\mathcal{D}_n}$. Let $\mathcal{D} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, X, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, f, g, h, X_0)$ be the named DCES with component names $\bar{\mathsf{U}} = \mathsf{C}^{\mathsf{in}}$, $\bar{\mathsf{V}} = \mathsf{E}^{\mathsf{in}}$, $\bar{\mathsf{Y}} = \mathsf{C}^{\mathsf{out}}$, $\bar{\mathsf{Z}} = \mathsf{E}^{\mathsf{out}}$, states $X = X_1 \times \ldots \times X_n$, $X_0 = X_{0,1} \times \ldots \times X_{0,n}$, and, for all $x = (x_1, \ldots, x_n) \in X$, $x' = (x_1', \ldots, x_n') \in X$, $u \in \Sigma_{\bar{\mathsf{U}}}$, $v \in \Sigma_{\bar{\mathsf{V}}}$,

- $x' \in f(x, u, v)$ if and only if for all $i \in \{1, \ldots, n\}$:
  $x_i' \in f_i\Big(x_i, (u \cup G(x, u))|_{\bar{\mathsf{U}}_i}, (v \cup H(x, x', v))|_{\bar{\mathsf{V}}_i}\Big)$,
- $g(x, u) = \bigcup_{i=1}^{n} g_i\Big(x_i, (u \cup G(x, u))|_{\bar{\mathsf{U}}_i}\Big)$, and
- $h(x, x', v) = \bigcup_{i=1}^{n} h_i\Big(x_i, x_i', (v \cup H(x, x', v))|_{\bar{\mathsf{V}}_i}\Big)$.

Condition 1 of Definition 4.10 holds obviously by Definition 4.12. For proving condition 2, let $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$. We have

$$(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S_{\mathcal{D}}$$

if and only if (by Definition 4.7)

$$\begin{aligned}
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} :\\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))\\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : s_{\bar{\mathsf{Y}}}(t) = g(r(t), s_{\bar{\mathsf{U}}}(t))\\
&3.\ \forall t \in \mathbb{R}_{>0} : s_{\bar{\mathsf{Z}}}(t) = h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))\\
&4.\ r(0) \in X_0
\end{aligned}$$

if and only if (by construction of f, g, h, and X_0 of the named DCES $\mathcal{D}$)

$$\begin{aligned}
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} :\\
&\forall i \in \{1, \ldots, n\} :\\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t)_i \in f_i\Big(r(t^-)_i, \Big(s_{\bar{\mathsf{U}}}(t^-) \cup G(r(t^-), s_{\bar{\mathsf{U}}}(t^-))\Big)|_{\bar{\mathsf{U}}_i},\\
&\qquad\qquad \Big(s_{\bar{\mathsf{V}}}(t) \cup H(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))\Big)|_{\bar{\mathsf{V}}_i}\Big)\\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : s_{\bar{\mathsf{Y}}}(t)|_{\bar{\mathsf{Y}}_i} = g_i\Big(r(t)_i, \Big(s_{\bar{\mathsf{U}}}(t) \cup G(r(t), s_{\bar{\mathsf{U}}}(t))\Big)|_{\bar{\mathsf{U}}_i}\Big)\\
&3.\ \forall t \in \mathbb{R}_{>0} : s_{\bar{\mathsf{Z}}}(t)|_{\bar{\mathsf{Z}}_i} = h_i\Big(r(t^-)_i, r(t)_i,\\
&\qquad\qquad \Big(s_{\bar{\mathsf{V}}}(t) \cup H(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))\Big)|_{\bar{\mathsf{V}}_i}\Big)\\
&4.\ r(0)_i \in X_{0,i}
\end{aligned}$$

if and only if (we split the run r into the runs $r_1, \ldots, r_n$ (we abbreviate $r_1(\cdot), \ldots, r_n(\cdot)$ by $\bar{r}(\cdot)$ in the parameters of G and H), and move the signal restrictions into the arguments of the unions)

$$
\begin{aligned}
&\exists r_i : \mathbb{R}_{\geq 0} \to X_i, \text{ for } i \in \{1, \ldots, n\} \text{ (right-continuous, finite-variable)} :\\
&\forall i \in \{1, \ldots, n\} :\\
&1.\ \forall t \in \mathbb{R}_{>0} :\\
&\qquad r_i(t) \in f_i\Big(r_i(t^-), \Big(s_{\bar{\mathsf{U}}}(t^-)|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} \cup G(\bar{r}(t^-), s_{\bar{\mathsf{U}}}(t^-))|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{io}}}\Big),\\
&\qquad\qquad \Big(s_{\bar{\mathsf{V}}}(t)|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} \cup H(\bar{r}(t^-), \bar{r}(t), s_{\bar{\mathsf{V}}}(t))|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{io}}}\Big)\Big)\\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : s_{\bar{\mathsf{Y}}}(t)|_{\bar{\mathsf{Y}}_i} = g_i\Big(r_i(t), \Big(s_{\bar{\mathsf{U}}}(t)|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} \cup G(\bar{r}(t), s_{\bar{\mathsf{U}}}(t))|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{io}}}\Big)\Big)\\
&3.\ \forall t \in \mathbb{R}_{>0} : s_{\bar{\mathsf{Z}}}(t)|_{\bar{\mathsf{Z}}_i} = h_i\Big(r_i(t^-), r_i(t),\\
&\qquad\qquad \Big(s_{\bar{\mathsf{V}}}(t)|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} \cup H(\bar{r}(t^-), \bar{r}(t), s_{\bar{\mathsf{V}}}(t))|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{io}}}\Big)\Big)\\
&4.\ r_i(0) \in X_{0,i}
\end{aligned}
$$

if and only if (by Definition 4.7, and by the fact that the unique solutions G and H determine for all systems the signal components with names in C^{io} and E^{io})

$$
\begin{aligned}
&\exists (s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_{\mathcal{D}_i} \text{ (for all } i \in \{1, \ldots, n\}) :\\
&\text{(a) } \forall i, j \in \{1, \ldots, n\} :\\
&\qquad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j},\\
&\qquad s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j} = s_{\bar{\mathsf{Z}}_j}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j}.\\
&\text{(b) } \forall i \in \{1, \ldots, n\} :\\
&\qquad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} = s_{\bar{\mathsf{U}}}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}}, \quad s_{\bar{\mathsf{Y}}_i} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i},\\
&\qquad s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} = s_{\bar{\mathsf{V}}}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}}, \quad s_{\bar{\mathsf{Z}}_i} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i}.
\end{aligned}
$$

Thus, $\mathcal{S}_{\mathcal{D}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}, \ldots, \mathcal{S}_{\mathcal{D}_n}$. ■

The construction in this proof is the basis for our parallel interconnection operator for DCESs.

Definition 4.15 (Discrete parallel interconnection) *Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be consistent, and let the unique solution property of Theorem 4.13 hold. The named DCES $\mathcal{D}$ gained from the construction above is called the* discrete parallel interconnection *of $\mathcal{D}_1, \ldots, \mathcal{D}_n$, denoted by $\mathcal{D}_1 \| \ldots \| \mathcal{D}_n$.* □

In the following example, two small DCESs are composed using the parallel interconnection operator.

Example 4.16 Consider again the DCES $\mathcal{D}_1$ and $\mathcal{D}_2$ of Example 4.14. We take a look at case 2, where $a_1 = \mathrm{a}$, $b_1 = \mathrm{a}$, and $a_2 = \mathrm{a}$, $b_2 = \mathrm{a}$. As stated already, the unique solution property of Theorem 4.13 holds, and the solutions are $G(x, \emptyset) = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}$ and $H(x, x, \emptyset) = \emptyset$. So we can define

$$\mathcal{D} = \mathcal{D}_1 \| \mathcal{D}_2,$$

and by Definition 4.15 above and the construction in the proof of Theorem 4.13, we get the DCES $\mathcal{D} = (\emptyset, \emptyset, \{(x, x)\}, \bar{\mathsf{Y}}, \emptyset, f, g, h, \{(x, x)\})$ with $\bar{\mathsf{Y}} = \{\mathsf{U}_1, \mathsf{U}_2\}$, $\alpha(\mathsf{U}_1) = \alpha(\mathsf{U}_2) = \{\mathrm{a}, \mathrm{b}\}$, and

- $f((x, x), \emptyset, \emptyset) = \{(x, x)\}$,
- $g((x, x), \emptyset) = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}$, and
- $h((x, x), (x, x), \emptyset) = \emptyset$.

Thus, $\mathcal{D}$ has one state, no inputs, and two condition output components named U_1 and U_2; both deliver constantly the condition symbol a.

There is only one run of $\mathcal{D}$; so by Definition 4.7, we have as formal semantics of $\mathcal{D}$ the CES $\mathcal{S}_{\mathcal{D}} = (\emptyset, \emptyset, \bar{\mathsf{Y}}, \emptyset, S_{\mathcal{D}})$ with

$$S_{\mathcal{D}} = \{((\emptyset, \emptyset, s_{\bar{\mathsf{Y}}}, \emptyset) \mid s_{\bar{\mathsf{Y}}}(t) = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\} \text{ for all } t \in \mathbb{R}_{\geq 0}\}. \qquad \diamond$$

4.4 Hiding Output Components in CESs and DCESs

When using parallel interconnection for system combination, all output signals of the single systems remain visible in the composed system. This is, however, not always desired. Often signals are used for internal communication among a small set of systems, and no other system has these signals as input. When this small set of systems is merged to one system using the parallel interconnection operator, these internal signals become redundant. Our semantics of the parallel interconnection, however, leaves these signals visible. This is comparable to the parallel operator in the language CSP [Hoa85], where the common symbols remain visible after the composition.

For signals that are considered as internal, we would like to have something like in CCS [Mil80, Mil89], where common actions are composed by the parallel operator to an "invisible" action τ, which can be removed later. Therefore, we now provide a hiding operator which simply removes such signals from a system's output components. We define this operator for named CESs and named DCESs.

Definition 4.17 (CES output component hiding) *Let* $\mathcal{S} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, S)$ *be a named CES, and let* $\bar{\mathsf{N}} \subseteq \mathcal{N}$*. The* output component hiding *of* $\bar{\mathsf{N}}$ *in* $\mathcal{S}$*, denoted by* $\mathrm{hide}(\mathcal{S}, \bar{\mathsf{N}})$*, is defined as the named CES* $\mathcal{S}' = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}', \bar{\mathsf{Z}}', S')$ *with*

- $\bar{\mathsf{Y}}' = \bar{\mathsf{Y}} \setminus \bar{\mathsf{N}}$, $\bar{\mathsf{Z}}' = \bar{\mathsf{Z}} \setminus \bar{\mathsf{N}}$, *and*
- $S' = \Big\{ (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}'}) \mid (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S \Big\}$

For $\mathsf{N} \in \mathcal{N}$*, we use* $\mathrm{hide}(\mathcal{S}, \mathsf{N})$ *as an abbreviation for* $\mathrm{hide}(\mathcal{S}, \{\mathsf{N}\})$*.* □

A semantic characterization of the hiding operator for named CESs is given in the following lemma.

Lemma 4.18 *Let* $\mathcal{S} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, S)$ *be a named CES, and let* $\bar{\mathsf{N}} \subseteq \mathcal{N}$*. Then, we have*

$$\begin{aligned}
&(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in S_{\mathrm{hide}(\mathcal{S}, \bar{\mathsf{N}})} \iff \\
&\exists s_{\bar{\mathsf{C}}} \in C(\bar{\mathsf{Y}} \cap \bar{\mathsf{N}}), s_{\bar{\mathsf{E}}} \in E(\bar{\mathsf{Z}} \cap \bar{\mathsf{N}}) : (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'} \cup s_{\bar{\mathsf{C}}}, s_{\bar{\mathsf{Z}}'} \cup s_{\bar{\mathsf{E}}}) \in S
\end{aligned}$$

for all $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}} \setminus \bar{\mathsf{N}}, \bar{\mathsf{Z}} \setminus \bar{\mathsf{N}})$*.* □

Proof Let $\mathcal{S}$ and $\bar{\mathsf{N}}$ be given as above. Let $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}} \setminus \bar{\mathsf{N}}, \bar{\mathsf{Z}} \setminus \bar{\mathsf{N}})$. We have

$$(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in S_{\mathrm{hide}(\mathcal{S}, \bar{\mathsf{N}})}$$

if and only if (by Definition 4.17)

$$(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in \Big\{ (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}} \setminus \bar{\mathsf{N}}}, s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}} \setminus \bar{\mathsf{N}}}) \mid (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S \Big\}$$

if and only if (by the properties of the restriction operator)

$$\exists s_{\bar{\mathsf{C}}} \in C(\bar{\mathsf{Y}} \cap \bar{\mathsf{N}}), s_{\bar{\mathsf{E}}} \in E(\bar{\mathsf{Z}} \cap \bar{\mathsf{N}}) : (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'} \cup s_{\bar{\mathsf{C}}}, s_{\bar{\mathsf{Z}}'} \cup s_{\bar{\mathsf{E}}}) \in S. \qquad \blacksquare$$

Hiding for discrete condition/event systems is done by restricting the condition and event output functions to the remaining names.

Definition 4.19 (DCES output component hiding) *Let* $\mathcal{D} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, X, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, f, g, h, X_0)$ *be a named DCES, and let* $\bar{\mathsf{N}} \subseteq \mathcal{N}$*. The* output component hiding *of* $\bar{\mathsf{N}}$ *in* $\mathcal{D}$*, denoted by* $\mathrm{hide}(\mathcal{D}, \bar{\mathsf{N}})$*, is defined as the named DCES* $\mathcal{D}' = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, X, \bar{\mathsf{Y}}', \bar{\mathsf{Z}}', f, g', h', X_0)$ *with*

- $\bar{\mathsf{Y}}' = \bar{\mathsf{Y}} \setminus \bar{\mathsf{N}}$, $\bar{\mathsf{Z}}' = \bar{\mathsf{Z}} \setminus \bar{\mathsf{N}}$, *and*
- $g'(x,u) = g(x,u)|_{\bar{\mathsf{Y}}'}$, $h'(x,x',v) = h(x,x',v)|_{\bar{\mathsf{Z}}'}$ *for all* x, x', u, v.

For $\mathsf{N} \in \mathcal{N}$, *we use* $\mathrm{hide}(\mathcal{D}, \mathsf{N})$ *as an abbreviation for* $\mathrm{hide}(\mathcal{D}, \{\mathsf{N}\})$. □

By checking Definition 4.5 it is obvious that well-behavedness (stuttering and output triggering) is preserved when applying the hiding operator for DCESs. Furthermore, both versions of the hiding operator have the following properties for all CESs or DCESs $\mathcal{S}$ and all $\bar{\mathsf{N}}, \bar{\mathsf{M}} \in \mathcal{N}$:

- $\mathrm{hide}(\mathcal{S}, \bar{\mathsf{N}}) = \mathcal{S}$, if $\bar{\mathsf{N}} \cap (\bar{\mathsf{Y}} \cup \bar{\mathsf{Z}}) = \emptyset$.
- $\mathrm{hide}(\mathrm{hide}(\mathcal{S}, \bar{\mathsf{N}}), \bar{\mathsf{M}}) = \mathrm{hide}(\mathrm{hide}(\mathcal{S}, \bar{\mathsf{M}}), \bar{\mathsf{N}}) = \mathrm{hide}(\mathcal{S}, \bar{\mathsf{N}} \cup \bar{\mathsf{M}})$.

Note that the first property also states that *input* components are not removed.

The semantic connection between the CES version and the DCES version of the hiding operator is given in the following lemma: If we use the semantics of a DCES to transform it into a CES, and we hide some outputs, it makes no difference to the CES whether the hiding is done *before* or *after* the transformation.

Lemma 4.20 *Let* $\mathcal{D}$ *be a named DCES, and let* $\bar{\mathsf{N}} \subseteq \mathcal{N}$. *Then we have* $\mathcal{S}_{\mathrm{hide}(\mathcal{D},\bar{\mathsf{N}})} = \mathrm{hide}(\mathcal{S}_{\mathcal{D}}, \bar{\mathsf{N}})$. □

Proof Let $\mathcal{D}$ be given as in Definition 4.19, and let $\bar{\mathsf{N}} \subseteq \mathcal{N}$. The component name sets of $\mathcal{S}_{\mathrm{hide}(\mathcal{D},\bar{\mathsf{N}})}$ and $\mathrm{hide}(\mathcal{S}_{\mathcal{D}}, \bar{\mathsf{N}})$ are obviously equal.

It remains to prove $S_{\mathrm{hide}(\mathcal{D},\bar{\mathsf{N}})} = S_{\mathrm{hide}(\mathcal{S}_{\mathcal{D}},\bar{\mathsf{N}})}$. Let $\bar{\mathsf{Y}}' = \bar{\mathsf{Y}} \setminus \bar{\mathsf{N}}$, $\bar{\mathsf{Z}}' = \bar{\mathsf{Z}} \setminus \bar{\mathsf{N}}$, and let $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}', \bar{\mathsf{Z}}')$. We have

$$(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in S_{\mathrm{hide}(\mathcal{D},\bar{\mathsf{N}})}$$

if and only if (by Definition 4.19 and Definition 4.7)

$$\begin{aligned}
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} : \\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t)) \\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : s_{\bar{\mathsf{Y}}'}(t) = g(r(t), s_{\bar{\mathsf{U}}}(t))|_{\bar{\mathsf{Y}}'} \\
&3.\ \forall t \in \mathbb{R}_{>0} : s_{\bar{\mathsf{Z}}'}(t) = h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))|_{\bar{\mathsf{Z}}'} \\
&4.\ r(0) \in X_0
\end{aligned}$$

if and only if (we introduce the signals $s_{\bar{\mathsf{C}}}$ and $s_{\bar{\mathsf{E}}}$ which contain the information hidden by the restriction of g and h)

$$
\begin{aligned}
&\exists s_{\bar{\mathsf{C}}} \in C(\bar{\mathsf{Y}} \cap \bar{\mathsf{N}}), s_{\bar{\mathsf{E}}} \in E(\bar{\mathsf{Z}} \cap \bar{\mathsf{N}}) : \\
&\exists r : \mathbb{R}_{\geq 0} \to X \text{ (right-continuous, finite-variable)} : \\
&1.\ \forall t \in \mathbb{R}_{>0} : r(t) \in f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t)) \\
&2.\ \forall t \in \mathbb{R}_{\geq 0} : (s_{\bar{\mathsf{Y}}'} \cup s_{\bar{\mathsf{C}}})(t) = g(r(t), s_{\bar{\mathsf{U}}}(t)) \\
&3.\ \forall t \in \mathbb{R}_{>0} : (s_{\bar{\mathsf{Z}}'} \cup s_{\bar{\mathsf{E}}})(t) = h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t)) \\
&4.\ r(0) \in X_0
\end{aligned}
$$

if and only if (by Definition 4.7)

$$\exists s_{\bar{\mathsf{C}}} \in C(\bar{\mathsf{Y}} \cap \bar{\mathsf{N}}), s_{\bar{\mathsf{E}}} \in E(\bar{\mathsf{Z}} \cap \bar{\mathsf{N}}) : (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'} \cup s_{\bar{\mathsf{C}}}, s_{\bar{\mathsf{Z}}'} \cup s_{\bar{\mathsf{E}}}) \in S_{\mathcal{D}}$$

if and only if (by Lemma 4.18)

$$(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}'}, s_{\bar{\mathsf{Z}}'}) \in S_{\mathrm{hide}(\mathcal{S}_{\mathcal{D}}, \bar{\mathsf{N}})}.$$

■

Example 4.21 We show that if we want to express the discrete cascade interconnection by using the discrete parallel interconnection operator, we have to hide the signals between the two DCESs. Let $\mathcal{D}_i = (\{\mathsf{U}_i\}, \{\mathsf{V}_i\}, X_i, \{\mathsf{Y}_i\}, \{\mathsf{Z}_i\}, f_i, g_i, h_i, X_{0,i})$, be a DCES, for $i \in \{1, 2\}$, with $\mathsf{Y}_1 = \mathsf{U}_2$ and $\mathsf{Z}_1 = \mathsf{V}_2$, and all other names being different (like in Figure 3.1, but with named DCESs instead of unnamed CESs). In a first step, we define

$$\mathcal{D} = \mathcal{D}_1 \| \mathcal{D}_2.$$

This operation is well-defined, since the unique solution property of Theorem 4.13 holds; the solutions are just the outputs of $\mathcal{D}_1$: $G(x_1, x_2, u) = g_1(x_1, u)$ and $H(x_1, x_2, x_1', x_2', v) = h_1(x_1, x_1', v)$.

Looking at the sets of output component names in $\mathcal{D}$, we find that they contain Y_1 and Z_1, i.e., the outputs of $\mathcal{D}_1$ are still visible in $\mathcal{D}$. This does not match our notion of cascade interconnection (Definition 3.1). So we hide these names in $\mathcal{D}$ by defining

$$\mathcal{D}' = \mathrm{hide}(\mathcal{D}, \{\mathsf{Y}_1, \mathsf{Z}_1\}).$$

Now $\mathcal{S}_{\mathcal{D}'}$ is a cascade interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_2}$.

We do not give a formal proof for this, since the next section shows that the discrete parallel interconnection operator is able to express all operators for composing DCES we have seen in Chapter 3 (cascade, feedback, and loop). ◇

4.5 Special Cases of the Discrete Parallel Interconnection

Theorem 4.22 *The discrete loop interconnection is a special case of the discrete parallel interconnection (by adding component names and using the hide operator).* □

Proof Let $\mathcal{D}_i = (U_{i,1} \times U_{i,2}, V_{i,1} \times V_{i,2}, X_i, Y_{i,1} \times Y_{i,2}, Z_{i,1} \times Z_{i,2}, f_i, g_i, h_i, x_{0,i})$, $i \in \{1, 2\}$, be two DCESs with matching alphabets

$$U_{1,2} = Y_{2,2}, V_{1,2} = Z_{2,2}, \text{ and } U_{2,2} = Y_{1,2}, V_{2,2} = Z_{1,2}$$

(cf. Figure 3.4), and let the unique solution property of Theorem 3.12 hold. Since these two DCESs are no named DCESs, we have to introduce names to make the discrete parallel interconnection operator applicable. Therefore, we transform $\mathcal{D}_1$ and $\mathcal{D}_2$ to the named DCESs $\mathcal{D}'_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, X_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, f'_i, g'_i, h'_i, \{x_{0,i}\})$, $i \in \{1, 2\}$, with component name sets

$$\begin{aligned}
&\bar{\mathsf{U}}_1 = \{\mathsf{U}_1, \mathsf{C}_{2\to1}\},\ \bar{\mathsf{V}}_1 = \{\mathsf{V}_1, \mathsf{E}_{2\to1}\},\ \bar{\mathsf{Y}}_1 = \{\mathsf{Y}_1, \mathsf{C}_{1\to2}\},\ \bar{\mathsf{Z}}_1 = \{\mathsf{Z}_1, \mathsf{E}_{1\to2}\},\\
&\bar{\mathsf{U}}_2 = \{\mathsf{U}_2, \mathsf{C}_{1\to2}\},\ \bar{\mathsf{V}}_2 = \{\mathsf{V}_2, \mathsf{E}_{1\to2}\},\ \bar{\mathsf{Y}}_2 = \{\mathsf{Y}_2, \mathsf{C}_{2\to1}\},\ \bar{\mathsf{Z}}_2 = \{\mathsf{Z}_2, \mathsf{E}_{2\to2}\},
\end{aligned}$$

associated alphabets

$$\begin{aligned}
&\alpha(\mathsf{U}_1) = U_{1,1}, \quad \alpha(\mathsf{V}_1) = V_{1,1}, \quad \alpha(\mathsf{Y}_1) = Y_{1,1}, \quad \alpha(\mathsf{Z}_1) = Z_{1,1},\\
&\alpha(\mathsf{U}_2) = U_{2,1}, \quad \alpha(\mathsf{V}_2) = V_{2,1}, \quad \alpha(\mathsf{Y}_2) = Y_{2,1}, \quad \alpha(\mathsf{Z}_2) = Z_{2,1},
\end{aligned}$$

$$\begin{aligned}
&\alpha(\mathsf{C}_{2\to1}) = U_{1,2} = Y_{2,2}, \quad \alpha(\mathsf{E}_{2\to1}) = V_{1,2} = Z_{2,2},\\
&\alpha(\mathsf{C}_{1\to2}) = U_{2,2} = Y_{1,2}, \quad \alpha(\mathsf{E}_{1\to2}) = V_{2,2} = Z_{1,2},
\end{aligned}$$

for all $x, x' \in X_1$, $u \in \Sigma_{\bar{\mathsf{U}}_1}$, $v \in \Sigma_{\bar{\mathsf{V}}_1}$,

$$\begin{aligned}
f'_1(x, u, v) &= f_1\Big(x, (u(\mathsf{U}_1), u(\mathsf{C}_{2\to1})), (v(\mathsf{V}_1), v(\mathsf{E}_{2\to1}))\Big),\\
g'_1(x, u) &= g_1\Big(x, (u(\mathsf{U}_1), u(\mathsf{C}_{2\to1}))\Big),\\
h'_1(x, u, v) &= h_1\Big(x, x', (v(\mathsf{V}_1), v(\mathsf{E}_{2\to1}))\Big),
\end{aligned}$$

and, for all $x, x' \in X_2$, $u \in \Sigma_{\bar{\mathsf{U}}_2}$, $v \in \Sigma_{\bar{\mathsf{V}}_2}$,

$$\begin{aligned}
f'_2(x, u, v) &= f_2\Big(x, (u(\mathsf{U}_2), u(\mathsf{C}_{1\to2})), (v(\mathsf{V}_2), v(\mathsf{E}_{1\to2}))\Big),\\
g'_2(x, u) &= g_2\Big(x, (u(\mathsf{U}_2), u(\mathsf{C}_{1\to2}))\Big),\\
h'_2(x, u, v) &= h_2\Big(x, x', (v(\mathsf{V}_2), v(\mathsf{E}_{1\to2}))\Big).
\end{aligned}$$

To make the signals of the unnamed systems and the named systems comparable, we define a function γ mapping evaluations φ of $\mathcal{D}'_1$ and $\mathcal{D}'_2$ to pairs of condition or event symbols which can be used by $\mathcal{D}_1$ and $\mathcal{D}_2$:

$$\gamma(\varphi) = \begin{cases} (\varphi(\mathsf{U}_1), \varphi(\mathsf{C}_{2\to 1})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{U}}_1} & (\varphi(\mathsf{U}_2), \varphi(\mathsf{C}_{1\to 2})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{U}}_2} \\ (\varphi(\mathsf{V}_1), \varphi(\mathsf{E}_{2\to 1})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{V}}_1} & (\varphi(\mathsf{V}_2), \varphi(\mathsf{E}_{1\to 2})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{V}}_2} \\ (\varphi(\mathsf{Y}_1), \varphi(\mathsf{C}_{1\to 2})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{Y}}_1} & (\varphi(\mathsf{Y}_2), \varphi(\mathsf{C}_{2\to 1})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{Y}}_2} \\ (\varphi(\mathsf{Z}_1), \varphi(\mathsf{E}_{1\to 2})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{Z}}_1} & (\varphi(\mathsf{Z}_2), \varphi(\mathsf{E}_{2\to 1})), \text{ if } \varphi \in \Sigma_{\bar{\mathsf{Z}}_2} \end{cases}$$

We overload γ in the obvious way to named condition and event signals, and to named behavior sets. With these definitions, it is obvious that $\gamma(S_{\mathcal{D}'_1}) = S_{\mathcal{D}_1}$ and $\gamma(S_{\mathcal{D}'_2}) = S_{\mathcal{D}_2}$.

