An Automata based Model for the Implementation of a TTCN Simulator

Jens Grabowski

IAM-96-006

February 1996

Abstract

This technical report provides a theoretical basis for the implementation of a TTCN test case simulator. In order to base the implementation on existing code, an automata based model has been chosen, i.e., in the model a TTCN test case shows an automaton like behavior. The procedure is the following: TTCN test cases are transformed into an internal structure, which afterwards can be interpreted by using the provided enabling conditions and execution rules. The model covers control and data flow aspects of normal and concurrent TTCN.

Acknowledgements

This work was supported by the KWF-Project No. 2555.1 *'Graphical Methods in the Test Process'*.

CR Categories and Subject Descriptors:

C.2.2 [Computer-Communication Networks]: Network Protocols; D.2.5 [Software Engineering:] Testing and Debugging; D.3.0 [Programming Languages]: General

General Terms:

TTCN, Validation, Simulation, Test Case Specification

1 Introduction

The aim of this technical report is to provide a theoretical basis for the implementation of a simulator for TTCN[1] test cases. This means a model which allows to interpret TTCN test cases has to be provided. In order to base our implementation on existing code and existing tools, in particular the SAMSTAG tool [2, 3, 4, 18], we choose an automata model. The model provides an interleaving semantics. Concurrency is described by indeterminism, i.e., the execution of a TTCN test case is described by all interleaved traces of concurrently executed events.

The notations which are used for the definition of the entire model are influenced by [1], [15] (i.e., Petri net theory), [18] (i.e., the SAMSTAG approach), [8], and [17].

However, the procedure is the following. At first the control flow aspects and then the data aspects of a TTCN test case are modeled. Finally, control flow and data aspects are combined in order to treat all aspects of a TTCN test case.

The control flow aspects are treated by transforming the dynamic part of a test case description into a directed graph with several sorts of edges. The nodes represent test events and the edges describe the influence on the control flow by the different sorts of test events. The graph is called *test case control structure*. Based on the test case control structure *control states* for TTCN test cases are defined. In such a control state several test events may be executable. They can be determined by using a *control enabling condition* and executed by applying an *execution rule*. The execution rule describes the transformation of the actual control state into a new control state when a control enabled test event is executed. A repeated use of enabling condition and execution rule allows the simulation of the control flow of a TTCN test case.

A similar approach is used for modeling the the influence of data on the behavior of a TTCN test case. We define a *test case data data structure* which relates each test event to a predicate and a set of expressions. Furthermore, it includes a set of typed *state variables*. The predicates are used to specify enabling conditions which depend on data values. The expressions define the transformation of *state variables* when a test event is executed. Subsequently, a tuple which includes a value for each state variables forms a *data state* of a TTCN test case. Based on the knowledge of data structure and data state we define a *data enabling condition* and an *execution rule* for data enabled test events.

The models for control flow and data aspects are the basis for the treatment of complete TTCN test cases. Enabling conditions and execution rules of both models are combined and form enabling condition and execution rule for complete TTCN test cases.

The proposed model is able to handle most TTCN constructs including parallel TTCN. Only recursive tree attachments can not be treated. For practical reasons we didn't model the influence of the language constructs TIMER START, TIMEOUT, CANCEL, and TERMINATE on the control flow, but it is shown how these constructs can be introduced.

This chapter is organized in the following way. Section 2 introduces some basic definitions. The control flow of a TTCN test case is modeled in Section 3. Section 4 describes the influence of data on the behavior of TTCN test cases. Control flow and data aspects

[1]The details concerning TTCN can be found in the language definition [11, 12, 13]. Tutorials on TTCN can be found in [2] and [16].

are combined in Section 5. In Section 6 we provide two examples which describe how the entire model works. A remark concerning complexity (Section 7), a comparison with other approaches (Section 8) and a summary (Section 9) conclude this chapter.

2 Basic definitions

Generally, we use the standard notations which need no special explanation. We only introduce some definitions for graphs, sequences, queues, and data handling which may facilitate the reading of the following sections.