Now we are ready to compare the loop and parallel operators. Obviously, $\{\mathcal{D}'_1, \mathcal{D}'_2\}$ is consistent. The unique solution property of Theorem 4.13 also holds for the discrete parallel interconnection of $\mathcal{D}'_1$ and $\mathcal{D}'_2$, since the solutions are those of the loop interconnection. We define the loop interconnection of $\mathcal{D}_1$ and $\mathcal{D}_2$:

$$\mathcal{D} = \mathcal{D}_1 \leftrightarrow \mathcal{D}_2,$$

and we use the parallel interconnection and the hiding operator to define

$$\mathcal{D}' = \text{hide}(\mathcal{D}'_1 \| \mathcal{D}'_2, \{\mathsf{C}_{1\to 2}, \mathsf{E}_{1\to 2}, \mathsf{C}_{2\to 1}, \mathsf{E}_{2\to 1}\}).$$

Note that in $\mathcal{D}'$ we explicitly hide the "internal" signals, while they are hidden implicitly in the construction of $\mathcal{D}$.

It is easy to check that $\mathcal{D}$ and $\mathcal{D}'$ have the same input and output alphabets, since the component names of $\mathcal{D}'$ are $\bar{\mathsf{U}} = \{\mathsf{U}_1, \mathsf{U}_2\}$, $\bar{\mathsf{V}} = \{\mathsf{V}_1, \mathsf{V}_2\}$, $\bar{\mathsf{Y}} = \{\mathsf{Y}_1, \mathsf{Y}_2\}$, and $\bar{\mathsf{Z}} = \{\mathsf{Z}_1, \mathsf{Z}_2\}$, which are just the names that remain after hiding the trivial components in the parallel interconnection.

Now we show $\gamma(S_{\mathcal{D}'}) = S_{\mathcal{D}}$. Let $(s_U, s_V, s_Y, s_Z) \in \text{Beh}(U_{1,1} \times U_{2,1}, V_{1,1} \times V_{2,1}, Y_{1,1} \times Y_{2,1}, Z_{1,1} \times Z_{2,1})$. We have

$$(s_U, s_V, s_Y, s_Z) \in \gamma(S_{\mathcal{D}'})$$

if and only if (by definition of γ)

$$\begin{aligned} &\exists (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \text{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}) : \\ &\gamma((s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}})) = (s_U, s_V, s_Y, s_Z) \wedge \\ &(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S_{\mathcal{D}'} \end{aligned}$$

if and only if (since $\mathcal{D}' = \text{hide}(\mathcal{D}'_1 \| \mathcal{D}'_2, \{\mathsf{C}_{1\to2}, \mathsf{E}_{1\to2}, \mathsf{C}_{2\to1}, \mathsf{E}_{2\to1}\})$, and by Lemma 4.18 and Lemma 4.20)

$$\begin{aligned}
&\exists (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \text{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}) : \\
&\gamma((s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}})) = (s_U, s_V, s_Y, s_Z) \wedge \\
&\exists s_{\bar{\mathsf{C}}} \in C(\{\mathsf{C}_{1\to2}, \mathsf{C}_{2\to1}\}), s_{\bar{\mathsf{E}}} \in E(\{\mathsf{E}_{1\to2}, \mathsf{E}_{2\to1}\}) : \\
&(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}} \cup s_{\bar{\mathsf{C}}}, s_{\bar{\mathsf{Z}}} \cup s_{\bar{\mathsf{E}}}) \in S_{\mathcal{D}'_1 \| \mathcal{D}'_2}
\end{aligned}$$

if and only if (by Definition 4.10, condition 2, and Theorem 4.13)

$$\begin{aligned}
&\exists (s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \text{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}) : \\
&\gamma((s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}})) = (s_U, s_V, s_Y, s_Z) \wedge \\
&\exists s_{\bar{\mathsf{C}}} \in C(\{\mathsf{C}_{1\to2}, \mathsf{C}_{2\to1}\}), s_{\bar{\mathsf{E}}} \in E(\{\mathsf{E}_{1\to2}, \mathsf{E}_{2\to1}\}) : \\
&\exists (s_{\bar{\mathsf{U}}_1}, s_{\bar{\mathsf{V}}_1}, s_{\bar{\mathsf{Y}}_1}, s_{\bar{\mathsf{Z}}_1}) \in S_{\mathcal{D}'_1}, (s_{\bar{\mathsf{U}}_2}, s_{\bar{\mathsf{V}}_2}, s_{\bar{\mathsf{Y}}_2}, s_{\bar{\mathsf{Z}}_2}) \in S_{\mathcal{D}'_2} : \\
&\text{(a)}\quad s_{\bar{\mathsf{U}}_1}|_{\mathsf{C}_{2\to1}} = s_{\bar{\mathsf{Y}}_2}|_{\mathsf{C}_{2\to1}},\ s_{\bar{\mathsf{U}}_2}|_{\mathsf{C}_{1\to2}} = s_{\bar{\mathsf{Y}}_1}|_{\mathsf{C}_{1\to2}}, \\
&\qquad\ s_{\bar{\mathsf{V}}_1}|_{\mathsf{E}_{2\to1}} = s_{\bar{\mathsf{Z}}_2}|_{\mathsf{E}_{2\to1}},\ s_{\bar{\mathsf{V}}_2}|_{\mathsf{E}_{1\to2}} = s_{\bar{\mathsf{Z}}_1}|_{\mathsf{E}_{1\to2}} \\
&\text{(b)}\quad s_{\bar{\mathsf{U}}_1}|_{\mathsf{U}_1} = s_{\bar{\mathsf{U}}}|_{\mathsf{U}_1},\ s_{\bar{\mathsf{U}}_2}|_{\mathsf{U}_2} = s_{\bar{\mathsf{U}}}|_{\mathsf{U}_2}, \\
&\qquad\ s_{\bar{\mathsf{V}}_1}|_{\mathsf{V}_1} = s_{\bar{\mathsf{V}}}|_{\mathsf{V}_1},\ s_{\bar{\mathsf{V}}_2}|_{\mathsf{V}_2} = s_{\bar{\mathsf{V}}}|_{\mathsf{V}_2}, \\
&\qquad\ s_{\bar{\mathsf{Y}}_1} = (s_{\bar{\mathsf{Y}}} \cup s_{\bar{\mathsf{C}}})|_{\bar{\mathsf{Y}}_1},\ s_{\bar{\mathsf{Y}}_2} = (s_{\bar{\mathsf{Y}}} \cup s_{\bar{\mathsf{C}}})|_{\bar{\mathsf{Y}}_2}, \\
&\qquad\ s_{\bar{\mathsf{Z}}_1} = (s_{\bar{\mathsf{Z}}} \cup s_{\bar{\mathsf{E}}})|_{\bar{\mathsf{Z}}_1},\ s_{\bar{\mathsf{Z}}_2} = (s_{\bar{\mathsf{Z}}} \cup s_{\bar{\mathsf{E}}})|_{\bar{\mathsf{Z}}_2}
\end{aligned}$$

if and only if (by definition of γ)

$$\begin{aligned}
&\exists \bar{s}_{U_1} \in C(U_{1,2}), \bar{s}_{U_2} \in C(U_{2,2}), \bar{s}_{V_1} \in C(V_{1,2}), \bar{s}_{V_2} \in C(V_{2,2}) : \\
&\exists (s_U^1, s_V^1, s_Y^1, s_Z^1) \in \gamma(S_{\mathcal{D}'_1}), (s_U^2, s_V^2, s_Y^2, s_Z^2) \in \gamma(S_{\mathcal{D}'_2}) : \\
&\text{(a)}\quad (s_U^1)_2 = (s_Y^2)_2,\ (s_U^2)_2 = (s_Y^1)_2, \\
&\qquad\ (s_V^1)_2 = (s_Z^2)_2,\ (s_V^2)_2 = (s_Z^1)_2 \\
&\text{(b)}\quad (s_U^1)_1 = (s_U)_1,\ (s_U^2)_1 = (s_U)_2, \\
&\qquad\ (s_V^1)_1 = (s_V)_1,\ (s_V^2)_1 = (s_V)_2, \\
&\qquad\ s_Y^1 = ((s_Y)_1, \bar{s}_{U_2}),\ s_Y^2 = ((s_Y)_2, \bar{s}_{U_1}), \\
&\qquad\ s_Z^1 = ((s_Z)_1, \bar{s}_{V_2}),\ s_Z^2 = ((s_Z)_2, \bar{s}_{V_1})
\end{aligned}$$

if and only if (we have $\gamma(S_{\mathcal{D}'_1}) = S_{\mathcal{D}_1}$ and $\gamma(S_{\mathcal{D}'_2}) = S_{\mathcal{D}_2}$, and the existentially quantified signals of these systems are completely determined in (a) and (b))

$$\begin{aligned}
&\exists \bar{s}_{U_1} \in C(U_{1,2}), \bar{s}_{V_1} \in E(V_{1,2}), \bar{s}_{U_2} \in C(U_{2,2}), \bar{s}_{V_2} \in E(V_{2,2}) : \\
&\Big(((s_U)_1, \bar{s}_{U_1}), ((s_V)_1, \bar{s}_{V_1}), ((s_Y)_1, \bar{s}_{U_2}), ((s_Z)_1, \bar{s}_{V_2})\Big) \in S_{\mathcal{D}_1}, \\
&\Big(((s_U)_2, \bar{s}_{U_2}), ((s_V)_2, \bar{s}_{V_2}), ((s_Y)_2, \bar{s}_{U_1}), ((s_Z)_2, \bar{s}_{V_1})\Big) \in S_{\mathcal{D}_2}
\end{aligned}$$

if and only if (by Definition 3.10, $\mathcal{D} = \mathcal{D}_1 \leftrightarrow \mathcal{D}_2$, and Theorem 3.12)

$$(s_U, s_V, s_Y, s_Z) \in S_{\mathcal{D}}. \qquad \blacksquare$$

Since we already know that the discrete cascade and feedback operators are expressible in terms of the discrete loop operator, we conclude that these operations can also be expressed by the discrete parallel interconnection operator. So we have reached the goal to have one universal operator for all possible ways of connecting discrete condition/event systems.

Theorem 4.23 *The discrete cascade interconnection and the discrete feedback connection are special cases of the discrete parallel interconnection.* □

Proof This follows with Theorem 4.22 directly from Theorem 3.16 and Theorem 3.17 (cascade and feedback are special cases of loop). ■

Remark 4.24 If we want to model the feedback connection directly by using the parallel interconnection, we need a second system $\mathcal{D}_{\mathsf{id}}$ to be inserted in the feedback loop. This system just copies its inputs to its outputs. It is needed because the parallel interconnection operator (the definition of a named (D)CES, actually) does not allow a direct connection from one system to itself. This restriction also ensures that that the application of the *unary* parallel interconnection operator always results in the original system. ◇

The following example shows how this is done formally.

Example 4.25 Given a DCES $\mathcal{D} = (\{\mathsf{U}_1, \mathsf{U}_2\}, \{\mathsf{V}_1, \mathsf{V}_2\}, X, \{\mathsf{Y}_1, \mathsf{Y}_2\}, \{\mathsf{Z}_1, \mathsf{Z}_2\}, f, g, h, X_0)$ with $\alpha(\mathsf{U}_2) = \alpha(\mathsf{Y}_2)$ and $\alpha(\mathsf{V}_2) = \alpha(\mathsf{Z}_2)$, we like to model the discrete feedback connection of $\mathcal{D}$. Since Definition 4.5 does not allow $\mathsf{U}_2 = \mathsf{Y}_2$ and $\mathsf{V}_2 = \mathsf{Z}_2$, we cannot choose the names in that way and simply apply the parallel interconnection operator to $\mathcal{D}$.

Therefore we define a second DCES $\mathcal{D}_{\mathsf{id}} = (\{\mathsf{Y}_2\}, \{\mathsf{Z}_2\}, \{x\}, \{\mathsf{U}_2\}, \{\mathsf{V}_2\}, f_{\mathsf{id}}, g_{\mathsf{id}}, h_{\mathsf{id}}, \{x\})$ by $f(x, u, v) = \{x\}$, $g(x, u)(\mathsf{U}_2) = u(\mathsf{Y}_2)$, and $h(x, x, v)(\mathsf{V}_2) = v(\mathsf{Z}_2)$, for all $u \in \Sigma_{\mathsf{Y}_2}$, $v \in \Sigma_{\mathsf{Z}_2}$. Obviously, the behavior relation of $\mathcal{S}_{\mathcal{D}_{\mathsf{id}}}$ is

$$\begin{aligned} S_{\mathcal{D}_{\mathsf{id}}} = \{&(s_{\mathsf{Y}_2}, s_{\mathsf{Z}_2}, s_{\mathsf{U}_2}, s_{\mathsf{V}_2}) \in \mathrm{Beh}(\{\mathsf{Y}_2\}, \{\mathsf{Z}_2\}, \{\mathsf{U}_2\}, \{\mathsf{V}_2\}) \mid \\ &\forall t \in \mathbb{R}_{\geq 0} : s_{\mathsf{U}_2}(t)(\mathsf{U}_2) = s_{\mathsf{Y}_2}(t)(\mathsf{Y}_2) \wedge s_{\mathsf{V}_2}(t)(\mathsf{V}_2) = s_{\mathsf{Z}_2}(t)(\mathsf{Z}_2)\}, \end{aligned}$$

so at each moment, the condition, respectively event, output symbol equals the condition, respectively event, input symbol. A block diagram for the connection of $\mathcal{D}$ and $\mathcal{D}_{\mathsf{id}}$ is shown in Figure 4.4 (compare this with Figure 3.3).

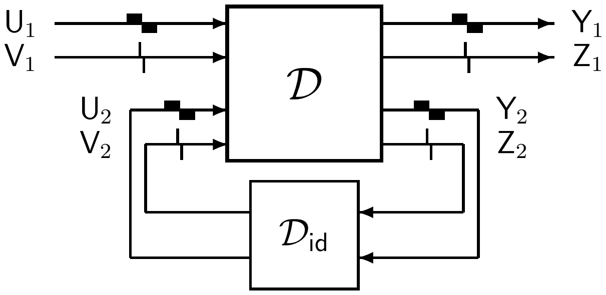


Figure 4.4: Modeling feedback by parallel using $\mathcal{D}_{\mathsf{id}}$.

Then we apply the parallel interconnection operator to $\mathcal{D}$ and $\mathcal{D}_{\mathsf{id}}$ and hide the signals between $\mathcal{D}_{\mathsf{id}}$ and $\mathcal{D}$: $\mathcal{D}' = \mathrm{hide}(\mathcal{D} \| \mathcal{D}_{\mathsf{id}}, \{\mathsf{U}_2, \mathsf{V}_2, \mathsf{Y}_2, \mathsf{Z}_2\})$. As result, $\mathcal{S}_{\mathcal{D}'}$ is a feedback connection of $\mathcal{S}_{\mathcal{D}}$. $\diamond$

Since we only allow unique solutions for the values in the feedback loop, the usefulness of this feedback construction is questionable; more interesting systems arise when we allow more than one solution. In the next section, we will redefine the discrete parallel interconnection operator to work with multiple solutions.

4.6 Weakening the Unique Solution Property

Now we want to allow more than one solution for the signals between the systems. Before we start, we show in the following example why we need more than one initial state in DCESs to define a parallel operator with multiple solutions.

Example 4.26 Consider the DCES $\mathcal{D} = (\{\mathsf{U}\}, \emptyset, \{x\}, \{\mathsf{Y}\}, \emptyset, f, g, h, \{x\})$ with $\alpha(\mathsf{U}) = \alpha(\mathsf{Y}) = \{\mathrm{a}, \mathrm{b}\}$ and, for all $u \in \Sigma_{\mathsf{U}}$,

- $f(x, u, \emptyset) = \{x\}$,
- $g(x, u)(\mathsf{Y}) = u(\mathsf{U})$, and
- $h(x, x, \emptyset) = \emptyset$.

This system has just one state x, no event components, and copies its condition input symbol (which can be a or b) to its condition output. Now we would like to construct a DCES $\mathcal{D}' = (\emptyset, \emptyset, X, \{\mathsf{Y}\}, \emptyset, f', g', h', X_0)$ which describes the connection of $\mathcal{D}$'s input component U to its output component Y

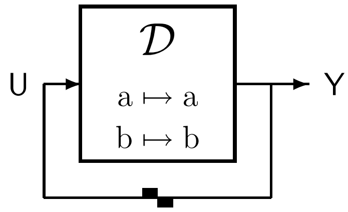


Figure 4.5: A system with two solutions for the feedback value.

(this is a feedback connection without events and without external inputs), cf. Figure 4.5.

Since we want multiple solutions this time, the DCES $\mathcal{D}'$ should have a as well as b as a solution for the feedback value. So for all $t \in \mathbb{R}_{\geq 0}$, we want to find both possibilities $(\emptyset, \emptyset, s_{\mathsf{Y}}, \emptyset) \in S_{\mathcal{D}'}$ with $s_{\mathsf{Y}}(t)(\mathsf{Y}) = \mathrm{a}$ *and* $(\emptyset, \emptyset, s_{\mathsf{Y}}, \emptyset) \in S_{\mathcal{D}'}$ with $s_{\mathsf{Y}}(t)(\mathsf{Y}) = \mathrm{b}$.

If we have just one initial state x_0, we cannot find both possibilities for time 0. Every run of $\mathcal{D}'$ starts at time 0 with this state x_0, leading to $s_{\mathsf{Y}}(0) = g'(x_0, \emptyset)$. Since g' allows no nondeterministic choice, only one value for $s_{\mathsf{Y}}(0)(\mathsf{Y})$ is possible.

If we allow two initial states x_1 and x_2, we can define $g'(x_1, \emptyset)(\mathsf{Y}) = \mathrm{a}$ and $g'(x_2, \emptyset)(\mathsf{Y}) = \mathrm{b}$, allowing both solutions for s_{Y}. $\diamond$

For the remainder of this section, let $n \in \mathbb{N}$, and, for all $i \in \{1, \ldots, n\}$, let $\mathcal{D}_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, X_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, f_i, g_i, h_i, X_{0,i})$ be a named DCES.

Note that the following theorem is the same as Theorem 4.13 except that we do *not* demand *unique* solutions.

Theorem 4.27 *Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be consistent. A named DCES $\mathcal{D}$ can be effectively constructed such that $\mathcal{S}_{\mathcal{D}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}, \ldots, \mathcal{S}_{\mathcal{D}_n}$, if the following* multiple solution property *holds for $\mathcal{D}_1, \ldots, \mathcal{D}_n$:*

- *For all $x_1 \in X_1, \ldots, x_n \in X_n$, $u \in \Sigma_{\mathsf{C}}^{\mathsf{in}}$, there exists a $y \in \Sigma_{\mathsf{C}^{\mathsf{io}}}$ such that for all $i \in \{1, \ldots, n\}$, we have*

$$g_i\left(x_i, (u \cup y)|_{\bar{\mathsf{U}}_i}\right)|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}} = y|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}}.$$

- *For all $x_1, x_1' \in X_1, \ldots, x_n, x_n' \in X_n$, $v \in \Sigma_{\mathsf{E}}^{\mathsf{in}}$, there exists a $z \in \Sigma_{\mathsf{E}^{\mathsf{io}}}$ such that for all $i \in \{1, \ldots, n\}$, we have*

$$h_i\left(x_i, x_i', (v \cup z)|_{\bar{\mathsf{V}}_i}\right)|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}} = z|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}}. \qquad \square$$

The construction used in the following proof works as follows. Like in the proof of Theorem 4.13, we use a Cartesian product of the sets of states of $\mathcal{D}_1, \ldots, \mathcal{D}_n$ as the set of states for the composed system $\mathcal{D}$. The main difference to the unique solution case lies in the selection of the solutions that are used for the functions g and h. If we have more than one possibility for an evaluation of C^{io} or E^{io}, one of these evaluations has to be selected nondeterministically. This selection, however, cannot be done in g or h, since these functions do not allow any nondeterministic choice of output evaluations.

We solve this problem by integrating the nondeterministic choice into f. The Cartesian product in the state space of $\mathcal{D}$ is extended by two more sets, one containing all evaluations of C^{io}, and one containing all evaluations of E^{io}. So these two sets contain all possible solutions demanded in the multiple solution property. Now f is constructed in such a way that the last two components of the state (these contain the evaluations) change arbitrarily during each transition; this is where the nondeterministic choice takes place. The output functions g and h can now use these last two components of the state to produce their output. Such an extension of the state space is used by a similar strategy in [SK91] for the feedback connection with multiple solutions.

It may be the case that the evaluation taken from the state's last two components is *not* an allowed solution, since *all* evaluations over C^{io} and E^{io} are contained in the state space, and not every evaluation has to be a solution in each situation. In this case, we fall back on an evaluation which has been picked from the set of solutions beforehand.

Proof Let $\mathcal{D}_1, \ldots, \mathcal{D}_n$ be given as above, let the multiple solution property hold, and let $G(x, u) \subseteq \Sigma_{\mathsf{C}^{\mathsf{io}}}$, respectively, $H(x, x', v) \subseteq \Sigma_{\mathsf{E}^{\mathsf{io}}}$ be the sets of solutions y, respectively, z, for $x, x' \in X_1 \times \ldots \times X_n$, $u \in \Sigma^{\mathsf{in}}_{\mathsf{C}}$, and $v \in \Sigma^{\mathsf{in}}_{\mathsf{E}}$. We fix one element of each of the solution sets, and denote it by $\mathrm{pick}(G(x, u))$, respectively, $\mathrm{pick}(H(x, x', v))$.

We construct a named DCES $\mathcal{D}$ such that $\mathcal{S}_{\mathcal{D}}$ is parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}, \ldots, \mathcal{S}_{\mathcal{D}_n}$. Let $\mathcal{D} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, X, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, f, g, h, X_0)$ be the named DCES with

- component names $\bar{\mathsf{U}} = \mathsf{C}^{\mathsf{in}}$, $\bar{\mathsf{V}} = \mathsf{E}^{\mathsf{in}}$, $\bar{\mathsf{Y}} = \mathsf{C}^{\mathsf{out}}$, $\bar{\mathsf{Z}} = \mathsf{E}^{\mathsf{out}}$,
- states $X = X_1 \times \ldots \times X_n \times \Sigma_{\mathsf{C}^{\mathsf{io}}} \times \Sigma_{\mathsf{E}^{\mathsf{io}}}$,
- initial states $X_0 = X_{0,1} \times \ldots \times X_{0,n} \times \Sigma_{\mathsf{C}^{\mathsf{io}}} \times \Sigma_{\mathsf{E}^{\mathsf{io}}}$,

and, for all states $x = (x_1, \ldots, x_n, y, z) \in X$, $x' = (x'_1, \ldots, x'_n, y', z') \in X$, input condition evaluations $u \in \Sigma_{\bar{\mathsf{U}}}$, and input event evaluations $v \in \Sigma_{\bar{\mathsf{V}}}$ (we use $\bar{x}$, resp. $\bar{x}'$, as an abbreviation for $(x_1, \ldots, x_n)$, resp. $(x'_1, \ldots, x'_n)$),

- $x' \in f(x,u,v)$ if and only $x'_i \in f_i\Big(x_i, (u \cup g(x,u))_{|\bar{\mathsf{U}}_i}, (v \cup h(x,x',v))_{|\bar{\mathsf{V}}_i}\Big)$, for all $i \in \{1,\dots,n\}$ (note that there are no restrictions to y' and z'; they are selected arbitrarily),

- $g(x,u) = \bigcup_{i=1}^n g_i\Big(x_i, (u \cup \hat{y}(x,u))_{|\bar{\mathsf{U}}_i}\Big)$,
  where $\hat{y}(x,u) = \begin{cases} y & \text{, if } y \in G(\bar{x},u) \\ \mathrm{pick}(G(\bar{x},u)), & \text{if } y \notin G(\bar{x},u) \end{cases}$, and

- $h(x,x',v) = \bigcup_{i=1}^n h_i\Big(x_i, x'_i, (v \cup \hat{z}(x,x',v))_{|\bar{\mathsf{V}}_i}\Big)$, where $\hat{z}(x,x',v)$
  $= \begin{cases} \mathbf{0} & \text{, if } (x = x') \wedge (v = \mathbf{0}) \\ z' & \text{, if } ((x \neq x') \vee (v \neq \mathbf{0})) \wedge z' \in H(\bar{x},\bar{x}',v) \\ \mathrm{pick}(H(\bar{x},\bar{x}',v)), & \text{if } ((x \neq x') \vee (v \neq \mathbf{0})) \wedge z' \notin H(\bar{x},\bar{x}',v) \end{cases}$.

Note that pick$(\cdot)$ has been fixed initially; so there is no nondeterministic choice in g or h (this takes place in f). Further note that the unions in the definitions of g and h are well-defined, since $\{\mathcal{D}_1,\dots,\mathcal{D}_n\}$ is consistent and therefore, there is no interference of the single systems' output functions.

We show that $\mathcal{D}$ is well-behaved. Let $x = (x_1,\dots,x_n,y,z) \in X$, $u \in \Sigma_{\bar{\mathsf{U}}}$. *Output triggering:* We have to show that $h(x,x,\mathbf{0}) = \mathbf{0}$. Since all $\mathcal{D}_i$ are well-behaved (output triggering), we have

$$\begin{aligned} h(x,x,\mathbf{0}_{\bar{\mathsf{V}}}) &= \textstyle\bigcup_{i=1}^n h_i\Big(x_i, x_i, (\mathbf{0}_{\bar{\mathsf{V}}} \cup \hat{z}(x,x,\mathbf{0}_{\bar{\mathsf{V}}}))_{|\bar{\mathsf{V}}_i}\Big) \\ &= \textstyle\bigcup_{i=1}^n h_i(x_i, x_i, (\mathbf{0}_{\bar{\mathsf{V}}} \cup \mathbf{0}_{\mathsf{E}^{\mathrm{io}}})_{|\bar{\mathsf{V}}_i}) \\ &= \textstyle\bigcup_{i=1}^n h_i(x_i, x_i, \mathbf{0}_{\bar{\mathsf{V}}_i}) = \bigcup_{i=1}^n \mathbf{0}_{\bar{\mathsf{Z}}_i} = \mathbf{0}_{\bar{\mathsf{Z}}}. \end{aligned}$$

Stuttering: We have to show that $x \in f(x,u,\mathbf{0})$. Let $i \in \{1,\dots,n\}$. Since $\mathcal{D}_i$ is well-behaved (stuttering) and $h(x,x,\mathbf{0}) = \mathbf{0}$ (see above), we have

$$\begin{aligned} x_i &\in f_i\Big(x_i, (u \cup g(x,u))_{|\bar{\mathsf{U}}_i}, \mathbf{0}_{\bar{\mathsf{V}}_i}\Big) \\ &= f_i\Big(x_i, (u \cup g(x,u))_{|\bar{\mathsf{U}}_i}, (\mathbf{0}_{\bar{\mathsf{V}}} \cup h(x,x,\mathbf{0}_{\bar{\mathsf{V}}}))_{|\bar{\mathsf{V}}_i}\Big). \end{aligned}$$

So $\mathcal{D}$ is well-behaved.

Now we prove that $\mathcal{S}_{\mathcal{D}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1},\dots,\mathcal{S}_{\mathcal{D}_n}$. Condition 1 of Definition 4.10 holds obviously by Definition 4.12. For proving condition 2, let $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \mathrm{Beh}(\bar{\mathsf{U}},\bar{\mathsf{V}},\bar{\mathsf{Y}},\bar{\mathsf{Z}})$.

First we prove the implication from left to right. Let $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S_{\mathcal{D}}$.

This implies by Definition 4.7 that we have a run $r : \mathbb{R}_{\geq 0} \to X$ (right-continuous, finite-variable) of $\mathcal{D}$ such that

1. $r(t) \in f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))$ for all $t \in \mathbb{R}_{>0}$,
2. $s_{\bar{\mathsf{Y}}}(t) = g(r(t), s_{\bar{\mathsf{U}}}(t))$ for all $t \in \mathbb{R}_{\geq 0}$,
3. $s_{\bar{\mathsf{Z}}}(t) = h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))$ for all $t \in \mathbb{R}_{>0}$, and
4. $r(0) \in X_0$.

We construct the signals for the single systems, which are existentially quantified in condition 2 of Definition 4.10. For all $i \in \{1, \dots, n\}$, let

$$s_{\bar{\mathsf{U}}_i} = (s_{\bar{\mathsf{U}}} \cup s_{\bar{\mathsf{Y}}})|_{\bar{\mathsf{U}}_i},\ s_{\bar{\mathsf{V}}_i} = (s_{\bar{\mathsf{V}}} \cup s_{\bar{\mathsf{Z}}})|_{\bar{\mathsf{V}}_i},\ s_{\bar{\mathsf{Y}}_i} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i},\ s_{\bar{\mathsf{Z}}_i} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i}.$$

Conditions 2(a) and 2(b) of Definition 4.10 are fulfilled obviously.

It remains to prove $(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_{\mathcal{D}_i}$ for all $i \in \{1, \dots, n\}$. Let $i \in \{1, \dots, n\}$. We project the run r of $\mathcal{D}$ on its ith component to get a run of $\mathcal{D}_i$: Define $r_i : \mathbb{R}_{\geq 0} \to X_i$ by $r_i(t) = r(t)_i$ for all $t \in \mathbb{R}_{\geq 0}$. Obviously, r_i is right-continuous and finite-variable, since r is. For proving that r_i really is a run of $\mathcal{D}_i$ over $(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i})$, we have to show

1. $r_i(t) \in f_i(r_i(t^-), s_{\bar{\mathsf{U}}_i}(t^-), s_{\bar{\mathsf{V}}_i}(t))$ for all $t \in \mathbb{R}_{>0}$,
2. $s_{\bar{\mathsf{Y}}_i}(t) = g_i(r_i(t), s_{\bar{\mathsf{U}}_i}(t))$ for all $t \in \mathbb{R}_{\geq 0}$,
3. $s_{\bar{\mathsf{Z}}_i}(t) = h_i(r_i(t^-), r_i(t), s_{\bar{\mathsf{V}}_i}(t))$ for all $t \in \mathbb{R}_{>0}$, and
4. $r_i(0) \in X_{0,i}$.

Claim 1 holds: For all $t \in \mathbb{R}_{>0}$, we have

$$\begin{aligned} r_i(t) = r(t)_i &\in f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))_i \\ &= f_i\Big(r(t^-)_i, \Big(s_{\bar{\mathsf{U}}}(t^-) \cup g(r(t^-), s_{\bar{\mathsf{U}}}(t^-))\Big)|_{\bar{\mathsf{U}}_i}, \\ &\qquad \Big(s_{\bar{\mathsf{V}}}(t) \cup h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))\Big)|_{\bar{\mathsf{V}}_i}\Big) \\ &= f_i\Big(r(t^-)_i, (s_{\bar{\mathsf{U}}}(t^-) \cup s_{\bar{\mathsf{Y}}}(t^-))|_{\bar{\mathsf{U}}_i}, (s_{\bar{\mathsf{V}}}(t) \cup s_{\bar{\mathsf{Z}}}(t))|_{\bar{\mathsf{V}}_i}\Big) \\ &= f_i(r_i(t^-), s_{\bar{\mathsf{U}}_i}(t^-), s_{\bar{\mathsf{V}}_i}(t)). \end{aligned}$$

Claim 2 holds: For all $t \in \mathbb{R}_{\geq 0}$, we have

$$\begin{aligned} s_{\bar{\mathsf{Y}}_i}(t) = s_{\bar{\mathsf{Y}}}(t)|_{\bar{\mathsf{Y}}_i} &= g(r(t), s_{\bar{\mathsf{U}}}(t))|_{\bar{\mathsf{Y}}_i} \\ &= g_i\Big(r(t)_i, \Big(s_{\bar{\mathsf{U}}}(t) \cup \hat{y}(r(t), s_{\bar{\mathsf{U}}}(t))\Big)|_{\bar{\mathsf{U}}_i}\Big) \\ &= g_i\Big(r(t)_i, (s_{\bar{\mathsf{U}}}(t) \cup s_{\bar{\mathsf{Y}}}(t))|_{\bar{\mathsf{U}}_i}\Big) \\ &= g_i(r_i(t), s_{\bar{\mathsf{U}}_i}(t)). \end{aligned}$$

In the equation above, the multiple solution property and $\hat{y}(r(t), s_{\bar{\mathsf{U}}}(t))$ being one of the solutions implies $\hat{y}(r(t), s_{\bar{\mathsf{U}}}(t))|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{io}}} = s_{\bar{\mathsf{Y}}}(t)|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{io}}}$.

Claim 3 holds: For all $t \in \mathbb{R}_{>0}$, we have

$$\begin{aligned} s_{\bar{\mathsf{Z}}_i}(t) = s_{\bar{\mathsf{Z}}}(t)|_{\bar{\mathsf{Z}}_i} &= h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))|_{\bar{\mathsf{Z}}_i} \\ &= h_i\Big(r(t^-)_i, r(t)_i, \Big(s_{\bar{\mathsf{V}}}(t) \cup \hat{z}(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))\Big)|_{\bar{\mathsf{V}}_i}\Big) \\ &= h_i\Big(r(t^-)_i, r(t)_i, (s_{\bar{\mathsf{V}}}(t) \cup s_{\bar{\mathsf{Z}}}(t))|_{\bar{\mathsf{V}}_i}\Big) \\ &= h_i(r_i(t^-), r_i(t), s_{\bar{\mathsf{V}}_i}(t)). \end{aligned}$$

In the equation above, the multiple solution property and $\hat{z}(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))$ being one of the solutions implies $\hat{z}(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{io}}} = s_{\bar{\mathsf{Z}}}(t)|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{io}}}$.

Claim 4 holds: $r_i(0) = r(0)_i \in (X_0)_i = X_{0,i}$.

So r_i is a run of $\mathcal{D}_i$ over $(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i})$. Thus, $(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_{\mathcal{D}_i}$ for all $i \in \{1, \ldots, n\}$.

Now we prove the implication from right to left. Let $(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_{\mathcal{D}_i}$, for all $i \in \{1, \ldots, n\}$, such that conditions 2(a) and 2(b) of Definition 4.10 hold. By Definition 4.7, we find for all $i \in \{1, \ldots, n\}$ a run $r_i : \mathbb{R}_{\geq 0} \to X_i$ (right-continuous, finite-variable) of $\mathcal{D}_i$ such that

1. $r_i(t) \in f_i(r_i(t^-), s_{\bar{\mathsf{U}}_i}(t^-), s_{\bar{\mathsf{V}}_i}(t))$ for all $t \in \mathbb{R}_{>0}$,
2. $s_{\bar{\mathsf{Y}}_i}(t) = g_i(r_i(t), s_{\bar{\mathsf{U}}_i}(t))$ for all $t \in \mathbb{R}_{\geq 0}$,
3. $s_{\bar{\mathsf{Z}}_i}(t) = h_i(r_i(t^-), r_i(t), s_{\bar{\mathsf{V}}_i}(t))$ for all $t \in \mathbb{R}_{>0}$, and
4. $r_i(0) \in X_{0,i}$.

We use the runs r_i and the signals $s_{\bar{\mathsf{Y}}}$ and $s_{\bar{\mathsf{Z}}}$ to construct a run r of $\mathcal{D}$. The main difficulty lies in the construction of r_{n+2}, i.e., the component which suggests the event evaluations for E^{io}. It would be easy just to define $r_{n+2} = s_{\bar{\mathsf{Z}}}|_{\mathsf{E}^{\mathsf{io}}}$, but if there exists a $t \in \mathbb{R}_{>0}$ with $s_{\bar{\mathsf{Z}}}|_{\mathsf{E}^{\mathsf{io}}}(t) \neq \mathbf{0}$, r_{n+2} would not be right-continuous at t, and r would be no run. Therefore, we define $r_{n+2}(t) = s_{\bar{\mathsf{Z}}}|_{\mathsf{E}^{\mathsf{io}}}(t)$ for all $t \in \mathbb{R}_{>0}$ with $s_{\bar{\mathsf{Z}}}(t) \neq \mathbf{0}$, but we also prolong each proper event symbol in r_{n+2} for a finite amount of time $\varepsilon > 0$, i.e., $r_{n+2}(t + x) = s_{\bar{\mathsf{Z}}}|_{\mathsf{E}^{\mathsf{io}}}(t)$ for all $x \in [0, \varepsilon)$. This ε must be small enough not to interfere with other events or transitions taking place.

This needs some preparation. Let $d_0, d_1, \ldots$ be the strictly ordered finite or infinite sequence of all discontinuity points of $s_{\bar{\mathsf{V}}}$, $s_{\bar{\mathsf{Z}}}$, and r_i, for all $i \in \{1, \ldots, n\}$. Note that for any bounded subinterval of $\mathbb{R}_{\geq 0}$, there are only finitely many d_j contained in it. So the d_j are all the points in time where we make a transition in one of the single systems or receive a proper input event or generate a proper output event.

Now define $r : \mathbb{R}_{\geq 0} \to X$ by $r(t) = (r_1(t), \ldots, r_n(t), y(t), z(t))$ with

- $y(t) = s_{\bar{\mathsf{Y}}}(t)|_{\mathsf{C}^{\mathsf{io}}}$, and
- $z(t) = \begin{cases} s_{\bar{\mathsf{Z}}}(d_j)|_{\mathsf{E}^{\mathsf{io}}}, \text{ if } t \in [d_j, \frac{d_j+d_{j+1}}{2}) \text{ for a } j \in \mathbb{N} \\ \mathbf{0}_{\mathsf{E}^{\mathsf{io}}} \qquad\quad , \text{ otherwise} \end{cases}$,

for all $t \in \mathbb{R}_{\geq 0}$. Note that z is right-continuous and finite-variable, and that for all d_j, we have $z(d_j) = s_{\bar{\mathsf{Z}}}(d_j)|_{\mathsf{E}^{\mathsf{io}}}$, and, if $s_{\bar{\mathsf{Z}}}(d_j) \neq \mathbf{0}$, we have $z(d_j^-) = \mathbf{0}$, i.e., d_j is a discontinuity point of z. Obviously, r is right-continuous and finite-variable, since all r_i and $s_{\bar{\mathsf{Y}}}$ are, and since z is by construction.

Now we prove that r is a run of $\mathcal{D}$ over $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}})$. We have to show:

1. $r(t) \in f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))$ for all $t \in \mathbb{R}_{>0}$,
2. $s_{\bar{\mathsf{Y}}}(t) = g(r(t), s_{\bar{\mathsf{U}}}(t))$ for all $t \in \mathbb{R}_{\geq 0}$,
3. $s_{\bar{\mathsf{Z}}}(t) = h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))$ for all $t \in \mathbb{R}_{>0}$, and
4. $r(0) \in X_0$.

Claim 4 holds:

$$\begin{aligned} r(0) &= (r_1(0), \ldots, r_n(0), y(0), z(0)) \\ &\in X_{0,1} \times \ldots \times X_{0,n} \times \Sigma_{\mathsf{C}^{\mathsf{io}}} \times \Sigma_{\mathsf{E}^{\mathsf{io}}} = X_0. \end{aligned}$$

Claim 2 holds: Let $t \in \mathbb{R}_{\geq 0}$. We have $y(t) \in G(\bar{r}(t), s_{\bar{\mathsf{U}}}(t))$, since for all $i \in \{1, \ldots, n\}$ the solution property holds for $y(t)$:

$$\begin{aligned} & g_i\Big(r_i(t), (s_{\bar{\mathsf{U}}}(t) \cup y(t))|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}}\Big)|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}} \\ &= g_i\Big(r_i(t), (s_{\bar{\mathsf{U}}}(t) \cup s_{\bar{\mathsf{Y}}}(t)|_{\mathsf{C}^{\mathsf{io}}})|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}}\Big)|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}} \\ &= g_i(r_i(t), s_{\bar{\mathsf{U}}_i}(t))|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}} \\ &= s_{\bar{\mathsf{Y}}_i}(t)|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}} = s_{\bar{\mathsf{Y}}}(t)|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}} = y(t)|_{\bar{\mathsf{Y}}_i \cap \mathsf{C}^{\mathsf{io}}}. \end{aligned}$$

So $\hat{y}(r(t), s_{\bar{\mathsf{U}}}(t)) = y(t)$, yielding

$$\begin{aligned} s_{\bar{\mathsf{Y}}}(t) &= \textstyle\bigcup_{i=1}^n s_{\bar{\mathsf{Y}}_i}(t) = \bigcup_{i=1}^n g_i(r_i(t), s_{\bar{\mathsf{U}}_i}(t)) \\ &= \textstyle\bigcup_{i=1}^n g_i\Big(r_i(t), (s_{\bar{\mathsf{U}}}(t) \cup s_{\bar{\mathsf{Y}}}(t))|_{\bar{\mathsf{U}}_i}\Big) \\ &= \textstyle\bigcup_{i=1}^n g_i(r_i(t), s_{\bar{\mathsf{U}}}(t)|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{in}}} \cup s_{\bar{\mathsf{Y}}}(t)|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{io}}}) \\ &= \textstyle\bigcup_{i=1}^n g_i(r_i(t), s_{\bar{\mathsf{U}}}(t)|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{in}}} \cup y(t)|_{\bar{\mathsf{U}}_i \cap \mathsf{C}^{\mathsf{io}}}) \\ &= \textstyle\bigcup_{i=1}^n g_i\Big(r_i(t), (s_{\bar{\mathsf{U}}}(t) \cup y(t))|_{\bar{\mathsf{U}}_i}\Big) \\ &= \textstyle\bigcup_{i=1}^n g_i\Big(r_i(t), \Big(s_{\bar{\mathsf{U}}}(t) \cup \hat{y}(r(t), s_{\bar{\mathsf{U}}}(t))\Big)|_{\bar{\mathsf{U}}_i}\Big) \\ &= g(r(t), s_{\bar{\mathsf{U}}}(t)). \end{aligned}$$

For proving claim 3, we do a case distinction to separate the different cases in the definition of $\hat{z}$. Let $t \in \mathbb{R}_{>0}$.

Case A: $r(t^-) = r(t)$ and $s_{\bar{\mathsf{V}}}(t) = \mathbf{0}$. By well-behavedness (output triggering) of $\mathcal{D}$, we have $h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t)) = \mathbf{0}$. By construction of z and since $z(t^-) = z(t)$, we have $s_{\bar{\mathsf{Z}}}(t) = \mathbf{0}$. This implies $s_{\bar{\mathsf{Z}}}(t) = h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))$.

Case B: $r(t^-) \neq r(t)$ or $s_{\bar{\mathsf{V}}}(t) \neq \mathbf{0}$. We prove claim 3 similarly as claim 2: By construction of z, we have $z(t) = s_{\bar{\mathsf{Z}}}(t)|_{\mathsf{E}^{\mathsf{io}}}$. Furthermore, we have $z(t) \in H(\bar{r}(t^-), \bar{r}(t), s_{\bar{\mathsf{V}}}(t))$, since for all $i \in \{1, \ldots, n\}$ the solution property holds for $z(t)$:

$$\begin{aligned}
&h_i\Big(r_i(t^-), r_i(t), (s_{\bar{\mathsf{V}}}(t) \cup z(t))|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}}\Big)|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}} \\
&= h_i\Big(r_i(t^-), r_i(t), (s_{\bar{\mathsf{V}}}(t) \cup s_{\bar{\mathsf{Z}}}(t)|_{\mathsf{E}^{\mathsf{io}}})|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}}\Big)|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}} \\
&= h_i(r_i(t^-), r_i(t), s_{\bar{\mathsf{V}}_i}(t))|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}} \\
&= s_{\bar{\mathsf{Z}}_i}(t)|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}} = s_{\bar{\mathsf{Z}}}(t)|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}} = z(t)|_{\bar{\mathsf{Z}}_i \cap \mathsf{E}^{\mathsf{io}}}.
\end{aligned}$$

So $\hat{z}(r(t^-), r(t), s_{\bar{\mathsf{Z}}}(t)) = z(t)$, yielding

$$\begin{aligned}
s_{\bar{\mathsf{Z}}}(t) &= \textstyle\bigcup_{i=1}^n s_{\bar{\mathsf{Z}}_i}(t) = \bigcup_{i=1}^n h_i(r_i(t^-), r_i(t), s_{\bar{\mathsf{V}}_i}(t)) \\
&= \textstyle\bigcup_{i=1}^n h_i\Big(r_i(t^-), r_i(t), (s_{\bar{\mathsf{V}}}(t) \cup s_{\bar{\mathsf{Z}}}(t))|_{\bar{\mathsf{V}}_i}\Big) \\
&= \textstyle\bigcup_{i=1}^n h_i(r_i(t^-), r_i(t), s_{\bar{\mathsf{V}}}(t)|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{in}}} \cup s_{\bar{\mathsf{Z}}}(t)|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{io}}}) \\
&= \textstyle\bigcup_{i=1}^n h_i(r_i(t^-), r_i(t), s_{\bar{\mathsf{V}}}(t)|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{in}}} \cup z(t)|_{\bar{\mathsf{V}}_i \cap \mathsf{E}^{\mathsf{io}}}) \\
&= \textstyle\bigcup_{i=1}^n h_i\Big(r_i(t^-), r_i(t), (s_{\bar{\mathsf{V}}}(t) \cup z(t))|_{\bar{\mathsf{V}}_i}\Big) \\
&= \textstyle\bigcup_{i=1}^n h_i\Big(r_i(t^-), r_i(t), \Big(s_{\bar{\mathsf{V}}}(t) \cup \hat{z}(r_i(t^-), r(t), s_{\bar{\mathsf{V}}}(t))\Big)|_{\bar{\mathsf{V}}_i}\Big) \\
&= h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t)).
\end{aligned}$$

Finally, claim 1 holds: Since claims 2 and 3 hold, we have for all $t \in \mathbb{R}_{>0}$:

$$\begin{aligned}
r(t)_i &= r_i(t) \\
&\in f_i(r_i(t^-), s_{\bar{\mathsf{U}}_i}(t^-), s_{\bar{\mathsf{V}}_i}(t)) \\
&= f_i\Big(r_i(t^-), (s_{\bar{\mathsf{U}}}(t^-) \cup s_{\bar{\mathsf{Y}}}(t^-))|_{\bar{\mathsf{U}}_i}, (s_{\bar{\mathsf{V}}}(t) \cup s_{\bar{\mathsf{Z}}}(t))|_{\bar{\mathsf{V}}_i}\Big) \\
&= f_i\Big(r_i(t^-), \Big(s_{\bar{\mathsf{U}}}(t^-) \cup g(r(t^-), s_{\bar{\mathsf{U}}}(t^-))\Big)|_{\bar{\mathsf{U}}_i}, \\
&\qquad\qquad \Big(s_{\bar{\mathsf{V}}}(t) \cup h(r(t^-), r(t), s_{\bar{\mathsf{V}}}(t))\Big)|_{\bar{\mathsf{V}}_i}\Big) \\
&= f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))_i,
\end{aligned}$$

for $i \in \{1, \ldots, n\}$, and

$$r(t)_{n+1} = y(t) \in \Sigma_{\mathsf{C}^{\mathsf{io}}} = f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))_{n+1},$$

$$r(t)_{n+2} = z(t) \in \Sigma_{\mathsf{E}^{\mathsf{io}}} = f(r(t^-), s_{\bar{\mathsf{U}}}(t^-), s_{\bar{\mathsf{V}}}(t))_{n+2}.$$

So r is a run of $\mathcal{D}$ over $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}})$. Thus, $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S_{\mathcal{D}}$. Condition 2 of Definition 4.10 has been proven, so $\mathcal{S}_{\mathcal{D}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}, \ldots, \mathcal{S}_{\mathcal{D}_n}$. ∎

Definition 4.28 (Discrete parallel interconnection with multiple solutions) *Let $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ be consistent, and let the multiple solution property of Theorem 4.27 hold. The named DCES $\mathcal{D}$ gained from the construction above is called the* discrete parallel interconnection with multiple solutions *of $\mathcal{D}_1, \ldots, \mathcal{D}_n$, denoted by $\mathcal{D}_1 \,|||\, \ldots \,|||\, \mathcal{D}_n$.* □

Remark 4.29 To reduce the number of states and transitions in the resulting DCES in the construction above, one can

- reduce the states to those with $y \in \bigcup_{u \in \Sigma_{\mathsf{C}}^{\mathsf{in}}} G(\bar{x}, u)$ since these y are the only possible choices for a condition output evaluation in that state (this can be done for z in a similar way), and
- reduce the set of initial states X_0 to states with $z = \mathbf{0}$, since z is only used to make a nondeterministic choice of the event output evaluation when *entering* a state (however, this reduces neither the state space X nor the number of transitions).

We do not apply these reductions, since they make the proof of Theorem 4.27 harder to write and to read. For the implementation, however, if we aim towards model checking, space reduction might become vital. ◇

We conclude this chapter by giving an example for a discrete parallel interconnection having multiple solutions.

Example 4.30 Reconsider case 3 of Example 4.14. We have two interconnected DCES $\mathcal{D}_1$ and $\mathcal{D}_2$. For $i \in \{1, 2\}$, we have the named DCES $\mathcal{D}_i = (\{\mathsf{U}_i\}, \emptyset, \{x\}, \{\mathsf{Y}_i\}, \emptyset, f_i, g_i, h_i, \{x\})$ with $\alpha(\mathsf{U}_i) = \alpha(\mathsf{Y}_i) = \{\mathrm{a}, \mathrm{b}\}$, and, for all $u \in \Sigma_{\mathsf{U}_i}$,

- $f_i(x, u, \emptyset) = \{x\}$,
- $g_i(x, u)(\mathsf{Y}_i) = u(\mathsf{U}_i)$, and
- $h_i(x, x, \emptyset) = \emptyset$.

As formal semantics of $\mathcal{D}_i$, we have the CES $\mathcal{S}_{\mathcal{D}_i} = (\{\mathsf{U}_i\}, \emptyset, \{\mathsf{Y}_i\}, \emptyset, S_{\mathcal{D}_i})$ with

$$\begin{aligned} S_{\mathcal{D}_i} = \{(s_{\mathsf{U}_i}, \emptyset, s_{\mathsf{Y}_i}, \emptyset) \in \mathrm{Beh}(\{\mathsf{U}_i\}, \emptyset, \{\mathsf{Y}_i\}, \emptyset) \mid \\ \forall t \in \mathbb{R}_{\geq 0} : s_{\mathsf{U}_i}(t)(\mathsf{U}_i) = s_{\mathsf{Y}_i}(t)(\mathsf{Y}_i)\}, \end{aligned}$$

for $i \in \{1, 2\}$. In both systems, the condition output symbol equals the condition input symbol at every moment.

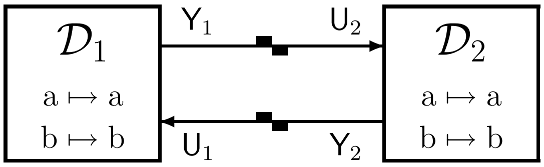


Figure 4.6: A parallel interconnection of DCESs with multiple solutions.

The connections between $\mathcal{D}_1$ and $\mathcal{D}_2$ are characterized by $\mathsf{U}_1 = \mathsf{Y}_2 \neq \mathsf{Y}_1 = \mathsf{U}_2$, leading to the block diagram shown in Figure 4.6. We have no external inputs, so $\mathsf{C}^{\mathsf{in}} = \mathsf{E}^{\mathsf{in}} = \emptyset$. Furthermore, we have $\mathsf{C}^{\mathsf{io}} = \mathsf{C}^{\mathsf{out}} = \{\mathsf{U}_1, \mathsf{U}_2\}$ and $\mathsf{E}^{\mathsf{io}} = \mathsf{E}^{\mathsf{out}} = \emptyset$.

Checking the multiple solution property of Theorem 4.27, we get the solution sets $G((x, x), \emptyset) = \{\{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}, \{\mathsf{U}_1 \mapsto \mathrm{b}, \mathsf{U}_2 \mapsto \mathrm{b}\}\}$ and $H((x, x), (x, x), \emptyset) = \{\emptyset\}$. Now we use Definition 4.28 and construct the discrete parallel interconnection $\mathcal{D} = \mathcal{D}_1 \,|||\, \mathcal{D}_2$. In the construction, we have to fix one element of each solution sets; we take $\mathrm{pick}(G((x, x), \emptyset)) = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}$ and $\mathrm{pick}(H((x, x), (x, x), \emptyset)) = \emptyset$.