2.1 Directed graphs

Definition 2.1 (Directed Graph) *A directed graph is defined by a pair (N, E), where:*

(a) $N \neq \emptyset$ is a finite set of nodes; and

(b) $E \subseteq N \times N$ is a finite set of edges.

Within a graph there may exist several different relations between its nodes, or the edges may be labeled with additional information. The definition of such graphs only is more extensive, but not much more complicated. As an example we present the definition of a directed graph with labeled edges.

Definition 2.2 (Directed graph labeled edges) *A directed graph with labeled edges is defined by a tuple $LG = (N, L, \delta)$, where:*

(a) $N \neq \emptyset$ is a finite set of nodes;

(b) $L \neq \emptyset$ is a finite set of labels; and

(c) $\delta \subseteq N \times L \times N$ is the set of labeled edges of LG.

The relation of two nodes within a directed graph can be described comfortably by using the terms *successor* and *predecessor*. We define them formally.

Definition 2.3 (Successor and predecessor) *Let $G = (N, E)$ be a directed graph and $a, b \in N$.*

1. *The set $a^\bullet$ is defined by $a^\bullet = \{b \mid (a, b) \in E\}$. An element of $a^\bullet$ is called a successor of a.*

2. *The set ${}^\bullet a$ is defined by ${}^\bullet a = \{b \mid (b, a) \in E\}$. An element of ${}^\bullet a$ is called a predecessor of a.*

For graphs with more than one relation the sets $a^\bullet$, and ${}^\bullet a$ have to be related to the corresponding relation. We distinguish the different sets by annotating them with the relation name, e.g. $a^\bullet_E$ and ${}^\bullet a_E$ where E denotes a relation.

2.2 Sequences

In the following we are often working with sequences. For example, a trace is a sequence of events or the contents of a message queue is a sequence of messages.

Definition 2.4 (Sets of sequences) *Let A be an arbitrary set, then we define the following four sets:*

1. *A^* are the finite sequences over A,*
2. *A^ω are the infinite sequences over A, and*
3. *$A^\infty = A^* \cup A^\omega$ are the finite and infinite sequences.*

Definition 2.5 (Operations on sequences) *Let $S \subseteq A^\infty$, and $t, u, v \in A^\infty$, and $a, b, c, d, a_0, \ldots, a_n \in A$.*

1. *$\bot$ is the empty sequence.*
2. *$t \cdot u$ denotes the concatenation of t and u. In cases where the meaning is unambiguous this notation may abbreviated by omitting the dot, e.g., $t \cdot u$ may be abbreviated by tu.*
3. *$\langle a_0, \ldots, a_n \rangle$ is the finite sequence consisting of the elements $a_0, \ldots, a_n$.*
4. *t_k denotes the k-th element of the sequence t.*
5. *$t^{(k)}$ denotes the sequence consisting of the first k elements of t.*
6. *$t \sqsubset u$ "t is a strict prefix of u" holds, iff $\exists v \neq \bot : t \cdot v = u$.*
7. *$t \sqsubseteq u$ "t is a prefix of u" holds, iff $\exists v : t \cdot v = u$.*
8. *$\#t$ denotes the length of t (Note, if t is infinite then $\#t = \infty$).*
9. *$first(t)$ denotes the first element of t.*
10. *$rest(t)$ denotes the rest of t.*
11. *$last(t)$ denotes the last element of t.*
12. *$a \textcircled{c} t$ denotes the filtered trace of t, i.e., the trace which contains only the element a, e.g., $a \textcircled{c} \langle a, b, a, c \rangle = \langle a, a \rangle$. As a generalization of this* filter *operation, the first operand may also be a set.*

2.3 Queues

The test components in *parallel TTCN* communicate via infinite FIFO queues. We introduce some basic notations to handle queues.

Definition 2.6 (Queue and the state of a queue) *Let X be a set.*

1. *A queue q is a container for a sequence $w \in X^*$. The sequence can only be altered and accessed only by applying the operations enqueue, dequeue, and next (cf. Definition 2.7).*

2. *The actual state s of the queue q denotes the actual stored sequence in the queue.*

To handle the state of a queue we define the function qstate. Let Q_X be a set of queues over the alphabet X.