We get the DCES $\mathcal{D} = (\emptyset, \emptyset, X, \{\mathsf{U}_1, \mathsf{U}_2\}, \emptyset, f, g, h, X_0)$ with

- $\begin{aligned} X = X_0 &= \{x\} \times \{x\} \times \Sigma_{\{\mathsf{U}_1, \mathsf{U}_2\}} \times \Sigma_\emptyset \\ &= \Big\{(x, x, \{\mathsf{U}_1 \mapsto y_1, \mathsf{U}_2 \mapsto y_2\}, \emptyset) \,|\, y_1, y_2 \in \{\mathrm{a}, \mathrm{b}\}\Big\}, \end{aligned}$
- $f(\bar{x}, \emptyset, \emptyset) = X$ for all $\bar{x} \in X$,
- $g((x, x, u, \emptyset), \emptyset) = \begin{cases} \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}, \text{ if } u = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\} \\ \{\mathsf{U}_1 \mapsto \mathrm{b}, \mathsf{U}_2 \mapsto \mathrm{b}\}, \text{ if } u = \{\mathsf{U}_1 \mapsto \mathrm{b}, \mathsf{U}_2 \mapsto \mathrm{b}\} \\ \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}, \text{ if } u = \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{b}\} \\ \{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}, \text{ if } u = \{\mathsf{U}_1 \mapsto \mathrm{b}, \mathsf{U}_2 \mapsto \mathrm{a}\} \end{cases}$, and
- $h(\bar{x}, \bar{x}', \emptyset) = \emptyset$ for all $\bar{x}, \bar{x}' \in X$.

Figure 4.7 shows a block diagram of $\mathcal{D}$. There are four control states which can change arbitrarily; in one state the condition output $\{\mathsf{U}_1 \mapsto \mathrm{b}, \mathsf{U}_2 \mapsto \mathrm{b}\}$ is generated, and $\{\mathsf{U}_1 \mapsto \mathrm{a}, \mathsf{U}_2 \mapsto \mathrm{a}\}$ in the other three states.

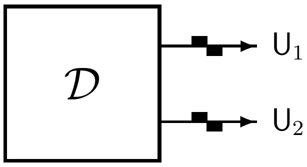


Figure 4.7: Block diagram of $\mathcal{D} = \mathcal{D}_1 \mathbin{|||} \mathcal{D}_2$.

As semantics of the DCES $\mathcal{D}$, we get the CES $\mathcal{S}_{\mathcal{D}} = (\emptyset, \emptyset, \{\mathsf{U}_1, \mathsf{U}_2\}, \emptyset, S_{\mathcal{D}})$ with

$$\begin{aligned} S_{\mathcal{D}} = \{ & (\emptyset, \emptyset, s_{\{\mathsf{U}_1,\mathsf{U}_2\}}, \emptyset) \in \mathrm{Beh}(\emptyset, \emptyset, \{\mathsf{U}_1, \mathsf{U}_2\}, \emptyset) \mid \\ & \forall t \in \mathbb{R}_{\geq 0} : s_{\{\mathsf{U}_1,\mathsf{U}_2\}}(t)(\mathsf{U}_1) = s_{\{\mathsf{U}_1,\mathsf{U}_2\}}(t)(\mathsf{U}_2)\}, \end{aligned}$$

i.e., the two output condition symbols of $\mathcal{S}_{\mathcal{D}}$ may change arbitrarily (in a finite-variable fashion), but they have to be equal in each time point. This is the behavior we expect from a parallel interconnection of $\mathcal{S}_{\mathcal{D}_1}$ and $\mathcal{S}_{\mathcal{D}_1}$ (see Definition 4.10, condition 2). $\diamond$

Chapter 5

Introducing Quantitative Timing

As we have already seen in Example 2.8, the DCES model lacks a possibility of measuring time. This is, however, very important for process control models, where the desired properties of systems often include time constraints.

We could of course use CESs as a model for such purposes, but they cannot be described conveniently in a finite, operational way like DCESs. This fact makes CESs far too abstract, whereas DCESs can easily be implemented due to their operational structure. So it is straightforward to augment DCESs with timing features in such a way that the operational structure is preserved.

This is done by connecting a set of CESs to the DCES. These CESs are able to measure time and transfer timing information to the DCES. The DCES then can use this information to perform time-dependent actions. A CESs which is used for delivering timing information is defined as the semantics of a *timer*, which we introduce next.

5.1 Timers

To add quantitative timing to discrete condition/event systems, we introduce the notion of *timers.* A timer T is characterized by a list of positive integers which serve as thresholds for a real-valued clock. The timer value starts with 0 and increases at the same speed as the global time. The timer can be reset to 0 by sending a reset event to the timer. Besides this reset input (named Res_T), a timer has two output components, a condition component (named Int_T) which states in which interval between the thresholds the timer value is, and an event component (named Ala_T) which outputs an event if the timer value is exactly one of the thresholds.

Definition 5.1 (Timer) *Let $\mathbb{T}$ be a global set of identifiers. Any $T \in \mathbb{T}$ is called* timer. *To each timer T we associate a nonempty, finite list $\Theta(T) = (\theta_T^1, \ldots, \theta_T^{n_T})$ of strictly increasing positive integers, called* thresholds, *and the following names and alphabets:*

- interval *condition name Int_T with alphabet $\alpha(\mathsf{Int}_T) = \{\mathrm{Int}_0, \ldots, \mathrm{Int}_{n_T}\}$,*
- alarm *event name Ala_T with alphabet $\alpha(\mathsf{Ala}_T) = \{\mathbf{0}, \mathrm{Ala}_1, \ldots, \mathrm{Ala}_{n_T}\}$,*
- reset *event name Res_T with alphabet $\alpha(\mathsf{Res}_T) = \{\mathbf{0}, \mathrm{Res}\}$.*

All condition and event symbols are considered to be different. For all $T_1, T_2 \in \mathbb{T}$ with $T_1 \neq T_2$, we demand $\{\mathsf{Int}_{T_1}, \mathsf{Ala}_{T_1}, \mathsf{Res}_{T_1}\} \cap \{\mathsf{Int}_{T_2}, \mathsf{Ala}_{T_2}, \mathsf{Res}_{T_2}\} = \emptyset$, i.e., each timer has its own, unique component names. □

We define a *timer function* which measures the amount of time passed since the last occurrence of any proper event in a given signal. This signal will be the reset signal for the timer. Figure 5.1 shows the timer function where the only reset events are at t_1, t_2, and t_3.

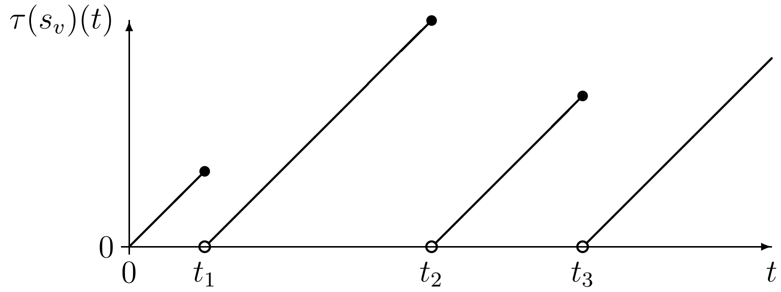


Figure 5.1: Example of a timer function.

Definition 5.2 (Timer function) *Let s_v be any event signal. The* timer function *$\tau(s_v) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with reset input signal s_v is defined by*

$$\tau(s_v)(t) = t - \max\left(\{0\} \cup \{t_0 \in [0, t) \mid s_v(t_0) \neq \mathbf{0}\}\right)$$

for all $t \in \mathbb{R}_{\geq 0}$. □

Note that if there is a proper event at time t in s_v, $\tau(s_v)(t)$ delivers the "old" timer value, and $\tau(s_v)(t+\varepsilon) = \varepsilon$ for all $\varepsilon > 0$ up to the next proper event (cf. Figure 5.1).

Now we can define the semantics of a timer T, which is done by associating a condition/event system $\mathcal{S}_T$ with T.

The behavior of $\mathcal{S}_T$ is as follows: Its event input is the reset signal for the clock of T; whenever $\mathcal{S}_T$ receives a proper event (which is Res), the clock is reset to 0. This is modeled by the timer function τ.

The output condition of $\mathcal{S}_T$ states in which interval between the timer's thresholds the current timer value is. If the timer value is smaller than its first threshold θ_T^1 or if the timer is receiving a reset event, the output condition symbol of $\mathcal{S}_T$ is Int_0. If the timer value is greater or equal than the last threshold $\theta_T^{n_T}$, this is denoted by the condition symbol Int_{n_T}. And if the value of the timer is between θ_T^i and θ_T^{i+1}, $\mathcal{S}_T$ delivers the condition symbol Int_i.

The output event of $\mathcal{S}_T$ states if the timer value is *exactly* one of the thresholds. Whenever a threshold θ_T^i is reached, the alarm event Ala_i is generated. In all other cases, the output event of $\mathcal{S}_T$ is the null symbol. Note that if a reset event occurs exactly at the time when a threshold is reached, $\mathcal{S}_T$ also generates the alarm event for that threshold. Furthermore note that there is no Ala_0 event.

$\mathcal{S}_T$ has no condition input components, so its set of condition input names is empty.

Definition 5.3 (Semantics of a timer) *For a timer $T \in \mathbb{T}$ the semantics of T is given by the condition/event system $\mathcal{S}_T = (\emptyset, \{\mathsf{Res}_T\}, \{\mathsf{Int}_T\}, \{\mathsf{Ala}_T\}, S_T)$ with S_T defined by $(\emptyset, s_v, s_y, s_z) \in S_T$ if and only if for every $t \in \mathbb{R}_{\geq 0}$ we have*

- $$s_y(t)(\mathsf{Int}_T) = \begin{cases} \mathrm{Int}_0 & \textit{, if } s_v(t) \neq \mathbf{0} \;\; \textit{or} \;\; \tau(s_v)(t) < \theta_T^1 \\ \mathrm{Int}_i & \textit{, if } s_v(t) = \mathbf{0} \textit{ and } \tau(s_v)(t) \in [\theta_T^i, \theta_T^{i+1}) \\ \mathrm{Int}_{n_T} & \textit{, if } s_v(t) = \mathbf{0} \textit{ and } \tau(s_v)(t) \geq \theta_T^{n_T} \end{cases} \qquad \textit{and}$$

- $$s_z(t)(\mathsf{Ala}_T) = \begin{cases} \mathrm{Ala}_i, & \textit{if } \tau(s_v)(t) = \theta_T^i \\ \mathbf{0} & \textit{, if } \tau(s_v)(t) \neq \theta_T^i \textit{ for all } i \in \{1, \ldots, n_T\} \end{cases}$$

for all $(\emptyset, s_v, s_y, s_z) \in \mathrm{Beh}(\emptyset, \{\mathsf{Res}_T\}, \{\mathsf{Int}_T\}, \{\mathsf{Ala}_T\})$. □

The following lemma states that timers are really something new; we cannot express them by using a DCES.

Lemma 5.4 *For any timer $T \in \mathbb{T}$, there is no DCES $\mathcal{D}$ with $\mathcal{S}_\mathcal{D} = \mathcal{S}_T$.* □

Proof Let $T \in \mathbb{T}$, and assume that we have a DCES $\mathcal{D} = (\emptyset, \{\mathsf{Res}_T\}, X, \{\mathsf{Int}_T\}, \{\mathsf{Ala}_T\}, f, g, h, X_0)$ with $\mathcal{S}_\mathcal{D} = \mathcal{S}_T$. We construct a run of $\mathcal{D}$ over a behavior s. Let $s_{\mathsf{Res}_T} \in E(\{\mathsf{Res}_T\})$, $s_{\mathsf{Int}_T} C(\{\mathsf{Int}_T\})$, and $s_{\mathsf{Ala}_T} \in E(\{\mathsf{Ala}_T\})$ such that for all $t \in \mathbb{R}_{\geq 0}$, $s_{\mathsf{Res}_T}(t) = \mathbf{0}$, $s_{\mathsf{Int}_T}(t) = g(x_0, \emptyset)$, and $s_{\mathsf{Ala}_T}(t) = \mathbf{0}$.

Obviously, we have $s = (\emptyset, s_{\mathsf{Res}_T}, s_{\mathsf{Int}_T}, s_{\mathsf{Ala}_T}) \in \mathrm{Beh}(\emptyset, \{\mathsf{Res}_T\}, \{\mathsf{Int}_T\}, \{\mathsf{Ala}_T\})$. Let $x_0 \in X_0$, and define $r : \mathbb{R}_{\geq 0} \to X$ by $r(t) = x_0$ for all $t \in \mathbb{R}_{\geq 0}$.

We show that r is run of $\mathcal{D}$ over s: Since we have defined s_{Res_T} in such a way that there are no proper input events, stuttering allows $\mathcal{D}$ to stay always in the same state x_0, and by output triggering, there are no proper output events. And s_{Int_T} directly matches Definition 4.6. So r is a run of $\mathcal{D}$ over s, and therefore, $s \in S_\mathcal{D}$.

Now we show that $s \notin S_T$. Since there are no proper input events, Definition 5.2 yields $\tau(s_{\mathsf{Res}_T})(t) = t$ for all $t \in \mathbb{R}_{\geq 0}$. So by Definition 5.3, we expect an alarm event at the first threshold of T: $s_{\mathsf{Ala}_T}(\theta_T^1)(\mathsf{Ala}_T) = \mathrm{Ala}_1$. This does not hold, so $s \notin S_T$. Thus, $S_\mathcal{D} \neq S_T$ and $\mathcal{S}_\mathcal{D} \neq \mathcal{S}_T$, contradicting our assumption. ■

5.2 Timed Condition/Event Systems

A timed condition/event system is a discrete condition/event system which is augmented with a set of timers. The outputs of the timers are connected to the inputs of the discrete condition/event system and vice versa. The signals between the discrete system and the timers are considered to be internal and are therefore not observable outside the system.

Definition 5.5 (Timed condition/event system) *A timed condition/event system (TCES) $\mathcal{T} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, \bar{T}, \mathcal{D})$ consists of*

- *a finite set of external condition input names $\bar{\mathsf{U}}$,*
- *a finite set of external event input names $\bar{\mathsf{V}}$,*
- *a finite set of external condition output names $\bar{\mathsf{Y}}$,*
- *a finite set of external event output names $\bar{\mathsf{Z}}$,*
- *a finite set of timers $\bar{T} \subseteq \mathbb{T}$, and*
- *a discrete condition/event system $\mathcal{D} = (\bar{\mathsf{U}}^+, \bar{\mathsf{V}}^+, X, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}^+, f, g, h, X_0)$, where*

– $\bar{\mathsf{U}}^+ = \bar{\mathsf{U}} \cup \bigcup_{T \in \bar{T}} \{\mathsf{Int}_T\}$,
– $\bar{\mathsf{V}}^+ = \bar{\mathsf{V}} \cup \bigcup_{T \in \bar{T}} \{\mathsf{Ala}_T\}$, *and*
– $\bar{\mathsf{Z}}^+ = \bar{\mathsf{Z}} \cup \bigcup_{T \in \bar{T}} \{\mathsf{Res}_T\}$.

Furthermore, let all sets of names be disjoint. □

Figure 5.2 shows a block diagram of a timed condition/event system with timer set $\bar{T} = \{T_1, \ldots, T_m\}$.

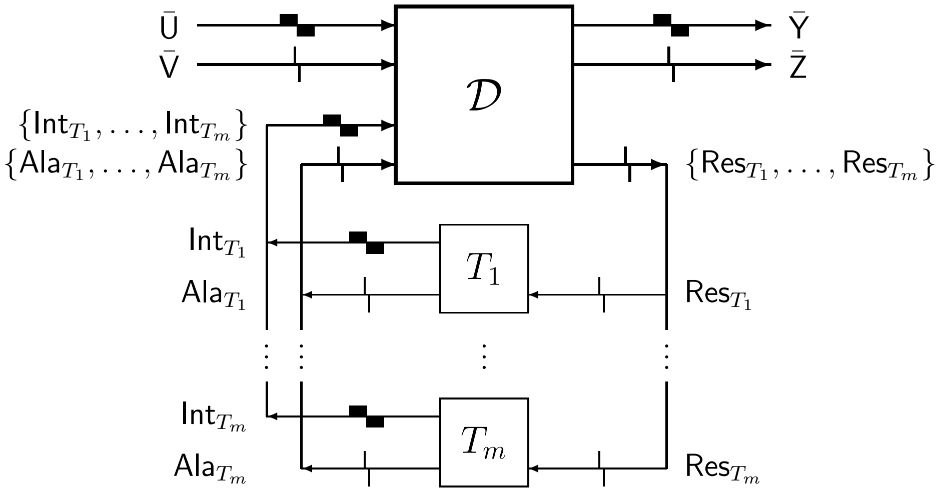


Figure 5.2: Block diagram of a timed condition/event system.

Now we define the semantics of a TCES. This is done by combining the signals of the TCES's discrete system with the signals of the TCES's timers. The components which are shown as connected in Figure 5.2 must have the same signals in $S_{\mathcal{D}}$ and in $S_{T_1}, \ldots, S_{T_m}$.

Definition 5.6 (Semantics of a TCES) *Let $\mathcal{T} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, \bar{T}, \mathcal{D})$ be a timed condition/event system. The* semantics of $\mathcal{T}$ *is defined by the condition/event system $\mathcal{S}_{\mathcal{T}} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, S_{\mathcal{T}})$ with $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S_{\mathcal{T}}$ if and only if there exist $(\emptyset, s_v^T, s_y^T, s_z^T) \in S_T$ for every $T \in \bar{T}$ such that*

$$(s_{\bar{\mathsf{U}}} \cup \bigcup_{T \in \bar{T}} s_y^T, s_{\bar{\mathsf{V}}} \cup \bigcup_{T \in \bar{T}} s_z^T, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}} \cup \bigcup_{T \in \bar{T}} s_v^T) \in S_{\mathcal{D}},$$

for all $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$. □

Obviously, a TCES having no timers is semantically equivalent to the DCES contained in it.

Lemma 5.7 *Let* $\mathcal{T} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, \bar{T}, \mathcal{D})$ *be a timed condition/event system with* $\bar{T} = \emptyset$*. Then we have* $\mathcal{S}_{\mathcal{T}} = \mathcal{S}_{\mathcal{D}}$*.* □

Another interesting but not surprising observation is that this construction is a parallel interconnection of the TCES's discrete part with its timers.

Lemma 5.8 *In Definition 5.6,* $\mathcal{S}_{\mathcal{T}}$ *is a parallel interconnection of* $\mathcal{S}_{\mathcal{D}}$ *and all* $\mathcal{S}_T$*,* $T \in \bar{T}$ *(if we hide the reset outputs of* $\mathcal{S}_{\mathcal{D}}$ *and the timers' outputs).* □

Proof Let $\mathcal{T} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, \bar{T}, \mathcal{D})$ be a timed condition/event system with $\bar{T} = \{T_1, \ldots, T_m\}$, $|\bar{T}| = m$. Define $n = m + 1$, $\mathcal{S}_n = \mathcal{S}_{\mathcal{D}}$, $S_n = S_{\mathcal{D}}$, and, for $i \in \{1, \ldots, n-1\}$, $\mathcal{S}_i = \mathcal{S}_{T_i}$, $S_i = S_{T_i}$. We show that $\mathcal{S}_{\mathcal{T}} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, S_{\mathcal{T}})$ is a parallel interconnection of $\mathcal{S}_1, \ldots, \mathcal{S}_n$, if we hide the reset outputs of $\mathcal{S}_n$ and all outputs of $\mathcal{S}_1, \ldots, \mathcal{S}_{n-1}$.

If we modify condition 1 of Definition 4.10 in accordance with this hiding (by omitting all timer related names in the two output name equations), we get

$$\begin{aligned}
\bar{\mathsf{U}} &= \left(\bigcup_{i=1}^{n-1} \emptyset \cup (\bar{\mathsf{U}} \cup \bigcup_{i=1}^{n-1}\{\mathsf{Int}_{T_i}\})\right) \setminus \left(\bigcup_{i=1}^{n-1}\{\mathsf{Int}_{T_i}\} \cup \bar{\mathsf{Y}}\right), \\
\bar{\mathsf{V}} &= \left(\bigcup_{i=1}^{n-1}\{\mathsf{Res}_{T_i}\} \cup (\bar{\mathsf{V}} \cup \bigcup_{i=1}^{n-1}\{\mathsf{Ala}_{T_i}\})\right) \setminus \\
&\quad \left(\bigcup_{i=1}^{n-1}\{\mathsf{Ala}_{T_i}\} \cup (\bar{\mathsf{Z}} \cup \bigcup_{i=1}^{n-1}\{\mathsf{Res}_{T_i}\})\right), \\
\bar{\mathsf{Y}} &= \bar{\mathsf{Y}}, \text{ and} \\
\bar{\mathsf{Z}} &= \bar{\mathsf{Z}},
\end{aligned}$$

which obviously holds, since $\bar{\mathsf{U}} \cap \bar{\mathsf{Y}} = \emptyset$ and $\bar{\mathsf{V}} \cap \bar{\mathsf{Z}} = \emptyset$.

Also condition 2 of Definition 4.10 needs a slight modification: In part (b), we restrict the output signals to the non-timer components, resulting in the equations

$$s_{\bar{\mathsf{Y}}_i}|_{\bar{\mathsf{Y}}_i \cap \bar{\mathsf{Y}}} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i \cap \bar{\mathsf{Y}}} \quad \text{and} \quad s_{\bar{\mathsf{Z}}_i}|_{\bar{\mathsf{Z}}_i \cap \bar{\mathsf{Z}}} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i \cap \bar{\mathsf{Z}}}.$$

For proving this modified condition 2, let $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$. We have

$$(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S_{\mathcal{T}}$$

if and only if (by Definition 5.6)

$$\begin{aligned}
&\exists(\emptyset, s_v^T, s_y^T, s_z^T) \in S_T \text{ (for every } T \in \bar{T}\text{)} : \\
&(s_{\bar{\mathsf{U}}} \cup \textstyle\bigcup_{T \in \bar{T}} s_y^T, s_{\bar{\mathsf{V}}} \cup \bigcup_{T \in \bar{T}} s_z^T, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}} \cup \bigcup_{T \in \bar{T}} s_v^T) \in S_{\mathcal{D}}
\end{aligned}$$

if and only if (we introduce existential quantification for the signals of $S_{\mathcal{D}} = S_n$ and use an index $i \in \{1, \ldots, n-1\}$ for the timers' signals)

$$\begin{aligned}
&\exists(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_i \text{ (for every } i \in \{1, \ldots, n\}) : \\
&s_{\bar{\mathsf{U}}_n} = s_{\bar{\mathsf{U}}} \cup \textstyle\bigcup_{i=1}^{n-1} s_{\bar{\mathsf{Y}}_i},\ s_{\bar{\mathsf{Y}}_n} = s_{\bar{\mathsf{Y}}}, \\
&s_{\bar{\mathsf{V}}_n} = s_{\bar{\mathsf{V}}} \cup \textstyle\bigcup_{i=1}^{n-1} s_{\bar{\mathsf{Z}}_i},\ s_{\bar{\mathsf{Z}}_n} = s_{\bar{\mathsf{Z}}} \cup \textstyle\bigcup_{i=1}^{n-1} s_{\bar{\mathsf{V}}_i}
\end{aligned}$$

if and only if (we split the signals of S_n into timer and non-timer components)

$$\begin{aligned}
&\exists(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_i \text{ (for every } i \in \{1, \ldots, n\}) : \\
&\text{(a) } s_{\bar{\mathsf{U}}_n}|_{\bar{\mathsf{U}}_n \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_n \cap \bar{\mathsf{Y}}_j} \text{ for } j \in \{1, \ldots, n-1\}, \\
&\quad\ s_{\bar{\mathsf{V}}_n}|_{\bar{\mathsf{V}}_n \cap \bar{\mathsf{Z}}_j} = s_{\bar{\mathsf{Z}}_j}|_{\bar{\mathsf{V}}_n \cap \bar{\mathsf{Z}}_j} \text{ for } j \in \{1, \ldots, n-1\}, \\
&\quad\ s_{\bar{\mathsf{Z}}_n}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_n} = s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_n} \text{ for } i \in \{1, \ldots, n-1\}. \\
&\text{(b) } s_{\bar{\mathsf{U}}_n}|_{\bar{\mathsf{U}}_n \cap \bar{\mathsf{U}}} = s_{\bar{\mathsf{U}}}|_{\bar{\mathsf{U}}_n \cap \bar{\mathsf{U}}},\ s_{\bar{\mathsf{Y}}_n}|_{\bar{\mathsf{Y}}_n \cap \bar{\mathsf{Y}}} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_n \cap \bar{\mathsf{Y}}}, \\
&\quad\ s_{\bar{\mathsf{V}}_n}|_{\bar{\mathsf{V}}_n \cap \bar{\mathsf{V}}} = s_{\bar{\mathsf{V}}}|_{\bar{\mathsf{V}}_n \cap \bar{\mathsf{V}}},\ s_{\bar{\mathsf{Z}}_n}|_{\bar{\mathsf{Z}}_n \cap \bar{\mathsf{Z}}} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_n \cap \bar{\mathsf{Z}}}.
\end{aligned}$$

if and only if (all equations we add in this step have signals over the empty name set on both sides are therefore trivially valid)

$$\begin{aligned}
&\exists(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_i \text{ (for every } i \in \{1, \ldots, n\}) : \\
&\text{(a) For all } i, j \in \{1, \ldots, n\}\text{:} \\
&\quad\ s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j}, \\
&\quad\ s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j} = s_{\bar{\mathsf{Z}}_j}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j}. \\
&\text{(b) For all } i \in \{1, \ldots, n\}\text{:} \\
&\quad\ s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} = s_{\bar{\mathsf{U}}}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}},\ s_{\bar{\mathsf{Y}}_i}|_{\bar{\mathsf{Y}}_i \cap \bar{\mathsf{Y}}} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i \cap \bar{\mathsf{Y}}}, \\
&\quad\ s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} = s_{\bar{\mathsf{V}}}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}},\ s_{\bar{\mathsf{Z}}_i}|_{\bar{\mathsf{Z}}_i \cap \bar{\mathsf{Z}}} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i \cap \bar{\mathsf{Z}}}.
\end{aligned}$$ ■

We give a small example for a timed condition/event system.

Example 5.9 Recall Example 2.8 on page 23. We presented a CES $\mathcal{S}$ that generates a "ping" event at any time $t \in \mathbb{N} \setminus \{0\}$. Now we present a TCES $\mathcal{T}$ with $\mathcal{S}_{\mathcal{T}} = \mathcal{S}$ (since we did not have names in Chapter 2, a simple conversion is necessary). One timer with a single threshold 1 is needed in $\mathcal{T}$. Every time the Ala_1 event is received from the timer, the DCES part $\mathcal{D}$ of $\mathcal{T}$ generates the "ping" as an output event and resets the timer by sending the Res event.

Let $T \in \mathbb{T}$ with $\Theta(T) = (1)$. Define $\mathcal{T} = (\{\mathsf{U}\}, \{\mathsf{V}\}, \{\mathsf{Y}\}, \{\mathsf{Z}\}, \{T\}, \mathcal{D})$ with $\alpha(\mathsf{U}) = \alpha(\mathsf{Y}) = \{0\}$, $\alpha(\mathsf{V}) = \{\mathbf{0}\}$, $\alpha(\mathsf{Z}) = \{\mathbf{0}, \text{ping}\}$, and $\mathcal{D} = (\bar{\mathsf{U}}^+, \bar{\mathsf{V}}^+, \{x\}, \{\mathsf{Y}\}, \bar{\mathsf{Z}}^+, f, g, h, \{x\})$, where (recall that g and h are functions mapping condition/event names to condition/event symbols)

- $\bar{\mathsf{U}}^+ = \{\mathsf{U}, \mathsf{Int}_T\}$, $\bar{\mathsf{V}}^+ = \{\mathsf{V}, \mathsf{Ala}_T\}$, $\bar{\mathsf{Z}}^+ = \{\mathsf{Z}, \mathsf{Res}_T\}$,
- $f(x, u, v) = \{x\}$ for all $u \in \Sigma_{\bar{\mathsf{U}}^+}$, $v \in \Sigma_{\bar{\mathsf{V}}^+}$,
- $g(x, u) = \{\mathsf{Y} \mapsto 0\}$ for all $u \in \Sigma_{\bar{\mathsf{U}}^+}$, and
- $h(x, x, v) = \begin{cases} \{\mathsf{Z} \mapsto \text{ping}, \mathsf{Res}_T \mapsto \text{Res}\}, \text{ if } v(\mathsf{Ala}_T) = \text{Ala}_1 \\ \{\mathsf{Z} \mapsto \mathbf{0} \quad, \mathsf{Res}_T \mapsto \mathbf{0} \quad \}, \text{ otherwise} \end{cases}.$

With these definitions, we have the expected behavior $\mathcal{S}_{\mathcal{T}} = \mathcal{S}$. $\diamond$

5.3 The Parallel Interconnection of Timed C/E Systems

Before we can reason about a parallel operator for timed condition/event systems, we define what consistency means for such systems. In addition to consistency for CESs and DCESs, we have to ensure that each discrete system uses its own set of timers; no sharing of timers is allowed.

Definition 5.10 (Consistent set of TCESs) *For $i \in \{1, \ldots, n\}$, let $\mathcal{T}_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, \bar{T}_i, \mathcal{D}_i)$ be a TCES. The set $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ is called* consistent, *if*

$$\left(\bar{\mathsf{Y}}_i \cup \bar{\mathsf{Z}}_i\right) \cap \left(\bar{\mathsf{Y}}_j \cup \bar{\mathsf{Z}}_j\right) = \emptyset \quad \textit{and} \quad \bar{T}_i \cap \bar{T}_j = \emptyset$$

for all $i, j \in \{1, \ldots, n\}$ with $i \neq j$. □

The parallel interconnection of TCESs can be constructed using the techniques presented in Chapter 4. We simply apply the discrete parallel interconnection to the DCESs inside the TCESs, getting a new DCES which serves as the discrete part of the TCES representing the composed system.

Theorem 5.11 *Let $\mathcal{T}_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, \bar{T}_i, \mathcal{D}_i)$ be a TCES with $\mathcal{D}_i = (\bar{\mathsf{U}}_i^+, \bar{\mathsf{V}}_i^+, X_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i^+, f_i, g_i, h_i, X_{0,i})$, for $i \in \{1, \ldots, n\}$. Let $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ be consistent. A TCES $\mathcal{T}$ can be effectively constructed such that $\mathcal{S}_{\mathcal{T}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{T}_1}, \ldots, \mathcal{S}_{\mathcal{T}_n}$, if the multiple solution property of Theorem 4.27 holds for $\mathcal{D}_1, \ldots, \mathcal{D}_n$.* □

The discrete part that has to be constructed for $\mathcal{T}$ is just the discrete parallel interconnection (with multiple solutions) of $\mathcal{D}_1, \ldots, \mathcal{D}_n$.

Proof Let $\mathcal{T}_i$ and $\mathcal{D}_i$, $i \in \{1, \ldots, n\}$, be given as in Theorem 5.11. Let the multiple solution property hold. Since $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ is consistent, $\{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$ is also consistent, and by Theorem 4.27 we can define

$$\mathcal{D} = (\bar{\mathsf{U}}^+, \bar{\mathsf{V}}^+, X, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}^+, f, g, h, X_0) = \mathcal{D}_1 \,|||\, \ldots \,|||\, \mathcal{D}_n.$$

Since $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ is consistent, we have no interference of the timer names, leading to

- $\bar{\mathsf{U}}^+ = \left(\left(\bigcup_{i=1}^n \bar{\mathsf{U}}_i\right) \setminus \left(\bigcup_{i=1}^n \bar{\mathsf{Y}}_i\right)\right) \cup \bigcup_{i=1}^n \bigcup_{T \in \bar{T}_i} \{\mathsf{Int}_T\}$,
- $\bar{\mathsf{V}}^+ = \left(\left(\bigcup_{i=1}^n \bar{\mathsf{V}}_i\right) \setminus \left(\bigcup_{i=1}^n \bar{\mathsf{Z}}_i\right)\right) \cup \bigcup_{i=1}^n \bigcup_{T \in \bar{T}_i} \{\mathsf{Ala}_T\}$,
- $\bar{\mathsf{Y}} = \bigcup_{i=1}^n \bar{\mathsf{Y}}_i$, and
- $\bar{\mathsf{Z}}^+ = \left(\bigcup_{i=1}^n \bar{\mathsf{Z}}_i\right) \cup \bigcup_{i=1}^n \bigcup_{T \in \bar{T}_i} \{\mathsf{Res}_T\}$.

Now we define the TCES $\mathcal{T} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, \bar{T}, \mathcal{D})$ with

- $\bar{\mathsf{U}} = \left(\bigcup_{i=1}^n \bar{\mathsf{U}}_i\right) \setminus \left(\bigcup_{i=1}^n \bar{\mathsf{Y}}_i\right)$,
- $\bar{\mathsf{V}} = \left(\bigcup_{i=1}^n \bar{\mathsf{V}}_i\right) \setminus \left(\bigcup_{i=1}^n \bar{\mathsf{Z}}_i\right)$,
- $\bar{\mathsf{Z}} = \bigcup_{i=1}^n \bar{\mathsf{Z}}_i$, and
- $\bar{T} = \bigcup_{i=1}^n \bar{T}_i$.

Now we show that $\mathcal{S}_{\mathcal{T}}$ is a parallel interconnection of $\mathcal{S}_{\mathcal{T}_1}, \ldots, \mathcal{S}_{\mathcal{T}_n}$. Condition 1 of Definition 4.10 is obviously fulfilled. For proving condition 2, let $(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in \mathrm{Beh}(\bar{\mathsf{U}}, \bar{\mathsf{V}}, \bar{\mathsf{Y}}, \bar{\mathsf{Z}})$. We have

$$(s_{\bar{\mathsf{U}}}, s_{\bar{\mathsf{V}}}, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}}) \in S_{\mathcal{T}}$$

if and only if (by Definition 5.6)

$$\exists(\emptyset, s_v^T\, s_y^T, s_z^T) \in S_T \text{ (for all } T \in \bar{T}) :$$
$$(s_{\bar{\mathsf{U}}} \cup \textstyle\bigcup_{T \in \bar{T}} s_y^T, s_{\bar{\mathsf{V}}} \cup \bigcup_{T \in \bar{T}} s_z^T, s_{\bar{\mathsf{Y}}}, s_{\bar{\mathsf{Z}}} \cup \bigcup_{T \in \bar{T}} s_v^T) \in S_{\mathcal{D}}$$

if and only if (since $S_{\mathcal{D}}$ is a parallel interconnection of $S_{\mathcal{D}_1}, \ldots, S_{\mathcal{D}_n}$, we can use Definition 4.10)

$$\exists(\emptyset, s_v^T\, s_y^T, s_z^T) \in S_T \text{ (for all } T \in \bar{T}) :$$
$$\exists(s_{\bar{\mathsf{U}}_i^+}, s_{\bar{\mathsf{V}}_i^+}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i^+}) \in S_{\mathcal{D}_i} \text{ (for all } i \in \{1, \ldots, n\}) :$$

(a) For all $i, j \in \{1, \ldots, n\}$:

$$s_{\bar{\mathsf{U}}_i^+}|_{\bar{\mathsf{U}}_i^+ \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_i^+ \cap \bar{\mathsf{Y}}_j},\ s_{\bar{\mathsf{V}}_i^+}|_{\bar{\mathsf{V}}_i^+ \cap \bar{\mathsf{Z}}_j^+} = s_{\bar{\mathsf{Z}}_j^+}|_{\bar{\mathsf{V}}_i^+ \cap \bar{\mathsf{Z}}_j^+}.$$

(b) For all $i \in \{1, \ldots, n\}$:

$$s_{\bar{\mathsf{U}}_i^+}|_{\bar{\mathsf{U}}_i^+ \cap \bar{\mathsf{U}}^+} = (s_{\bar{\mathsf{U}}} \cup \textstyle\bigcup_{T \in \bar{T}} s_y^T)|_{\bar{\mathsf{U}}_i^+ \cap \bar{\mathsf{U}}^+},\ s_{\bar{\mathsf{Y}}_i} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i},$$
$$s_{\bar{\mathsf{V}}_i^+}|_{\bar{\mathsf{V}}_i^+ \cap \bar{\mathsf{V}}^+} = (s_{\bar{\mathsf{V}}} \cup \textstyle\bigcup_{T \in \bar{T}} s_z^T)|_{\bar{\mathsf{V}}_i^+ \cap \bar{\mathsf{V}}^+},\ s_{\bar{\mathsf{Z}}_i^+} = (s_{\bar{\mathsf{Z}}} \cup \bigcup_{T \in \bar{T}} s_v^T)|_{\bar{\mathsf{Z}}_i}.$$

if and only if (we split the signals of the $S_{\mathcal{D}_i}$ into their non-timer and timer parts)

$$\begin{aligned}
&\exists(\emptyset, s_v^T\, s_y^T, s_z^T) \in S_T \text{ (for all } T \in \bar{T}) :\\
&\exists s_{\bar{\mathsf{U}}_i} \in \Sigma_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i} \in \Sigma_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i} \in \Sigma_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i} \in \Sigma_{\bar{\mathsf{Z}}_i},\\
&\exists s_{u_i}^T \in \Sigma_{\bar{\mathsf{U}}_i^+ \setminus \bar{\mathsf{U}}_i}, s_{v_i}^T \in \Sigma_{\bar{\mathsf{V}}_i^+ \setminus \bar{\mathsf{V}}_i}, s_{z_i}^T \in \Sigma_{\bar{\mathsf{Z}}_i^+ \setminus \bar{\mathsf{Z}}_i}\\
&\text{with } (s_{\bar{\mathsf{U}}_i} \cup s_{u_i}^T, s_{\bar{\mathsf{V}}_i} \cup s_{v_i}^T, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i} \cup s_{z_i}^T) \in S_{\mathcal{D}_i} \text{ (for all } i \in \{1, \ldots, n\}) :\\
&\text{(a) For all } i, j \in \{1, \ldots, n\}\text{:}\\
&\quad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j},\ s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j} = s_{\bar{\mathsf{Z}}_j}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j}.\\
&\text{(b) For all } i \in \{1, \ldots, n\}\text{:}\\
&\quad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} = s_{\bar{\mathsf{U}}}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}},\ s_{u_i}^T = \textstyle\bigcup_{T \in \bar{T}_i} s_y^T,\\
&\quad s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} = s_{\bar{\mathsf{V}}}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}},\ s_{v_i}^T = \textstyle\bigcup_{T \in \bar{T}_i} s_z^T,\\
&\quad s_{\bar{\mathsf{Y}}_i} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i},\\
&\quad s_{\bar{\mathsf{Z}}_i}|_{\bar{\mathsf{Z}}_i \cap \bar{\mathsf{Z}}} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i \cap \bar{\mathsf{Z}}},\ s_{z_i}^T = \textstyle\bigcup_{T \in \bar{T}_i} s_v^T.
\end{aligned}$$

if and only if (some existentially quantified timer signals are completely fixed by others; we replace $s_{u_i}^T$ by $\bigcup_{T \in \bar{T}_i} s_y^T$, $s_{v_i}^T$ by $\bigcup_{T \in \bar{T}_i} s_z^T$, and $s_{z_i}^T$ by $\bigcup_{T \in \bar{T}_i} s_v^T$, for all $i \in \{1, \ldots, n\}$)

$$\begin{aligned}
&\exists(\emptyset, s_v^T, s_y^T, s_z^T) \in S_T \text{ (for all } T \in \bar{T}_i) :\\
&\exists s_{\bar{\mathsf{U}}_i} \in \Sigma_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i} \in \Sigma_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i} \in \Sigma_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i} \in \Sigma_{\bar{\mathsf{Z}}_i}\\
&\text{with } (s_{\bar{\mathsf{U}}_i} \cup \textstyle\bigcup_{T \in \bar{T}_i} s_y^T, s_{\bar{\mathsf{V}}_i} \cup \bigcup_{T \in \bar{T}_i} s_z^T, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i} \cup \bigcup_{T \in \bar{T}_i} s_v^T) \in S_{\mathcal{D}_i}\\
&\text{(for all } i \in \{1, \ldots, n\}) :\\
&\text{(a) For all } i, j \in \{1, \ldots, n\}\text{:}\\
&\quad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j},\ s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j} = s_{\bar{\mathsf{Z}}_j}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j}.\\
&\text{(b) For all } i \in \{1, \ldots, n\}\text{:}\\
&\quad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} = s_{\bar{\mathsf{U}}}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}},\ s_{\bar{\mathsf{Y}}_i} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i},\\
&\quad s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} = s_{\bar{\mathsf{V}}}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}},\ s_{\bar{\mathsf{Z}}_i} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i}.
\end{aligned}$$

if and only if (by Definition 5.6)

$$\begin{aligned}
&\exists(s_{\bar{\mathsf{U}}_i}, s_{\bar{\mathsf{V}}_i}, s_{\bar{\mathsf{Y}}_i}, s_{\bar{\mathsf{Z}}_i}) \in S_{\mathcal{T}_i} \text{ (for all } i \in \{1, \ldots, n\}) :\\
&\text{(a) For all } i, j \in \{1, \ldots, n\}\text{:}\\
&\quad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j} = s_{\bar{\mathsf{Y}}_j}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{Y}}_j},\ s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j} = s_{\bar{\mathsf{Z}}_j}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{Z}}_j}.\\
&\text{(b) For all } i \in \{1, \ldots, n\}\text{:}\\
&\quad s_{\bar{\mathsf{U}}_i}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}} = s_{\bar{\mathsf{U}}}|_{\bar{\mathsf{U}}_i \cap \bar{\mathsf{U}}},\ s_{\bar{\mathsf{Y}}_i} = s_{\bar{\mathsf{Y}}}|_{\bar{\mathsf{Y}}_i},\\
&\quad s_{\bar{\mathsf{V}}_i}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}} = s_{\bar{\mathsf{V}}}|_{\bar{\mathsf{V}}_i \cap \bar{\mathsf{V}}},\ s_{\bar{\mathsf{Z}}_i} = s_{\bar{\mathsf{Z}}}|_{\bar{\mathsf{Z}}_i}.
\end{aligned}$$

■

Definition 5.12 (Discrete parallel interconnection of TCESs) *Let the set $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ of timed condition/event systems be consistent, and let the multiple solution property of Theorem 4.27 hold. The TCES $\mathcal{T}$ gained from the construction above is called the* discrete parallel interconnection *of $\mathcal{T}_1, \ldots, \mathcal{T}_n$, denoted by $\mathcal{T}_1 \,|||\, \ldots \,|||\, \mathcal{T}_n$.* □

Remark 5.13 Theorem 5.11 and Definition 5.12 can also be formulated for *unique* solutions, based on Theorem 4.13 and Definition 4.15. Construction and proof are analogous. We denote the *discrete parallel interconnection with unique solutions* of some consistent TCESs $\mathcal{T}_1, \ldots, \mathcal{T}_n$ by $\mathcal{T}_1 \| \ldots \| \mathcal{T}_n$. ◇

Furthermore, since every DCES is expressible as a TCES (see Lemma 5.7), we also allow DCESs as parameters for the two parallel interconnection operators given above.

Chapter 6

Example: The Evaporator System

The example we are about to give in this chapter is a part of an experimental batch plant used in the Process Control Laboratory at the Chemical Engineering Department, University of Dortmund. This plant is used to illustrate a chemical batch process, its operation, and its automated control. A description of the full plant can be found in [Kow95].

Figure 6.1 shows the part of the plant we are going to examine. It mainly consists of four physical devices: Two tanks in which liquids and steam can be kept, a condenser which is able to cool steam down to water, and a heating device for increasing the temperature of the liquid inside the first tank. There are one-way pipes between tank 1 and tank 2, tank 1 and the condenser, and there are pipes to fill tank 1, to empty tank 2, and to remove the water from the condenser. Some pipes can be opened or closed using valves.

The normal operation of this system part is a follows: Tank 1 is filled through valve 1 with a salt solution of low concentration. Then the heater is switched on, and the concentration of the solution rises, while steam evaporates from tank 1 into the condenser. If the condenser works properly, the steam cools down to water and leaves the condenser. When a certain concentration is reached, the heater is switched off, and tank 1 is emptied via valve 2 into tank 2. A postprocessing step takes place in tank 2, then the solution leaves this part of the system via valve 3.

6.1 The Problem

In our example, we are not interested in the normal operation, but on the system's reaction to certain malfunctions which may endanger the safety of

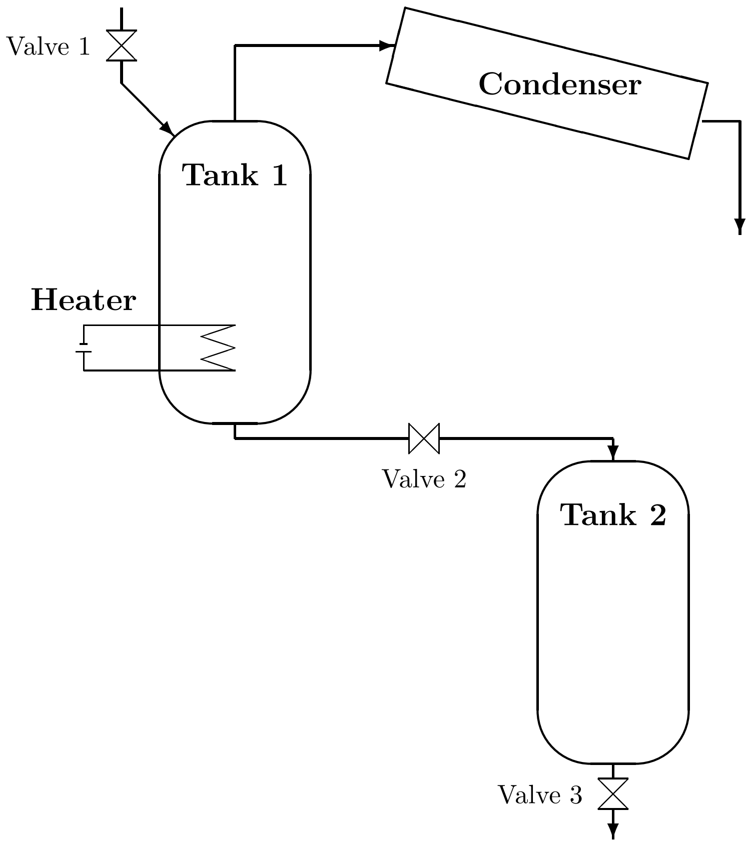


Figure 6.1: Diagram of the evaporator system.

the plant. Consider the following scenario: During the evaporation of the solution in tank 1, the condenser malfunctions, for example due to missing cooling agent. Hence, the steam cannot be cooled, and the pressure inside the condenser starts to rise. Now the heater in tank 1 has to be switched off to prevent further production of steam, since exceeding a certain pressure limit may cause the condenser to explode.

After having switched off the heater in the case of a condenser malfunction, another problem arises: If the temperature in tank 1 cools down below a certain temperature, the solution might get solid and cannot be drained into tank 2. Therefore, valve 2 must be opened early enough to let the solution flow into tank 2, where the postprocessing step prevents the solution from getting solid. However, before valve 2 can be opened, we must ensure that tank 2 is empty; otherwise an overflow occurs in tank 2. So we might have to open valve 3 first to drain tank 2.

Let us summarize the characteristics of a sensible strategy to react on a cooling malfunction in the condenser: The heater has to be switched off *early enough* to prevent an explosion of the condenser, but *not too early,* because the solution may become solid while we wait for tank 2 being emptied. To keep this time slot as large as possible, it seems to be reasonable to open valve 3 immediately if we detect a cooling malfunction and tank 2 is not empty.

Our task is now to model a controller which can cope with the problems discussed above. The controller, which can influence the system by switching off the heater and using valves 2 and 3 to drain both tanks, must fulfill the following safety properties:

1. If a cooling malfunction in the condenser occurs, the heater must be switched off fast enough such that the condenser does not explode.

2. If the heater is switched off, tank 1 is drained into tank 2 before the solution gets solid.

3. If valve 2 is opened to drain tank 1 into tank 2, tank 2 has to be empty.

We will give formalized versions of these properties after we have defined a formal model of the system with its controller.

The following timing parameters are given by the physical characteristics of the batch plant:

- If the condenser malfunctions and the production of steam in tank 1 continues, the condenser may explode after more than 8 time units.

- If the heater in tank 1 is switched off, it takes exactly 5 time units until the production of steam stops.

- If the production of steam stops in tank 1, the solution may get solid after more than 4 time units.

- It takes between 0 and 10 time units to empty tank 2.

The draining of tank 1 is supposed to be fast enough, so no timing constraint is given here.

6.2 The Model

The system consisting of the four physical devices and the controller is modeled by a parallel interconnection of timed condition/event systems, or, where no quantitative timing is needed, discrete condition/event systems. A block

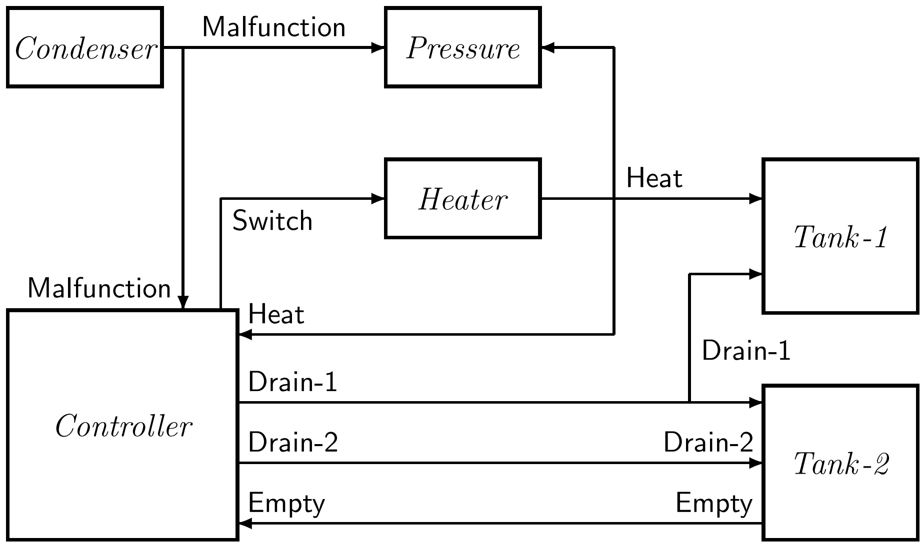


Figure 6.2: Block diagram of the evaporator system model.

diagram of the complete system is given in Figure 6.2. All signals shown are event signals.

The evaporator system consists of six modules. The modules and their functions are:

- *Tank-1* models tank 1, including its contents and valve 2.
- *Tank-2* models tank 2, including its contents and valve 3.
- *Heater* models the heater in tank 1.
- *Condenser* models the condenser.
- *Pressure* models the pressure in the condenser.
- *Controller* models the controller of the plant.

There exist several other possibilities for splitting up the system into modules; for example, one can model *Condenser* and *Pressure* as one module. The split chosen here is a tradeoff between a small number of modules and a low complexity of each module.

Before we define a TCES for each module, we take a look at the events used in the system. For each event name N in the system, we give the proper event in $\alpha(\mathsf{N})$, and we explain its meaning. Each of the alphabets has just one proper event.

- Malfunction: The event *mal* is generated by *Condenser* to indicate that it starts malfunctioning, i.e., the cooling of the steam no longer works.
- Switch: The event *off* is sent from *Controller* to *Heater* to indicate that the heating device should switch off.
- Heat: The event *off* from *Heater* indicates that the heating device has switched off and no steam is produced in tank 1 any longer.
- Drain-1: The event *drain* is sent from *Controller* to *Tank-1* to open valve 2 and start draining the contents of tank 1 into tank 2.
- Drain-2: The event *drain* is sent from *Controller* to *Tank-2* to open valve 3 and start draining tank 2.
- Empty: The event *empty* from *Tank-2* indicates that the draining of tank 2 is completed, tank 2 is empty, and valve 3 has been closed.

Since we are only interested in the malfunction scenario, we do *not* model the evaporator system's normal behavior, but only the parts that are involved in bringing the system to a safe state if a condenser malfunction occurs. Thus, we assume that all valves are closed, tank 1 has already been filled, the heater is switched on, and the condenser works properly. Furthermore, since we are interested in a worst case scenario, tank 2 is also filled initially.

6.2.1 The Modules

The modules of the evaporator system are specified as TCESs. In the module *Condenser*, no quantitative timing is needed, and thus, it is specified as a DCES. In addition to the connections shown in Figure 6.2, each module has one condition output component. The output symbol of this component is an abstraction of the current control state of the module. We will later use these symbols to verify the three properties mentioned before.

For every module, we give a block diagram showing its input and output components. To keep the diagrams simple, the timers and their connections are not shown. Furthermore, its discrete functions f, g, and h are given, using a graphical representation which is explained in detail in the definition of *Tank-1*.

6.2.1.1 The TCES *Tank-1*

Figure 6.3 shows the block diagram of the TCES *Tank-1*. *Tank-1* works as follows: Initially, tank 1 is full. If the *drain* event is received via component

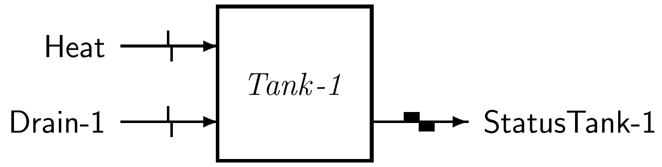


Figure 6.3: Block diagram of the TCES *Tank-1*.