3. *$qstate : Q_X \rightarrow X^*$ is a function which returns the actual state of a queue.*

Definition 2.7 (enqueue, dequeue, and next) *Let q be a queue which may store sequences over the set X, let $x \in X$ and $w \in X^* \backslash \perp$.*

1. *'enqueue' is an operation which alters the state of a queue by appending an element $x \in X$. For example, let w be $qstate(q)$ then $enqueue(q, x)$ alters the state of q to $w \cdot x$.*

2. *'dequeue' is an operation which alters the state of a queue by removing the first element. For example, let $x \cdot w$ be $qstate(q)$ then $dequeue(q)$ alters the state of q to w. $dequeue(q)$ does not do anything if it is applied to an empty queue.*

3. *'next' is an operation which returns the first element of the actual queue state. The state is not altered. For example, let $x \cdot w$ be $qstate(q)$ then $next(q)$ returns the element x. next returns $\perp$ if it is applied to an empty queue.*

2.4 Data handling

The following definitions do not claim to define the entire meaning of TTCN data types or ASN.1. They are general in order to describe the influence of data values on the control flow of TTCN test cases. We assume some intuitive knowledge concerning data types, typed variables, expressions, functions, and predicates.

Definition 2.8 (Data type, variable, assignment, function, predicate)

1. *A data type defines a (possibly infinite) set of data values, e.g., the data type $DAYS$ may characterize the elements of the set $\{Mon, Tue, Wed, Thu, Fri, Sat, Sun\}$.*

2. *A typed variable is a container for a value of a specific type. A typed variable has a name, i.e., the identifier of the container, and a value, i.e., the actual contents of the container. The value is referred to by using the variable name, e.g., let the variable y have the value 3 then the expression $x + 2$ denotes the integer value 5.*

3. *A function is an injective relation from one set of values, called domain of f or $dom(f)$, into another set of values, called range of f or $ran(f)$. Often this is stated by the statement: $f : dom(f) \rightarrow ran(f)$, e.g., the integer addition $+$ is characterized by: $+ : integer \rightarrow integer$.*

4. *A predicate P is a function with the range Boolean, i.e., the values $true$ and $false$. P is characterized by: $P : dom(P) \rightarrow \{true, false\}$. We say that a predicate is valid if it evaluates to $true$.*

We intend to simulate TTCN test cases without knowledge about a corresponding IUT. Sometimes the value of control variables of a test case may depend on parameter values of signals received from the IUT. In such a case we are not able to determine a variable value precisely. In order to indicate that a variable value is not known we introduce the special value *unknown* ($\propto$).

Definition 2.9 (The special value 'unknown' and its handling)

1. *The special value unknown $\propto$ denotes that a value value can not be determined exactly or is not known. We assume that the value $\propto$ can be used for all possible types of values.*

2. *If the evaluation of a function f depends on an unknown value, f evaluates to $\propto$. For example, $3+ \propto = \propto$, but, $0* \propto +1 = 1$.*

3. *If the evaluation of a predicate P depends on an unknown value, P evaluates to $true$. For example, $false \vee \propto = true$, but $false \wedge \propto = false$*

The special handling of unknown values in predicates may need some explanation. Within the simulation process of TTCN test cases we will use predicates to express enabling conditions for test events which depend on the values of test case control variables[2]. The incomplete knowledge about the corresponding IUT should not lead to the exclusion of simulation paths. Subsequently, all predicate values which depend on variables with unknown values should evaluate to *true*.

3 Modeling the control flow of TTCN test cases

In this section we present a model for the handling of the control flow aspects of a TTCN test case. We start with the definition of a test case control structure, followed by the definitions of a test case control state, a control enabling condition and an execution rule for control enabled events. Based on these definitions traces, reachability sets and global state graphs for the control flow of a TTCN test case are introduced. The section ends with some remarks concerning character and restrictions of the presented model.