Drain-1 from *Controller*, tank 1 gets empty. If the *off* event is received via component Heat from *Heater*, the solution in tank 1 is in danger of getting solid. A timer with a single threshold of 4 is started, and after 4 time units, the solution may get solid. If the *drain* event occurs within this period, tank 1 gets empty, and this part of the system is safe. If the *drain* event is sent after more than 4 time units, the solution might have become solid, or it might not. In this case it is not predictable if the system will get safe.

We define the TCES

$$\mathit{Tank\text{-}1} = (\emptyset, \{\mathsf{Heat}, \mathsf{Drain\text{-}1}\}, \{\mathsf{StatusTank\text{-}1}\}, \emptyset, \{T_{T1}\}, \mathcal{D}_{\mathit{Tank\text{-}1}})$$

with $\alpha(\mathsf{Heat}) = \{\mathbf{0}, \mathit{off}\}$, $\alpha(\mathsf{Drain\text{-}1}) = \{\mathbf{0}, \mathit{drain}\}$, $\alpha(\mathsf{StatusTank\text{-}1}) = \{\text{full}, \text{danger}, \text{empty}, \text{solid}\}$, $\Theta(T_{T1}) = (4)$, and

$$\begin{aligned}\mathcal{D}_{\mathit{Tank\text{-}1}} = (&\{\mathsf{Int}_{T_{T1}}\}, \{\mathsf{Heat}, \mathsf{Drain\text{-}1}, \mathsf{Ala}_{T_{T1}}\}, \{x_0, x_1, x_2, x_3\},\\ &\{\mathsf{StatusTank\text{-}1}\}, \{\mathsf{Res}_{T_{T1}}\}, f_1, g_1, h_1, \{x_0\}).\end{aligned}$$

The functions f_1, g_1 and h_1 are given by a graphical representation in Figure 6.4. The circles represent the control states; at the top of each circle the identifier x_i for that state is given. Since all condition output functions we define in this chapter are independent from the condition input symbols (there are only those sent from the timers), we just put the condition output symbol into each state. Thus, Figure 6.4 defines $g_1(x_0, u)(\mathsf{StatusTank\text{-}1}) =$ full, $g_1(x_1, u)(\mathsf{StatusTank\text{-}1}) =$ danger, $g_1(x_2, u)(\mathsf{StatusTank\text{-}1}) =$ solid, and $g_1(x_3, u)(\mathsf{StatusTank\text{-}1}) =$ empty, for all $u \in \Sigma_{\mathsf{Int}_{T_{T1}}}$. We will use these output condition symbols for verification purposes only.

The edges in Figure 6.4 define f_1 and h_1. An edge from a state x_i to a state x_j denotes that there are u, v such that $x_j \in f_1(x_i, u, v)$. The possible values for u and v are described by the labeling of that edge. For example, the edge from x_0 to x_1 defines the following state transitions:

$$\begin{aligned}&\text{For all } u \in \Sigma_{\mathsf{Int}_{T_{T1}}}, v \in \Sigma_{\{\mathsf{Heat}, \mathsf{Drain\text{-}1}, \mathsf{Ala}_{T_{T1}}\}}:\\ &x_1 \in f_1(x_0, u, v) \quad \text{if} \quad v(\mathsf{Heat}) = \mathit{off} \wedge v(\mathsf{Drain\text{-}1}) = \mathbf{0}.\end{aligned}$$

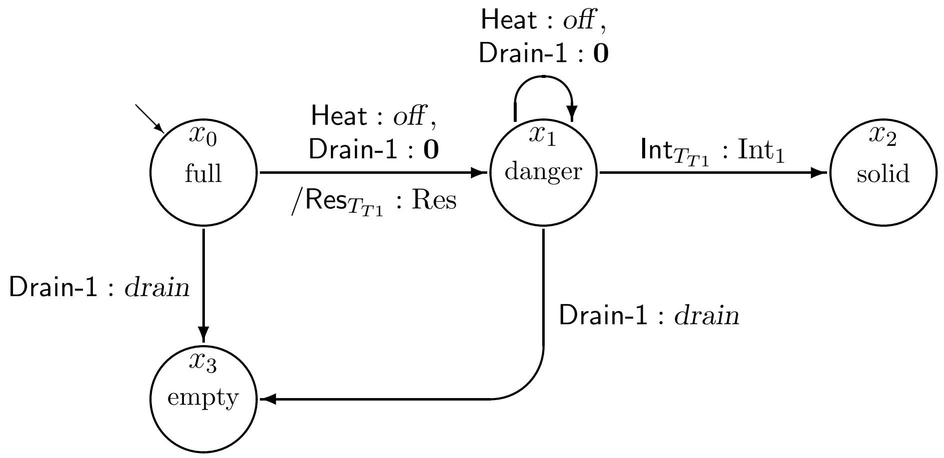


Figure 6.4: Discrete part of the TCES *Tank-1*.

Everything after the slash (/) is used to indicate the event output evaluation generated by h_1 during that transition. For example, the edge from x_0 to x_1 defines the following event output:

$$\text{For all } v \in \Sigma_{\{\mathsf{Heat},\mathsf{Drain\text{-}1},\mathsf{Ala}_{T_{T1}}\}}:$$
$$h_1(x_0, x_1, v)(\mathsf{Res}_{T_{T1}}) = \text{Res} \quad \text{if} \quad v(\mathsf{Heat}) = \mathit{off} \wedge v(\mathsf{Drain\text{-}1}) = \mathbf{0}.$$

Note that the parts of u and v not fixed in the labeling may have any value.

If for some x, u, and v there is no edge leaving state x having a labeling describing u and v, we assume $f_1(x, u, v) = \{x\}$, i.e., there is no state change in state x for the input evaluations u and v. Furthermore, we assume $x \in f_1(x, u, \mathbf{0})$ for any x and u to obtain stuttering. And if there is no edge from state x to x' describing the event output evaluation of some name N for some event input evaluation v, we assume $h_1(x, x', v)(\mathsf{N}) = \mathbf{0}$, i.e., there is no proper event generated. With these default values, stuttering and output triggering can be checked easily.

Take again a look at Figure 6.4. The edge from x_1 to x_2 allows a transition leaving x_1 at any time after timer T_{T1} has reached its first (and only) threshold, which is 4. If the event *drain* arrives after this threshold has been reached, there two transitions possible: one to the "good" state x_3, and one to the "bad" state x_2. This nondeterminism models that after 4 time units, a draining command from the controller may arrive early enough, or it may not. To be on the safe side in any case, the last possibility to send *drain* must be exactly 4 time units after *off* was sent by *Heater*.

The self-loop edge from x_1 to x_1 is only used to prevent the forcing of the transition from x_1 to x_2 in case further Heat : *off* events occur at state

x_1. Since the labeling on the edge from x_1 to x_2 does not contain the name Heat, there exists a transition from x_1 to x_2 which is triggered by Heat : *off*, and therefore the defaults for f_1 described above will not implicitly assume the self-loop we defined explicitly.

6.2.1.2 The TCES *Tank-2*

Figure 6.5 shows the block diagram of the TCES *Tank-2*. *Tank-2* works as

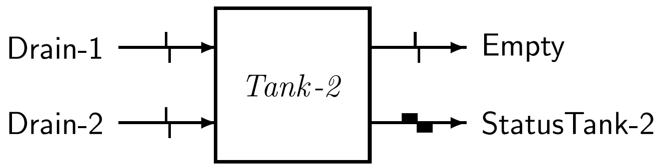


Figure 6.5: Block diagram of the TCES *Tank-2*.

follows: Initially, tank 2 is full. If the *drain* event is received via component Drain-2 from *Controller*, tank 2 starts to drain. A clock is used to ensure that within 10 time units, the *empty* symbol is sent via component Empty to *Controller*, denoting that tank 2 is empty. If the *drain* event occurs while tank 2 is still full or in the process of draining, tank 2 overflows.

We define the TCES

$$\begin{aligned}&\mathit{Tank\text{-}2}\\ &= (\emptyset, \{\mathsf{Drain\text{-}1}, \mathsf{Drain\text{-}2}\}, \{\mathsf{StatusTank\text{-}2}\}, \{\mathsf{Empty}\}, \{T_{T2}\}, \mathcal{D}_{\mathit{Tank\text{-}2}})\end{aligned}$$

with $\alpha(\mathsf{Drain\text{-}1}) = \{\mathbf{0}, \mathit{drain}\}$, $\alpha(\mathsf{Drain\text{-}2}) = \{\mathbf{0}, \mathit{drain}\}$, $\alpha(\mathsf{StatusTank\text{-}2}) = \{\text{full}, \text{empty}, \text{draining}, \text{overflow}\}$, $\alpha(\mathsf{Empty}) = \{\mathbf{0}, \mathit{empty}\}$, $\Theta(T_{T2}) = (10)$, and

$$\begin{aligned}\mathcal{D}_{\mathit{Tank\text{-}2}} = (&\{\mathsf{Int}_{T_{T2}}\}, \{\mathsf{Drain\text{-}1}, \mathsf{Drain\text{-}2}, \mathsf{Ala}_{T_{T2}}\}, \{x_0, x_1, x_2, x_3\},\\ &\{\mathsf{StatusTank\text{-}2}\}, \{\mathsf{Empty}, \mathsf{Res}_{T_{T2}}\}, f_2, g_2, h_2, \{x_0\}).\end{aligned}$$

The functions f_2, g_2 and h_2 are given by Figure 6.6. Note that there are two different kinds of transitions denoted by the edge from x_2 to x_1; one state transition that *may* be taken within 10 time units, and one transition that *must* be taken when 10 time units have elapsed. The first transition is only *enabled* by the condition symbol Int_0, while the second one is *forced* by the event symbol Ala_1, because we have no transition from x_2 to itself labeled with $\mathsf{Ala}_{T_{T2}}$: Ala_1. Since the occurrence of Ala_1 is allowed for transitions from x_2 to x_1, this is the only way to go. This models the fact that draining tank 2 may take up to 10 time units, but at time 10 tank 2 must be empty.

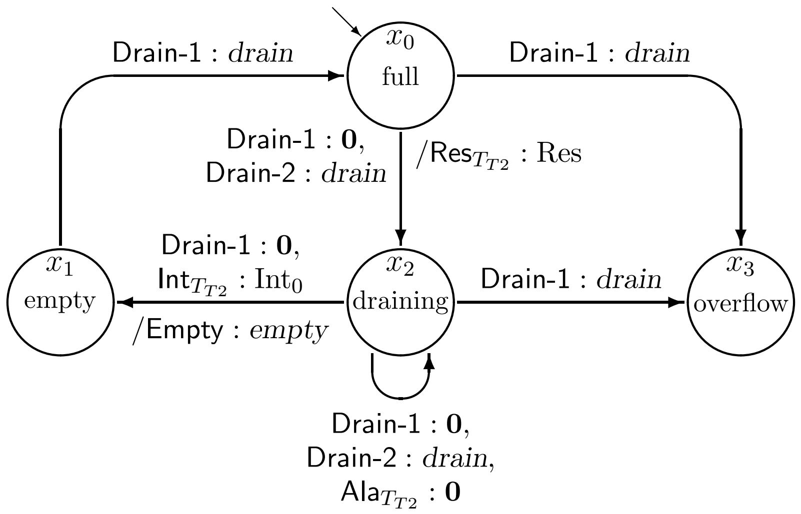


Figure 6.6: Discrete part of the TCES *Tank-2*.

Note however that draining tank 1 exactly 10 time units after draining tank 2 will lead to an overflow of tank 2. Any time later is safe in all cases.

The self-loop edge from x_2 to x_2 is only used to prevent the forcing of the transition from x_2 to x_1 in case further Drain-2 : *drain* events occur at state x_1 before 10 time units have elapsed.

6.2.1.3 The TCES *Heater*

Figure 6.7 shows the block diagram of the TCES *Heater*. *Heater* works as

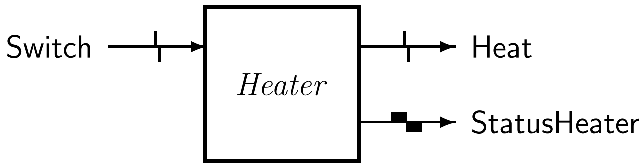


Figure 6.7: Block diagram of the TCES *Heater*.

follows: Initially, the heater is working, and steam is produced in tank 1. If the *off* event is received via component Switch from *Controller*, the heater is switched off. Since it takes 5 time units for the heater to cool down such that no steam is produced any more, a timer is used to send the *off* event after 5 time units via component Heat to denote that no steam is produced any longer.

We define the TCES

$$Heater = (\emptyset, \{\mathsf{Switch}\}, \{\mathsf{StatusHeater}\}, \{\mathsf{Heat}\}, \{T_H\}, \mathcal{D}_{Heater})$$

with $\alpha(\mathsf{Switch}) = \{\mathbf{0}, \mathit{off}\}$, $\alpha(\mathsf{Heat}) = \{\mathbf{0}, \mathit{off}\}$, $\alpha(\mathsf{StatusHeater}) = \{\text{on}, \text{cold}, \text{cooling}\}$, $\Theta(T_H) = (5)$, and

$$\begin{aligned}\mathcal{D}_{Heater} = (&\{\mathsf{Int}_{T_H}\}, \{\mathsf{Switch}, \mathsf{Ala}_{T_H}\}, \{x_0, x_1, x_2\},\\ &\{\mathsf{StatusHeater}\}, \{\mathsf{Heat}, \mathsf{Res}_{T_H}\}, f_H, g_H, h_H, \{x_0\}).\end{aligned}$$

The functions f_H, g_H and h_H are given by Figure 6.8. Note that unlike in Figure 6.6, where there is a transition that *may* be taken, the edge from x_1 to x_2 denotes that a transition *must* be taken if and only if the Ala_1 event occurs. Hence, the *off* event in component Heat is always sent *exactly* 5 time units after *off* has been received via component Switch.

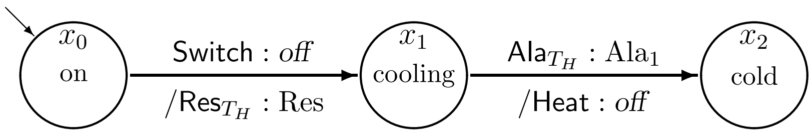


Figure 6.8: Discrete part of the TCES *Heater*.

6.2.1.4 The DCES *Condenser*

Figure 6.9 shows the block diagram of the DCES *Condenser*. Since this

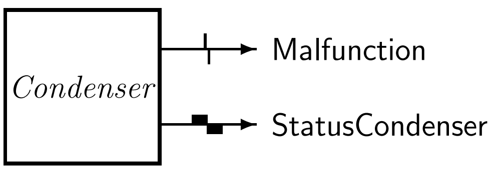


Figure 6.9: Block diagram of the DCES *Condenser*.

system needs no timers, is it modeled as a DCES. *Condenser* has two states; one showing that it is working, and one denoting that it is malfunctioning. Initially, the condenser is working. It can malfunction at any time by sending the *mal* event via component Malfunction.

We have the DCES

$$\begin{aligned}Condenser = (&\emptyset, \emptyset, \{x_0, x_1, x_2\}, \{\mathsf{StatusCondenser}\}, \{\mathsf{Malfunction}\},\\ &f_C, g_C, h_C, \{x_0\})\end{aligned}$$

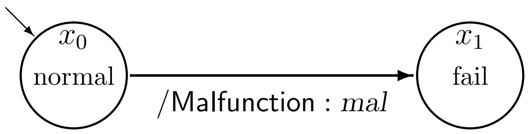


Figure 6.10: Discrete part of the DCES *Condenser*.

with $\alpha(\mathsf{Malfunction}) = \{\mathbf{0}, mal\}$ and $\alpha(\mathsf{StatusCondenser}) = \{\text{normal}, \text{fail}\}$. The functions f_C, g_C and h_C are given by Figure 6.10.

6.2.1.5 The TCES *Pressure*

Figure 6.11 shows the block diagram of the TCES *Pressure*. *Pressure* is

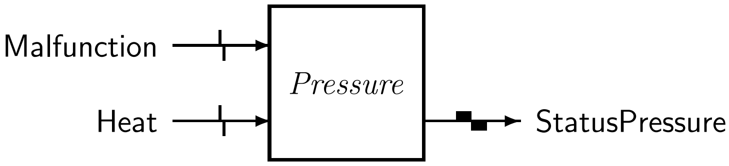


Figure 6.11: Block diagram of the TCES *Pressure*.

used to model the pressure inside the condenser. Initially, we have normal pressure. If a malfunction of the condenser occurs, the event *mal* is received via component Malfunction, and we are in danger, since steam is still produced in tank 1. A timer is used to wait for 8 time units before the pressure may get too high and the condenser explodes. If the event *off* is received early enough from *Heater* via component Heat, the steam production has been stopped, and we are safe again.

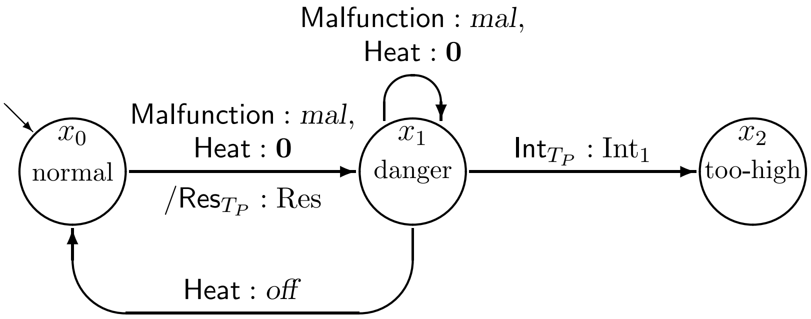


Figure 6.12: Discrete part of the TCES *Pressure*.

We define the TCES

$$\mathit{Pressure} = (\emptyset, \{\mathsf{Malfunction}, \mathsf{Heat}\}, \{\mathsf{StatusPressure}\}, \emptyset, \{T_P\}, \mathcal{D}_{\mathit{Pressure}})$$

with $\alpha(\mathsf{Malfunction}) = \{\mathbf{0}, \mathit{mal}\}$, $\alpha(\mathsf{Heat}) = \{\mathbf{0}, \mathit{off}\}$, $\alpha(\mathsf{StatusPressure}) = \{\text{normal}, \text{danger}, \text{too-high}\}$, $\Theta(T_P) = (8)$, and

$$\begin{aligned}\mathcal{D}_{\mathit{Pressure}} = (&\{\mathsf{Int}_{T_P}\}, \{\mathsf{Malfunction}, \mathsf{Heat}, \mathsf{Ala}_{T_P}\}, \{x_0, x_1, x_2\},\\ &\{\mathsf{StatusPressure}\}, \{\mathsf{Res}_{T_P}\}, f_P, g_P, h_P, \{x_0\}).\end{aligned}$$

The functions f_P, g_P and h_P are given by Figure 6.12. In the unlike case the malfunction occurs and the heater stops producing steam at the same time, we are safe and stay in state x_0. Like in Figure 6.4, the transition from state x_1 to x_2 may be taken any time after timer T_P has reached its threshold, and the last safe possibility to receive *off* must be exactly 8 time units after *mal* was sent by *Condenser*.

Again, the self-loop edge from x_1 to x_1 is only used to prevent the forcing of the transition from x_1 to x_2 by further $\mathsf{Malfunction} : \mathit{mal}$ events.

6.2.1.6 The TCES *Controller*

Figure 6.13 shows the block diagram of the TCES *Controller*. It works as

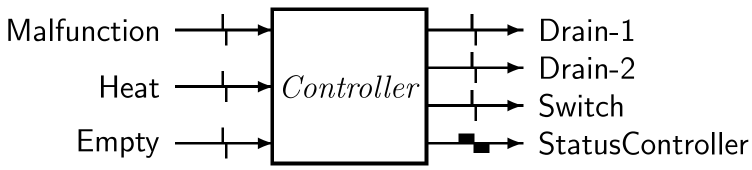


Figure 6.13: Block diagram of the TCES *Controller*.

follows: When the event *mal* is received from *Condenser*, the event *drain* is sent via component Drain-2 to *Tank-2* within 1 time unit, and tank 2 is drained. Then the heater is switched off by sending the event *off* via component Switch to *Heater*. This must happen more than 2 time units after *drain* was sent, and within 3 time units after *mal* was received. If the *empty* event arrives from *Tank-2* earlier, *off* is sent immediately. Then we wait for the event *off* to arrive from *Heater* and *empty* to arrive from *Tank-2* (if it has not been received yet). Exactly 4 time units after the *empty* event was received from *Tank-2*, the *drain* event is sent to *Tank-1*, tank 1 is emptied, and the system is safe.

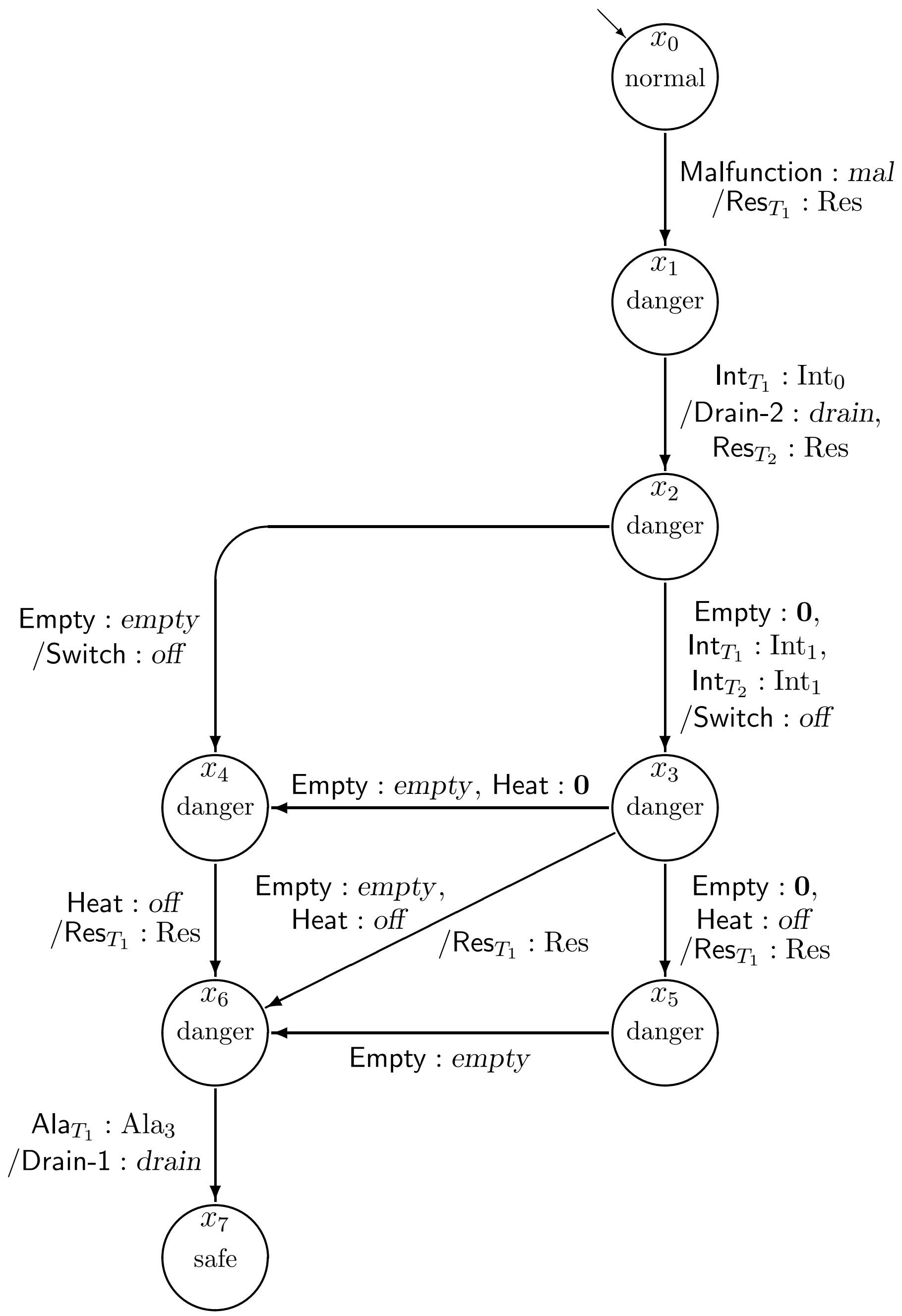


Figure 6.14: Discrete part of the TCES *Controller*.

We define the TCES

$$Controller = (\emptyset, \{\mathsf{Malfunction}, \mathsf{Heat}, \mathsf{Empty}\}, \{\mathsf{StatusController}\},\\ \{\mathsf{Drain\text{-}1}, \mathsf{Drain\text{-}2}, \mathsf{Switch}\}, \{T_1, T_2\}, \mathcal{D}_{Controller})$$

with $\alpha(\mathsf{Malfunction}) = \{\mathbf{0}, mal\}$, $\alpha(\mathsf{Heat}) = \{\mathbf{0}, \mathit{off}\}$, $\alpha(\mathsf{StatusController}) = \{\text{normal}, \text{danger}, \text{safe}\}$, $\alpha(\mathsf{Empty}) = \{\mathbf{0}, empty\}$, $\Theta(T_1) = (1, 3, 4)$, $\Theta(T_2) = (2)$, and

$$\mathcal{D}_{Controller} = (\{\mathsf{Int}_{T_1}, \mathsf{Int}_{T_2}\}, \{\mathsf{Malfunction}, \mathsf{Heat}, \mathsf{Empty}, \mathsf{Ala}_{T_1}, \mathsf{Ala}_{T_2}\},\\ \{x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7\}, \{\mathsf{StatusController}\},\\ \{\mathsf{Drain\text{-}1}, \mathsf{Drain\text{-}2}, \mathsf{Switch}, \mathsf{Res}_{T_1}, \mathsf{Res}_{T_2}\}, f, g, h, \{x_0\}).$$

The functions f, g and h are given by Figure 6.14. To keep the transition diagram simple, the following three self-loop edges are not drawn in Figure 6.14: One edge at state x_1 with label "$\mathsf{Ala}_{T_1} : \mathbf{0}$", an edge at state x_2 labeled with "$\mathsf{Empty} : \mathbf{0}$, $\mathsf{Ala}_{T_1} : \mathbf{0}$", and another edge at x_2 labeled with "$\mathsf{Empty} : \mathbf{0}$, $\mathsf{Ala}_{T_1} : \text{Ala}_1$". These edges ensure that the transition from state x_1 to x_2 is forced only by the Ala_1 event of timer T_1, and the transition from state x_2 to x_3 is forced only by the Ala_2 event of timer T_1.

We take a look at the timing taking place in *Controller*. Two timers, T_1 with threshold list $(1, 3, 4)$ and T_2 with a single threshold (2), are used. The transition from x_0 to x_1 resets timer T_1. Timer T_1 is used to measure the time elapsed since event *mal* occurred. The transition from x_1 to x_2 is enabled only while the value of T_1 is below T_1's first threshold, which is 1. When the value of T_1 is 1, the alarm event of T_1 forces the transition from x_1 to x_2 to be taken, since we have no transition from x_1 to itself labeled with $\mathsf{Ala}_{T_1} : \text{Ala}_1$. During this transition, timer T_2 is reset, and starts counting the time since the event *drain* was sent to *Tank-2*.

In state x_2, we either wait for the event *empty* to be sent by *Tank-2*, forcing us to go to state x_4, or we wait for a certain time to elapse before we go to state x_3. As said before, the point in time we are waiting for is *at least* 2 time units after *drain* was sent, and *at most* 3 time units after *mal* was received. Thus, both timer values have to be beyond their first threshold, denoted by the condition symbol Int_1. The edge from state x_2 to state x_3 enables the transition under these conditions. If the value of timer T_1 reaches 3, we have to take the transition from x_2 to x_3. This is forced by the event Ala_2 generated by T_1. Since we have no edge from x_2 to itself labeled with $\mathsf{Ala}_{T_1} : \text{Ala}_2$, we cannot stay in state x_2 and must go to state x_3.

In states x_3 and x_5, we wait for the event *empty* from *Tank-2*, and in states x_3 and x_4, we wait for the event *off* from *Heater*. Consequently, we have already received *empty* if we are in x_4 or x_6, and if we are in x_5 or x_6, we have already received *off*. If the *off* event arrives, timer T_1 is reset. After 4 time units, we make a transition from state x_6 to x_7 while sending the *drain* event to *Tank-1*.

6.3 Verification

Verification means checking if a system fulfills some desired properties. The level of formalization in such checks has a wide range from "good faith" (no formal reasoning at all) to the use of formal proof systems like Hoare style assertions, automated proofs with systems like PVS [ORS92], or model checkers like KRONOS [OY93], UPPAAL [BLL+96], or HyTech [HHWT95, HH95].

We take a look at some techniques suitable for our TCES model of the evaporator system. As said before, we are interested in verifying the following three properties:

1. If a cooling malfunction in the condenser occurs, the heater must be switched off fast enough such that the condenser does not explode.

2. If the heater is switched off, tank 1 is drained into tank 2 before the solution gets solid.

3. If valve 2 is opened to drain tank 1 into tank 2, tank 2 has to be empty.

From the definition of our modules it its clear that the "bad" things that should not happen only occur when control state x_2 is reached in the TCES *Pressure*, x_2 is reached in *Tank-1*, or x_3 is reached in *Tank-2*. Furthermore, we know from the definition of the condition output functions that reaching one of these states is equivalent with sending the too-high symbol via component StatusPressure, solid via StatusTank-1, or overflow via StatusTank-2.

Thus, there are two possibilities to look for the unwanted things: We can either check the reachability of the control states mentioned above, or we can look for the occurrence of the three condition symbols too-high, solid, and overflow in the signals of the compound system.

6.3.1 Using Timing Diagrams

Our first attempt to verify the three properties makes use of *timing diagrams,* which show a graphical representation of the system's signals. They are used to visualize the timing dependencies of different signals. If some signals are related, i.e., some symbol in a signal can only occur in a certain time range before or after some other symbol occurs in another signal, this relationship is shown in the diagram.

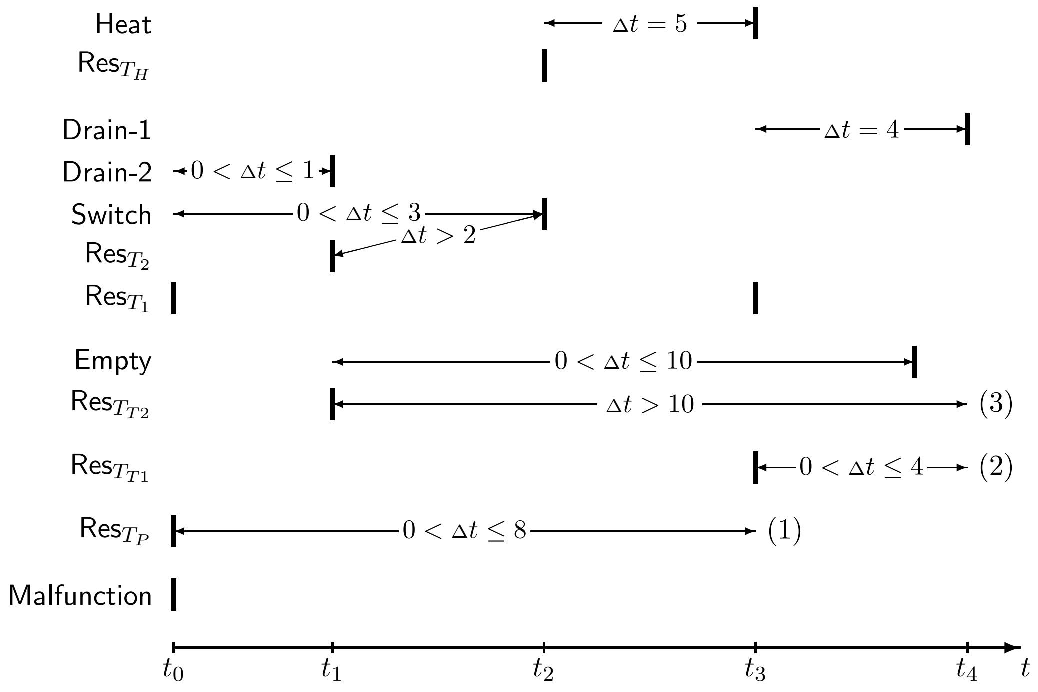


Figure 6.15: A timing diagram for the evaporator system.

Figure 6.15 shows a timing diagram for the evaporator system. Vertically stacked we have all event components in our system; grouped from bottom to top are the output components names of $\mathcal{D}_{\mathit{Condenser}}$, $\mathcal{D}_{\mathit{Pressure}}$, $\mathcal{D}_{\mathit{Tank\text{-}1}}$, $\mathcal{D}_{\mathit{Tank\text{-}2}}$, $\mathcal{D}_{\mathit{Controller}}$, and $\mathcal{D}_{\mathit{Heater}}$. The right part of the diagram shows the occurrences of all proper events of the components' alphabets. Time progress is drawn from left to right. Note that we have no linear scaling in the diagram.

The first proper event occurring in our system is Malfunction : *mal*[1] from *Condenser*, denoting that the condenser starts malfunctioning. We denote this point in time with t_0. The event *mal* triggers two other events: In *Pressure*, we change from state x_0 to x_1, resetting timer T_P. After more than 8 time units the transition to state x_2 may be taken, violating property 1 (the condenser should not explode). Thus, we must verify that the event Heat : *off* occurs within 8 time units. We denote this by (1) in the diagram.

The event *mal* also triggers in *Controller* the transition from state x_0 to x_1, resetting timer T_1. Within 1 time unit, *Controller* takes the transition

[1] We use the same notation for events as in the graphical representation for TCESs.

from x_1 to x_2 at time $t_1 \in (t_0, t_0 + 1]$, sending the event $\textsf{Drain-2} : \mathit{drain}$ to *Tank-2* and resetting timer T_2. *Tank-2* also resets its timer T_{T_2}, and within 10 time units, the event $\textsf{Empty} : \mathit{empty}$ will be sent. If the event $\textsf{Drain-1} : \mathit{drain}$ occurs within 10 time units after t_1, *Tank-2* may change to state x_3, violating property 3 (tank 2 should not overflow). Therefore, we must ensure that the event $\textsf{Drain-1} : \mathit{drain}$ occurs more than 10 time units after t_1. We denote this by (3) in the diagram.

In Figure 6.15, we do not consider the case that in *Controller* the transition from state x_2 to x_4 is taken, which is possible if the draining of tank 2 takes 3 or less time units. A different timing diagram has to be drawn for this case.

Controller moves on from state x_2 to x_3, generating the event $\textsf{Switch} : \mathit{off}$, if the value of timer T_1 is at most 3 *and* T_2 shows more than 2 time units. Therefore, we have two edges ending at the $\textsf{Switch} : \mathit{off}$ event at time t_2 in our timing diagram. The timing is described by $t_2 \leq 3$ and $t_2 > t_1 + 2$. The *off* event triggers the transition from state x_0 to x_1 in *Heater*, which resets its timer T_H. After exactly 5 time units, *Heater* changes its state to x_2, sending the event $\textsf{Heat} : \mathit{off}$ at time $t_3 = t_2 + 5$.

In the meantime, *Controller* uses its states x_3, x_4, x_5 and x_6 to wait for the events $\textsf{Heat} : \mathit{off}$ and $\textsf{Empty} : \mathit{empty}$ to occur. When the *off* event is occurs at time t_3, *Controller* resets its timer T_1, and *Tank-1* resets its timer T_{T_1} while changing the state to x_1. After more than 4 time units, *Tank-1* may go to state x_2, violating property 2 (the solution in tank 1 must not become solid). Thus, we must verify that the event $\textsf{Drain-1} : \mathit{drain}$ occurs within 4 time units. We denote this by (2) in the diagram.

Let us look at time t_2 again. *Controller* is in state x_3. To get to state x_7, we must get to state x_6 within 4 time units, since the event $\textsf{Ala}_{T_1} : \mathrm{Ala}_3$ occurs at time $t_4 = t_3{+}4$, and if we are not at state x_6 at time t_4, event $\textsf{Drain-1} : \mathit{drain}$ is not sent, eventually leading to a violation of property 2. Therefore, event $\textsf{Empty} : \mathit{empty}$ must occur between t_2 and t_4. Our diagram considers the case that *empty* occurs at a time $t > t_2$. We have to show $t < t_4$. From the timing in *Tank-2*, we know that $t - t_1 \leq 10$, and we also know $t_4 = t_3 + 4$, $t_3 = t_2 + 5$, and $t_2 > t_1 + 2$. This yields $t_4 = t_2 + 9 > t_1 + 11 > t_1 + 10 \geq t$.

Thus, *Controller* is at state x_6 when the event $\textsf{Ala}_{T_1} : \mathrm{Ala}_3$ occurs at time t_4 and go to state x_7, sending $\textsf{Drain-1} : \mathit{drain}$.

It remains to check the requirements (1), (2), and (3) in the diagram. By $0 < t_2 \leq 3$ and $t_3 = t_2 + 5$, we have $0 < t_3 \leq 8$, showing (1). By $t_4 = t_3 + 4$, we have (2): $0 < t_4 - t_3 \leq 4$. And $t_2 > t_1 + 2$, $t_3 = t_2 + 5$, and $t_4 = t_3 + 4$ yields $t_4 - t_1 > 11$, which is sufficient for (3).

We have proven the three properties we expected from our system. We left out another timing diagram which is more simple, but has also to be checked. The advantages of this visual way of handling timing restrictions are not obvious if too many signals are involved, which leads to very large diagrams. Nevertheless, timing diagrams are widely used for various purposes like the verification of communication protocols, schedules for an automated production, or the design of railroad timetables.

6.3.2 Using Timed Automata

In [Huu98], an effective transformation from a TCES to an equivalent timed automaton is given (see also [HLU+97b, HLU+97a]). Equivalence in this case is expressed by means of a relation between timed languages of timed automata and sets of behaviors of TCESs. This transformation has been used in the KONDISK project to build six timed automata which are equivalent to the six TCES modules of the evaporator system [HLL+98].

Since the product of these timed automata is equivalent to the discrete parallel interconnection of the TCESs, and since the state spaces are also comparable, the verification tool KRONOS [OY93] has been used as a model checker to verify that none of the three "bad" states is reachable. Another tool that was used successfully was HyTech [HHWT95, HH95].

6.3.3 Using the Parallel Interconnection Operator

Since we modeled our system as a set of interconnected TCESs, it is sensible to apply the parallel interconnection operator to obtain one TCES $\mathcal{T}$ having the semantics of the compound system.

When we have constructed such a TCES $\mathcal{T}$, we can use both techniques mentioned above to verify the desired properties, or we can do the verification on the semantics of $\mathcal{T}$. The semantic formalization of the three properties is simple, since we have introduced condition output signals which denote the control states of the modules. Thus, a signal-based version of the properties just says that the "bad" states are never reached:

1. $\forall s \in S_{\mathcal{T}} : \forall t \in \mathbb{R}_{\geq 0} : s_{\bar{\mathsf{Y}}}(t)(\mathsf{StatusPressure}) \neq$ too-high
2. $\forall s \in S_{\mathcal{T}} : \forall t \in \mathbb{R}_{\geq 0} : s_{\bar{\mathsf{Y}}}(t)(\mathsf{StatusTank\text{-}1}) \neq$ solid
3. $\forall s \in S_{\mathcal{T}} : \forall t \in \mathbb{R}_{\geq 0} : s_{\bar{\mathsf{Y}}}(t)(\mathsf{StatusTank\text{-}2}) \neq$ overflow

If we try to check the multiple solution property, the first part is trivial: Since the have no condition signals *between* the modules at all, the set of conditions names C^{io} (see Definition 4.12) is empty. Thus, the set of evaluations y of C^{io} we have to find in the multiple solution property is always $\Sigma_{\mathsf{C}^{\mathsf{io}}}$ which contains just the evaluation $y = \emptyset$.

In the second part, we have to find solutions for the events between the modules. Since every event generated by a module in our system is received by one or more modules, we have $\mathsf{E}^{\mathsf{io}} = \{\mathsf{Malfunction}, \mathsf{Drain\text{-}1}, \mathsf{Drain\text{-}2}, \mathsf{Empty}, \mathsf{Switch}, \mathsf{Heat}\}$. Now we have to find for any combination of pre-states in the modules and any combination of post-states in the modules at least one $z \in \Sigma_{\mathsf{E}^{\mathsf{io}}}$ which is a solution for the events between the modules.

This is the point where we face the state explosion problem. Each module has between 2 and 8 states, leading to a total of 2304 possible combinations of states. Since any combination of a pre- and a post-state has to be checked, we end up with over 5 million combinations. And for each of the combinations, the number of candidates for a solution is $|\Sigma_{\mathsf{E}^{\mathsf{io}}}| = 64$.

However, if we take a close look at the TCES definitions of our modules, we do *not* need to combine *all* six modules to verify the properties. Take property 3: It says that tank 2 has to be empty before tank 1 is drained. Looking at Figure 6.6, we see that tank 2 is empty (state x_1) if and only if *Tank-2* has sent Empty : *empty* and $\mathsf{Drain\text{-}1}$: *drain* has not yet been received. On the other hand, Figure 6.14 shows that there is only one transition where $\mathsf{Drain\text{-}1}$: *drain* is sent; this is when changing from state x_6 to x_7. Furthermore, state x_6 can only be reached if Empty : *empty* was received earlier.

We conclude that for verifying property 3, it is sufficient to combine the TCESs *Tank-2* and *Controller* by using the parallel interconnection operator. The resulting TCES can be checked against property 3. Analogously, the verification of property 1 only needs the TCESs *Condenser*, *Pressure*, *Heater*, and *Controller*, and for property 2, we need *Heater*, *Tank-1*, *Tank-2*, and *Controller*. This leads to a significant reduction of states and evaluations to be looked at.

Unfortunately, all three compositions above are quite large; therefore, we only show the discrete parallel interconnection of *Pressure* and *Condenser* to give an impression of how this works.

6.3.3.1 Composing *Pressure* and *Condenser*

Recall the definitions of the TCES *Pressure* and the DCES *Condenser*, especially their transition and output functions depicted in Figures 6.12 and 6.10. We apply the discrete parallel interconnection with *unique* solutions to *Pressure* and *Condenser*. Note that *Condenser* can be expressed as a TCES without clocks (see Lemma 5.7).

The validity of the unique solution property can be checked easily. Defining $PressCond = Pressure \| Condenser$, we get

$$
\begin{aligned}
PressCond = (&\emptyset, \{\mathsf{Heat}\}, \{\mathsf{StatusPressure}, \mathsf{StatusCondenser}\}, \\
&\{\mathsf{Malfunction}\}, \{T_P\}, \mathcal{D}_{PressCond})
\end{aligned}
$$

with

$$
\begin{aligned}
\mathcal{D}_{PressCond} = (&\{\mathsf{Int}_{T_P}\}, \{\mathsf{Heat}, \mathsf{Ala}_{T_P}\}, \\
&\{(x_0, x_0), (x_1, x_0), (x_2, x_0), (x_0, x_1), (x_1, x_1), (x_2, x_1)\}, \\
&\{\mathsf{StatusPressure}, \mathsf{StatusCondenser}\}, \{\mathsf{Malfunction}, \mathsf{Res}_{T_P}\}, \\
&f_{PC}, g_{PC}, h_{PC}, \{(x_0, x_0)\}).
\end{aligned}
$$

Figure 6.16 shows the block diagram of the TCES *PressCond*.

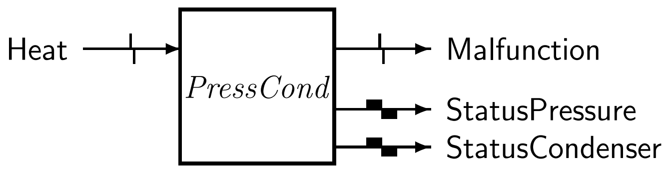


Figure 6.16: Block diagram of the TCES *PressCond*.

The functions f_{PC}, g_{PC}, and h_{PC} are depicted in Figure 6.17. In the upper half of each circle, there is the first component of the state identifier and f_{PC} (from *Pressure*), and in the lower part, there are the second components (from *Condenser*). An interesting transition is the one from state (x_0, x_0) to (x_0, x_1). It shows that we are not in danger when *mal* and *off* occur at the same time. This transition from x_0 to x_0 does also exist in *Pressure*, but there is no explicit edge for it in Figure 6.12. Note that the self-loop edge at state (x_0, x_0) allows us to stay at this state if the Heat : *off* event occurs. Without this edge, we must take the transition to (x_0, x_1) while sending the Malfunction : *mal* event.

States (x_1, x_0) and (x_2, x_0) are not reachable, since *Pressure* cannot leave x_0 without receiving the *mal* event. We simply can remove these states from *PressCond*, yielding the system in Figure 6.18 having the same semantics as *PressCond*. Removing states that cannot be reached is another way of keeping the state space small when using the parallel composition operator.

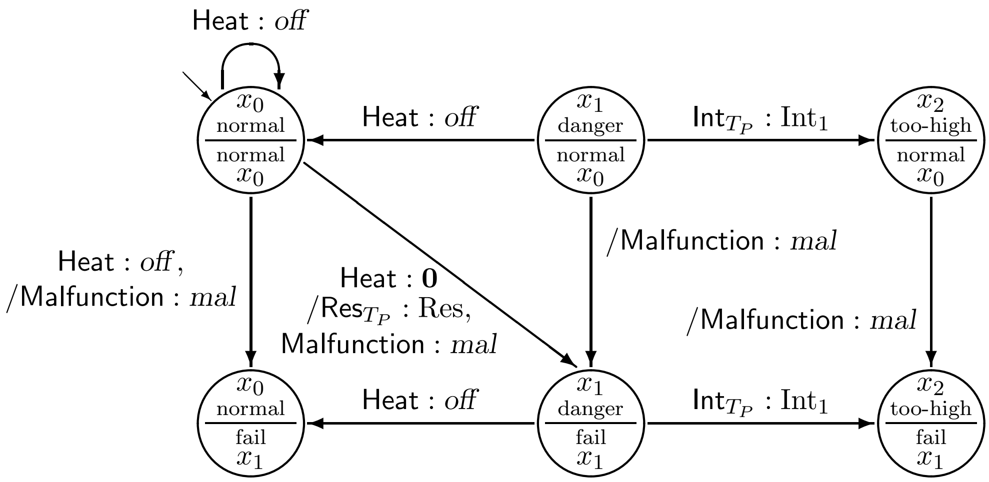


Figure 6.17: Discrete part of the TCES *PressCond*.

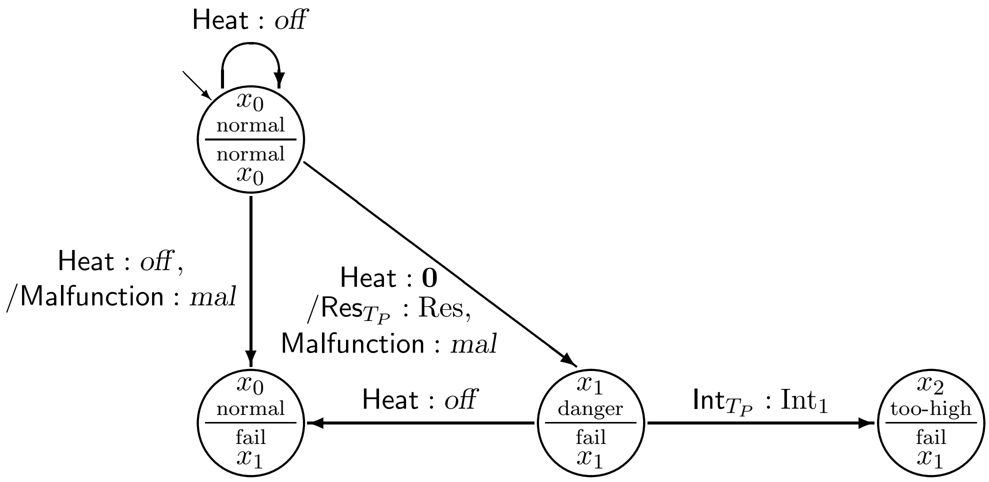


Figure 6.18: The reachable discrete part of the TCES *PressCond*.

Chapter 7

Verification of DCESs Using SMV

In this chapter we show how to use the model checker SMV [McM92, McM93] to verify discrete condition/event systems. A set of DCESs is translated into an SMV input file describing the parallel interconnection of these DCESs. Then the SMV model checker is used to verify properties of the system which are described as a CTL formula [CE81].

7.1 Transforming a system of DCESs into the SMV framework

Given a set of discrete condition/event systems, we transform each system into an SMV module. While communication between condition/event systems is achieved by using condition and event signals, SMV modules communicate via shared variables. Therefore, we introduce a variable for each event output occurring in one of the systems. This variable is local to and written by the module of the system generating the respective event, and it can be read by the modules of other systems having this event as input.

Conditions are handled differently. Since a condition output symbol only depends on a system's current state and its current condition input symbols, we do not need to introduce a variable, but use SMV's `DEFINE` declaration to describe the mapping of the state and the input symbols to the output symbol in a direct way. As an advantage of this construction, no variable is needed to store the condition output symbol, which can significantly decrease memory and time consumption during the model checking process. The `DEFINE` declaration will be given in the module of the system generating the respective condition, and the modules of systems having this condition as input can access it.

7.1.1 Conversion of Identifiers

The syntax of SMV only allows letters (`a`, ..., `z`, `A`, ..., `Z`), digits (`0`, ..., `9`), the hyphen (`-`), and the underscore symbol (`_`) in its identifiers, so some conversion might be necessary when transforming a DCES into SMV code. In the following we implicitly assume that such a conversion is performed when needed. This will be the case if a non-`typewriter` font face appears in the SMV code shown.

7.1.2 Translating a DCES into an SMV module

Let $\mathcal{D} = (\bar{\mathsf{U}}, \bar{\mathsf{V}}, X, \bar{\mathsf{Y}}, \bar{\mathsf{Z}}, f, g, h, X_0)$ be a named DCES with

$$\bar{\mathsf{U}} = \{\mathsf{U}_1, \ldots, \mathsf{U}_{n_U}\}, \bar{\mathsf{V}} = \{\mathsf{V}_1, \ldots, \mathsf{V}_{n_V}\}, X = \{x_1, \ldots, x_n\},$$
$$\bar{\mathsf{Y}} = \{\mathsf{Y}_1, \ldots, \mathsf{Y}_{n_Y}\}, \bar{\mathsf{Z}} = \{\mathsf{Z}_1, \ldots, \mathsf{Z}_{n_Z}\}, X_0 = \{x_{0,1}, \ldots, x_{0,m}\},$$

and some identifier *Name* assigned to the system.

First we generate the module header. The name of the module is composed by appending `_module` to *Name*. The parameters of the module are all input names of $\mathcal{D}$.

$$\texttt{MODULE}\ \mathit{Name}\texttt{\_module(}\mathsf{U}_1\texttt{,}\ldots\texttt{,}\mathsf{U}_{n_U}\texttt{,}\mathsf{V}_1\texttt{,}\ldots\texttt{,}\mathsf{V}_{n_V}\texttt{)}$$

If there are no parameters, i.e., $\bar{\mathsf{U}} = \bar{\mathsf{V}} = \emptyset$, we have to omit the parentheses.

We need one local variable `state` to store the control state of the system:

```
VAR
  state: {x_1, ..., x_n};
```

For each of the output event names of $\mathcal{D}$ we introduce one variable storing the symbol of that component:

```
VAR
  Z_1: {symbols(Z_1)};
  ⋮
  Z_nZ: {symbols(Z_nZ)};
```

Here *symbols*(N) denotes a comma separated list of the symbols in $\alpha(\mathsf{N})$. If N is an event name (here this is always the case), this list includes `0`, denoting the null symbol **0**. If there are no output events ($\bar{\mathsf{Z}} = \emptyset$), we omit this `VAR` declaration.

The variable `state` is initialized non-deterministically with one of the initial states:

```
INIT
  state in {x_{0,1}, ... ,x_{0,m}}
```

All output event variables are initialized with the null symbol:

```
INIT
  Z_1 = 0 & ... & Z_{n_Z} = 0
```

If there are no event outputs ($\bar{\mathsf{Z}} = \emptyset$), we omit this `INIT` declaration.

For each condition output name Y_i, $i \in \{1, \ldots, n_Y\}$, we examine g to generate a `DEFINE` declaration computing the condition output symbol:

```
DEFINE
  Y_1 :=
    case state=x_0: ... ;
              ⋮
         state=x_n: ... ;
    esac;
  ⋮
  Y_{n_Y} :=
    case state=x_0: ... ;
              ⋮
         state=x_n: ... ;
    esac;
```

The "..." above must be replaced by SMV expressions (consisting of further `case` ... `esac` constructs reading the values of the variables $\mathsf{U}_1, \ldots, \mathsf{U}_{n_U}$) computing the respective condition output symbol for Y_i.

If for some $i \in \{1, \ldots, n_Y\}$ and $j \in \{1, \ldots, n\}$ the condition output symbol for Y_i at state x_j does not depend on the condition inputs (or there are no condition inputs at all, i.e., $\bar{\mathsf{U}} = \emptyset$), then $g(x_j, \cdot)(\mathsf{Y}_i)$ is constant, and we can simply replace the "..." above by this constant. Example 7.2 on page 121 illustrates this. If there are no condition outputs ($\bar{\mathsf{Y}} = \emptyset$), we omit this `DEFINE` declaration.

Finally, we have to implement the transition function f and the event output function h. We make use of SMV's `TRANS` declaration which describes a transition relation between the current variable assignments and the variable assignments after one transition step. A transition step in SMV models a transition of the DCES in the following way: The variable assignments before the step model the control state and the condition input symbols at time t^-, and the variable assignments after the step describe the control state and the events at time t.

Thus, we can access the control state at time t^- by using `state`, a condition input symbol at time t^- by using its name U_i, and for time t, we access the state by using `next(state)` and an event with name N by using `next(N)`.

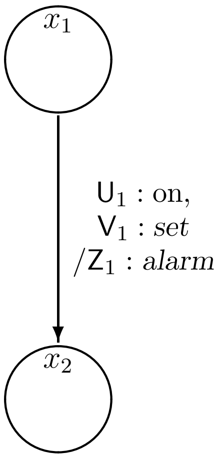
```
TRANS
  (state = x_1) &
  ( (U_1 = u_1 &
     ⋮
     U_{n_U} = u_{n_U} &
     next(V_1) = v_1 &
     ⋮
     next(V_{n_V}) = v_{N_V} &
     next(state) = x'_1 &
     next(Z_1) = h(x_1, x'_1, {V_1 ↦ v_1, ..., V_{n_V} ↦ v_{N_V}})(Z_1) &
     ⋮
     next(Z_{n_Z}) = h(x_1, x'_1, {V_1 ↦ v_1, ..., V_{n_V} ↦ v_{N_V}})(Z_{n_Z}) &
    ) |
    ⋮
  ) |
  ⋮
  (state = x_n) &
  (
   ⋮
  )
```

The inner block from "(U_1 = u_1" to ") |" has to be written for all $u_1 \in \alpha(\mathsf{U}_1)$, $\ldots$, $u_{n_U} \in \alpha(\mathsf{U}_{n_U})$, $v_1 \in \alpha(\mathsf{V}_1)$, $\ldots$, $v_{n_V} \in \alpha(\mathsf{V}_{n_V})$, and $x'_1 \in f(x_1, \{\mathsf{U}_1 \mapsto u_1, \ldots, \mathsf{U}_{n_U} \mapsto u_{N_U}\}, \{\mathsf{V}_1 \mapsto v_1, \ldots, \mathsf{V}_{n_V} \mapsto v_{N_V}\})$. Analogously this has to be done for the states $x_2, \ldots, x_n$.

Example 7.1 Figure 7.1 shows a transition of a DCES and the resulting piece of SMV code. $\diamond$

7.1.3 Combining the SMV modules

Now all modules have to be composed into one interconnected system. We connect the modules in same way as the parallel interconnection in Chapter 4




```
TRANS
  ⋮
  (state = x_1) &
  (
    ⋮
   (U_1 = on &
    next(V_1) = set &
    next(state) = x_2 &
    next(Z_1) = alarm
   ) |
    ⋮
```

Figure 7.1: A DCES transition and its SMV representation.

does: If a module M generates the output for a name N, all other modules having N as an input read the local variable N of M to obtain their input value.

If there are inputs for which there exists no matching output in the system, these inputs are considered to be external, i.e., their values must be provided by some environment. In SMV we define global variables for these external inputs; by default these can take on any value of their respective domains. This leads to a "chaotic" behavior of the environment, which is a typical test situation for open systems. Alternatively, we can use some additional SMV code to control these variables if we want more restrictive behavior of the environment.

We do not have to introduce global variables for outputs, since these are stored in the modules' local variables. If we need to access them for model checking purposes, they can be read directly in a CTL formula (cf. Example 7.3 on page 126).

Let $\mathcal{D}_i = (\bar{\mathsf{U}}_i, \bar{\mathsf{V}}_i, X_i, \bar{\mathsf{Y}}_i, \bar{\mathsf{Z}}_i, f_i, g_i, h_i, X_{0,i})$ be a named DCES, identified as Name_i, for $i \in \{1, \dots, n\}$. Like in Definition 4.12 we define the external inputs to our system:

$$\mathsf{C}^{\mathsf{in}} = \left(\bigcup_{i=1}^{n} \bar{\mathsf{U}}_i\right) \setminus \left(\bigcup_{i=1}^{n} \bar{\mathsf{Y}}_i\right) \text{ and } \mathsf{E}^{\mathsf{in}} = \left(\bigcup_{i=1}^{n} \bar{\mathsf{V}}_i\right) \setminus \left(\bigcup_{i=1}^{n} \bar{\mathsf{Z}}_i\right).$$

We number the elements of these sets like this:

$$\mathsf{C}^{\mathsf{in}} = \{\mathsf{C}^{\mathsf{in}}_1, \dots, \mathsf{C}^{\mathsf{in}}_l\} \text{ and } \mathsf{E}^{\mathsf{in}} = \{\mathsf{E}^{\mathsf{in}}_1, \dots, \mathsf{E}^{\mathsf{in}}_m\}.$$

The SMV program starts with the `main` module.

```
MODULE main
```

Now we define the variables of this module. These variables are the global variables for the other modules.

We introduce a global variable for each element of C^{in} and E^{in}:

$$
\begin{array}{l}
\texttt{VAR} \\
\quad \mathsf{C}^{\mathsf{in}}_1\texttt{:}\ \{symbols(\mathsf{C}^{\mathsf{in}}_1)\}\texttt{;} \\
\quad \vdots \\
\quad \mathsf{C}^{\mathsf{in}}_l\texttt{:}\ \{symbols(\mathsf{C}^{\mathsf{in}}_l)\}\texttt{;} \\
\quad \mathsf{E}^{\mathsf{in}}_1\texttt{:}\ \{symbols(\mathsf{E}^{\mathsf{in}}_1)\}\texttt{;} \\
\quad \vdots \\
\quad \mathsf{E}^{\mathsf{in}}_m\texttt{:}\ \{symbols(\mathsf{E}^{\mathsf{in}}_m)\}\texttt{;}
\end{array}
$$

If there are no external inputs ($\mathsf{C}^{\mathsf{in}} = \mathsf{E}^{\mathsf{in}} = \emptyset$), we omit this `VAR` declaration.

Since conditions are not given as variables but only as `DEFINE` declarations, we have to introduce the condition symbols we want use in our system to SMV's syntax checker. Therefore, we define a dummy variable that lists all these symbols:

$$
\begin{array}{l}
\texttt{VAR} \\
\quad \texttt{conditionsymbols:}\ \{symbols(\bigcup_{i=1}^{n} \bar{\mathsf{U}}_i \cup \bar{\mathsf{Y}}_i)\}\texttt{;}
\end{array}
$$

If there are no conditions in the system, we omit this `VAR` declaration.

Now we instantiate all the DCESs as modules. This is the place where the outputs of one module can be connected to the inputs of other modules. We define a helpful function

$$
Prod(\mathsf{N}) = \begin{cases} \mathsf{N}, \text{ if } \mathsf{N} \in \mathsf{C}^{\mathsf{in}} \cup \mathsf{E}^{\mathsf{in}} \\ Name_i\texttt{.}\mathsf{N}, \text{ if } \mathsf{N} \in \bar{\mathsf{Y}}_i \cup \bar{\mathsf{Z}}_i \text{ for an } i \in \{1, \ldots, n\} \end{cases},
$$

which, given a name N, either delivers this name itself if N is an external input, or composes the identifier of the DCES producing the signal named N, a dot, and N. Thus, *Prod* is an SMV expression pointing either to the global variable we defined for that component, or giving us access to the local variable of the module containing the output symbol we are looking for.

Now we can instantiate all the modules, and let *Prod* do the work of making the right connections:

$$
\begin{array}{l}
\texttt{VAR} \\
\quad \mathit{Name_1}:\ \mathit{Name_1}\texttt{\_module(}\mathit{Prod}(\mathsf{U}^1_1)\texttt{, ... ,}\mathit{Prod}(\mathsf{U}^1_{n^1_U})\texttt{,} \\
\qquad\qquad\qquad\qquad\qquad \mathit{Prod}(\mathsf{V}^1_1)\texttt{, ... ,}\mathit{Prod}(\mathsf{V}^1_{n^1_V})\texttt{);} \\
\quad \vdots \\
\quad \mathit{Name_n}:\ \mathit{Name_n}\texttt{\_module(}\mathit{Prod}(\mathsf{U}^n_1)\texttt{, ... ,}\mathit{Prod}(\mathsf{U}^n_{n^1_U})\texttt{,} \\
\qquad\qquad\qquad\qquad\qquad \mathit{Prod}(\mathsf{V}^n_1)\texttt{, ... ,}\mathit{Prod}(\mathsf{V}^n_{n^n_V})\texttt{);}
\end{array}
$$

with $\bar{\mathsf{U}}_i = \{\mathsf{U}^i_1, \ldots, \mathsf{U}^i_{n^i_U}\}$ and $\bar{\mathsf{V}}_i = \{\mathsf{V}^i_1, \ldots, \mathsf{V}^i_{n^i_V}\}$, for $i \in \{1, \ldots, n\}$. If $\mathcal{D}_i$ has no inputs ($\bar{\mathsf{U}}_i = \bar{\mathsf{V}}_i = \emptyset$), its module has no parameters, and we have to omit the parentheses.

The last thing we have to do is to initialize the external input events with the null event:

$$
\begin{array}{l}
\texttt{INIT} \\
\quad \mathsf{E}^{\mathsf{in}}_1 \texttt{ = 0 \& ... \& } \mathsf{E}^{\mathsf{in}}_m \texttt{ = 0}
\end{array}
$$

If there are no external input events ($\mathsf{E}^{\mathsf{in}} = \emptyset$), we omit this `INIT` declaration.

This completes the transformation of a set of DCESs into an SMV input file. One or more `SPEC` declarations can now be used to provide the SMV model checker with some properties of the system we want to check. Section 7.2 discusses this is detail.

Example 7.2 Figure 7.2 shows the connected block diagrams of the TCES *Controller* and TCES *Tank-2* presented in Chapter 6. Since we need DCESs here instead of TCESs, we completely ignore the timers in this example. If we apply the instructions above to these two systems, we get the SMV code shown in Figures 7.3, 7.4, and 7.5. The `TRANS` parts in the modules are left out. ◇

7.1.4 Well-definedness

The transformation of a set of DCESs into SMV code can be regarded as an implementation of the parallel interconnection introduced in Chapter 4. However, the parallel interconnection is only well-defined if the set of DCESs is consistent (Definition 4.11), and if the unique solution property (Theorem 4.13) or the multiple solution property (Theorem 4.27) holds.

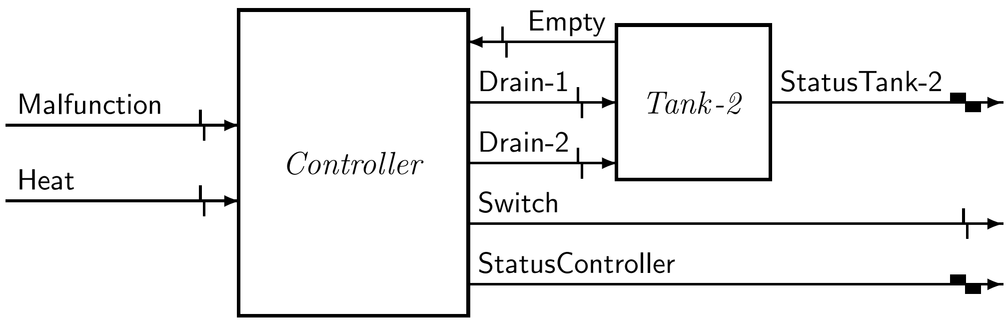


Figure 7.2: Block diagram of *Controller* and *Tank-2*.

```
MODULE main

VAR
  Malfunction: {0,mal};
  Heat: {0,off};

VAR
  conditionsymbols: {normal,danger,safe,full,empty,draining,overflow};

VAR
  Controller: Controller_module(Malfunction,Heat,Tank2.Empty);
  Tank2: Tank2_module(Controller.Drain1,Controller.Drain2);

INIT
  Heat=0 & Malfunction=0
```

Figure 7.3: The SMV module `main`.

```
MODULE Controller_module(Malfunction,Heat,Empty)

VAR
  state : {x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7};

VAR
  Drain1: {0,drain};
  Drain2: {0,drain};
  Switch: {0,off};

INIT
  state in {x_0}

DEFINE
  StatusController :=
    case
      state=x_0: normal;
      state=x_1: danger; state=x_2: danger; state=x_3: danger;
      state=x_4: danger; state=x_5: danger; state=x_6: danger;
      state=x_7: safe;
    esac;

INIT
  Switch=0 & Drain1=0 & Drain2=0
```

Figure 7.4: The SMV module `Controller`.

```
MODULE Tank2_module(Drain1,Drain2)

VAR
  state : {x_0,x_1,x_2,x_3};

VAR
  Empty: {0,empty};

INIT
  state in {x_0}

DEFINE
  StatusTank2 :=
    case
      state=x_0: full;
      state=x_1: empty;
      state=x_2: draining;
      state=x_3: overflow;
    esac;

INIT
  Empty=0
```

Figure 7.5: The SMV module `Tank2`.

Consistency can be checked easily by inspecting the systems' output names. The unique/multiple solution property, which must be fulfilled to avoid algebraic loops in the system, is much more difficult to check. Some algebraic loops can be detected by SMV. Since we implement conditions and events differently in SMV, we look at them separately when checking the unique/multiple solution property.

Condition outputs are given as `DEFINE` declarations, and SMV allows no circular definitions there. Since algebraic loops are always based on a circular definition, all algebraic loops using condition signals are detected by SMV, and the input file will be rejected with a "recursively defined" error message. Even if there is a circular definition without an algebraic loop, like in the cases 2 and 3 of Example 4.14, SMV will reject the input file. This is no severe restriction, since in most systems the condition output does not depend on the condition input, but is only an abstraction of the system's control state. In this case there are no condition-based algebraic loops.

Event outputs are given within the `TRANS` declaration. If there is an algebraic loop based on events, the transition relation generated by SMV is not total, i.e., for some inputs and some control states of the modules, there is no successor state, and a deadlock occurs. SMV detects such a deadlock state only if this state is unavoidable, i.e., it is eventually reached no matter how nondeterministic choices (including external inputs) are made. For example, if an external input event $\mathsf{Update} : 1$ causes an algebraic loop, and Update is not fixed by some SMV construct, SMV does not generate the $\mathsf{Update} : 1$ event, and the deadlock state remains undetected. To be on the safe side, checking the unique or multiple solution property (beforehand or using SMV) is recommended.

7.1.5 A Complete Example

We illustrate the translation of a DCES into SMV code using the following system:

$$\begin{aligned} \mathit{Switch} = (&\{\mathsf{Request}\}, \{\mathsf{Update}\}, \{x_0, x_1\}, \{\mathsf{Status}\}, \{\mathsf{Change}\}, \\ &f, g, h, \{x_0\}) \end{aligned}$$

with $\alpha(\mathsf{Request}) = \{\text{start}, \text{stop}\}$, $\alpha(\mathsf{Status}) = \{\text{off}, \text{on}\}$, and $\alpha(\mathsf{Update}) = \alpha(\mathsf{Change}) = \{\mathbf{0}, 1\}$. Figure 7.6 shows the block diagram of the DCES *Switch*.

The functionality of *Switch*, defined by Figure 7.7, is as follows: The system is either in the control state x_0 or x_1, and this information is passed to the environment by the condition output symbol $\mathsf{Status} : \text{off}$ or $\mathsf{Status} : \text{on}$.

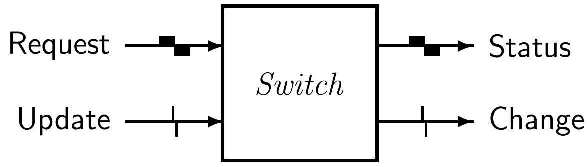


Figure 7.6: Block diagram of the DCES *Switch*.

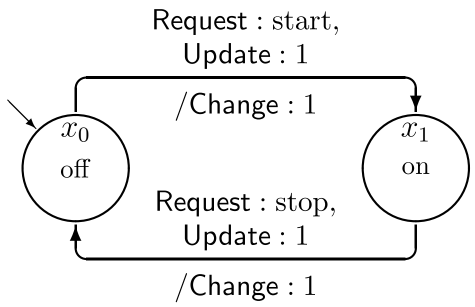


Figure 7.7: The transition system of the DCES *Switch*.

Initially, *Switch* is at state x_0. If the environment wants to change the state of *Switch* to x_1, it sets the condition Request : start and sends the event Update : 1, which immediately triggers the state change in *Switch*. Analogously, Request : stop and Update : 1 will force the state transition back to x_0. Note that changing Request without sending Update : 1 does not change the state.

Whenever the state of *Switch* changes, the event Change : 1 is generated. So if Update : 1 is sent twice without a change of Request inbetween, there is at most one Update : 1 event.

We apply the transformation to SMV on the DCES *Switch*. This leads to the code shown in Figure 7.8.

Obviously the expression representing the transition relation can be written in a more compact way.

7.2 Verification

SMV accepts CTL formulae [CE81] for checking system properties. The syntax used by SMV is given in [McM92, McM93]. Safety and liveness properties can be checked, and the evaluation of formulae can be restricted to fair paths using a second formula to describe all paths which are considered to be fair.

```
MODULE main

VAR  Request: {start,stop};
     Update: {0,1};

VAR  conditionsymbols: {off,on,start,stop};

VAR  Switch: Switch_module(Request,Update);

INIT  Update=0


MODULE Switch_module(Request,Update)

VAR  state: {x_0,x_1};

VAR  Change: {0,1};

INIT  state in {x_0}

DEFINE  Status := case
                    state=x_0: off;
                    state=x_1: on;
                  esac;

INIT  Change=0

TRANS
  (state=x_0) &
  ( (Request=start & next(Update)=0 & next(state)=x_0 & next(Change)=0) |
    (Request=stop  & next(Update)=0 & next(state)=x_0 & next(Change)=0) |
    (Request=start & next(Update)=1 & next(state)=x_1 & next(Change)=1) |
    (Request=stop  & next(Update)=1 & next(state)=x_0 & next(Change)=0)
  ) |
  (state=x_1) &
  ( (Request=start & next(Update)=0 & next(state)=x_1 & next(Change)=0) |
    (Request=stop  & next(Update)=0 & next(state)=x_1 & next(Change)=0) |
    (Request=start & next(Update)=1 & next(state)=x_1 & next(Change)=0) |
    (Request=stop  & next(Update)=1 & next(state)=x_0 & next(Change)=1)
  )
```

Figure 7.8: The SMV code for the DCES *Switch*.

Example 7.3 We want to verify that whenever the DCES *Switch* generates the Change : 1 event, an Update : 1 event must occur simultaneously. To verify this, we add the following lines to the module `main` of the SMV input file:

```
SPEC
  AG (Switch.Change=1 -> Update=1)
```

The output of SMV is:

```
-- specification AG (Switch.Change = 1 -> Update = 1) is true
```

Alternatively, we can put the lines above into the module `Switch_module`, but since Change is a local variable there, we have to write `Change=1` instead of `Switch.Change=1`.

Now we try to prove that whenever an an Update : 1 event occurs, a Change : 1 event is generated:

```
SPEC
  AG (Update=1 -> Switch.Change=1)
```

The output of SMV is:

```
-- specification AG (Update = 1 -> Switch.Change = 1) is false
-- as demonstrated by the following execution sequence
state 1.1:
Request = stop
Update = 0
conditionsymbols = stop
Switch.Status = off
Switch.state = x_0
Switch.Change = 0

state 1.2:
Update = 1
```

The execution sequence shows that if we are at state x_0, the input condition is Request : *stop*, and the event Update : 1 occurs, no Change : 1 event is generated, which is a counterexample to our given specification. ◇

Chapter 8

Conclusion

Formal verification is a necessity for the design of safety critical systems. The acceptance and success of formal methods in system design however depends on the choice of the modeling formalism, which must orient itself by the field of application in which systems are verified.

In chemical engineering, systems are often specified using block diagrams to represent modules. Information is exchanged between these modules using signal flows. The behavior of the modules is often described informally, which makes the verification of such systems almost impossible.

A suitable formal framework in this field are *discrete condition/event systems* (DCESs), which are representable as block diagrams and exchange signals generated by finite-state transition systems. Though DCESs are widely used for modeling systems, compositional theories for their verification can rarely be found in the literature. Moreover, there are no composition operators given for the important class of *timed condition/event systems* (TCESs).

Our aim is to create a formal framework for the modular verification of DCESs and TCESs. Examining the existing operators *feedback* and *cascade* used in, e.g., [SK91], we find that these are difficult to handle and that they are unable to model certain interconnections between DCESs. Therefore, we define the *parallel interconnection* operator, which is able to express almost any given interconnection structure between DCESs. The avoidance of algebraic loops in the systems is done in two different ways, using a *unique solution property* and a *multiple solution property.* Furthermore, the parallel interconnection is extended to TCESs. Summarizing, we define a new composition operator for DCESs and TCESs which is general, powerful, and easy to use.

The example in Chapter 6 shows that we do not need to compose all modules of a given system to verify certain properties. This is an important observation, since a reduction of the number of modules leads to a significantly smaller state space in the composed part. Thus, this is a promising technique to cope with the state explosion problem. Further work lies in the exploitation of such techniques, and in giving general rules how and when to apply which technique to a given system.

Chapter 7 shows that the model checker SMV can be used to verify properties of DCESs. A first prototype of a compiler converting a set of DCESs into SMV code automatically is already implemented, and some properties of larger systems have been sucessfully verified using SMV. Further tests are in progress.

We also need to define a *temporal logic* for TCESs to be able to specify time dependent properties of TCESs in a formal verification framework.

Within the KONDISK project, we are currently working on further improvements of the automatic translation from TCES to timed automata [HLL+98]. A goal is to build a tool containing a graphical frontend for defining TCESs using block diagrams and transition graphs, and to specify properties in a temporal logic. The TCESs are transformed into timed automata, and the temporal logic formula is converted into the verification language of a model checker for timed automata, e.g., KRONOS. Then this model checker is used to verify the properties, and the results, like certain states or transitions falsifying a property, are presented in the graphical frontend. Using various state space reduction techniques, the tool should be able to verify systems of reasonable size.

Bibliography

[AD94] R. Alur and D. Dill. A theory of timed automata. In *Proceedings of the 17th ICALP: Automata Languages and Programming*, volume 126, pages 183–235. Springer-Verlag, 1994.

[BG90] A. Benveniste and P. Le Guernic. Hybrid dynamical systems theory and the SIGNAL language. *IEEE Transactions on Automatic Control*, 35(5):535–546, May 1990.

[BG92] G. Berry and G. Gonthier. The Esterel synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, November 1992.

[BLL+96] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi. UPPAAL—a tool suite for automatic verification of real-time systems. In *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 232–243. Springer-Verlag, 1996.

[CE81] E.M. Clarke and E.A. Emerson. Synthesis of synchronisation skeletons for branching time temporal logic. In D. Kozen, editor, *Workshop on Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*. Springer-Verlag, 1981.

[CPHP87] P. Caspi, D. Pilaud, N. Halbwachs, and J. Plaice. LUSTRE: a declarative language for programming synchronous systems. In *14th ACM Symposium on Principles of Programming Languages*, München, January 1987.

[GGB87] T. Gauthier, P. Le Guernic, and L. Besnard. Signal, a delarative language for synchronous programming of real-time systems. In *Proc. 3rd Conf. on Functional Programming Languages and Computer Architectures*, volume 274. Springer-Verlag, 1987.

[Hal93] N. Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, 1993.

[HdR86] J. Hooman and W.-P. de Roever. The quest goes on: towards compositional proof systems for CSP. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Current Trends in Concurrency*, volume 224 of *Lecture Notes in Computer Science*, pages 343–395. Springer-Verlag, 1986.

[HdR90] J. Hooman and W.P. de Roever. Design and verification in real-time distributed computing: An introduction to compositional methods. In *Protocol, Specification, Testing and Verification, IX*, pages 37–56. North-Holland, 1990.

[HH95] T.A. Henzinger and P.-H. Ho. HyTech: The Cornell hybrid technology tool. In *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 265–294. Springer-Verlag, 1995.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: The next generation. In *RTSS'95*, pages 56–65, 1995.

[HLL+98] R. Huuck, Y. Lakhnech, B. Lukoschus, L. Urbina, S. Engell, S. Kowalewski, and J. Preußig. Integrating timed condition/event systems and timed automata for the verification of hybrid systems. *Parallel and Distributed Computing Practices*, 1(2):45–60, June 1998.

[HLU+97a] R. Huuck, Y. Lakhnech, L. Urbina, S. Engell, S. Kowalewski, and J. Preußig. Combining a computer science and control theory approach to the verification of hybrid systems. In *5th Int. Workshop on Parallel and Distributed Real-Time Systems (WP-DRTS '97)*, Geneva, Switzerland, April 1997. Proceedings appear in IEEE Computer Society Press.

[HLU+97b] R. Huuck, Y. Lakhnech, L. Urbina, S. Engell, S. Kowalewski, and J. Preußig. Comparing timed c/e systems with timed automata. In *HART '97*, volume 1201 of *Lecture Notes in Computer Science*, pages 81–86, Grenoble, France, March 1997. Springer-Verlag.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[Huu98] R. Huuck. Transformation von Echtzeit-Bedingungs-/Ereignissystemen in Echtzeitautomaten: Ein Weg zur automatischen Verifikation. Diplomarbeit (Master's thesis), Christian-Albrechts-Universität zu Kiel, June 1998.

[Kow95] S. Kowalewski. *Modulare diskrete Anlagenmodellierung zum systematischen Steuerungsentwurf*. PhD thesis, Fachbereich Chemietechnik, Universität Dortmund, 1995.

[Lev95] N.G. Leveson. *Safeware: System Safety and Computers*. Addison-Wesley, 1995.

[Luk98] B. Lukoschus. Komposition von Bedingung-/Ereignissystemen und deren Echtzeitvarianten. Diplomarbeit (Master's thesis), Christian-Albrechts-Universität zu Kiel, August 1998.

[McM92] K.L. McMillan. *The SMV system*. Carnegie-Mellon University, February 1992. Draft version describing SMV revision 2.2.

[McM93] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publisher, 1993.

[Mil80] R. Milner. *A Calculus for Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice-Hall, 1989.

[ORS92] S. Owre, J. Rushby, and N. Shankar. PVS: A prototype verification system. In *11th Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer-Verlag, 1992.

[OY93] A. Olivero and S. Yovine. *KRONOS: A Tool for Verifying Real-Time Systems. User's Guide and Reference Manual*. VERIMAG, Grenoble, France, 1993.

[SK91] R.S. Sreenivas and B.H. Krogh. On condition/event systems with discrete state realizations. In *Discrete Event Dynamic Systems: Theory and Applications 1*, pages 209–236. Kluwer Academic Publishers, Boston, USA, 1991.