[2]For example, a control variable may be the counter in a counter loop.

3.1 The control structure of a TTCN test case

In this section we introduce a directed graph, called *test case control structure*, with four sorts of edges. This graph is able to describe the control flow aspects of a TTCN test case. The transformation of a TTCN test case description into the graph is given informally by providing some examples which describe the meaning of nodes and edges of the graph.

Definition 3.1 (Test case control structure) *A test case control structure of a test case TC is defined by a directed graph with four sorts of edges. Formally it is given by the tuple $TCCS = (TE, t0, nte, attach, create, cm)$, where:*

(a) $TE \neq \emptyset$ *is a finite set of test events.*

(b) $t0 \notin TE$ *is the start node of the test case.*

(c) $nte \subseteq TE \times TE$ *is the next-test-event relation.*

(d) $attach \subseteq TE \times TE$ *is the attach relation. The attach relation is used to model the attachment of TTCN behavior trees.*

(e) $create \subseteq TE \times TE$ *is the create relation. It is used to model the creation of parallel test components by a main test component.*

(f) $cm \subseteq TE \times TE$ *is the coordination-message relation. The cm relation is used to model the exchange of coordination messages between parallel test components.*

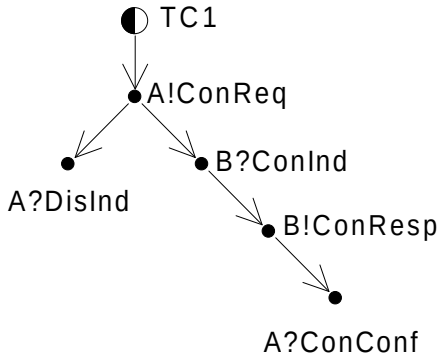
In the following we present some examples which may be helpful to describe the meaning of the test case control structure. We show some behavior descriptions of TTCN test cases and their representation in our graph structure. The formal mapping of a TTCN test case into a test case control structure will be defined later.

Example 3.1 (The meaning of test events and start node) Test events are the actions which have to be executed by the test devices in order to run the test case. In TTCN test events are specified within the *Behaviour Description* column of the dynamic behavior tables, e.g. Figures 1 (a), 2 (a), or 3 (a). Generally, each test event in a TTCN test case description is related to one test event in the corresponding test case control structure. There are only a few exceptions of this rule. The tree attachment, i.e., the Attach construct, is modeled by two nodes and the UNTIL statement within the Repeat construct is modeled by three nodes. The details will be explained later.

For technical purposes we introduce top nodes which are annotated with the name of the test case or test step description. The top nodes have no corresponding test event in the TTCN behavior description. These nodes can be considered to be the anchor nodes of TTCN trees. Later on they are used to enable the first event of attached trees and parallel test components.

The start node is one of these extra nodes. It is the anchor of the whole test case. It is used to identify the first test event of the test case. For example, the top node of the graph in Figure 1 (b) is given by the node *TC1* and in Figure 3 (b) the start node is called TC3.

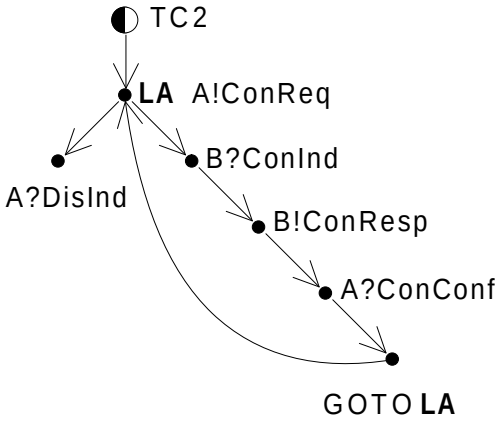
Test Case Dynamic Behaviour		
Test Case Name: TC1		
Nr	Label	Behaviour Description
1		A!ConReq
2		B?ConInd
3		B!ConResp
4		A?ConConf
5		A?DisInd



(a) TTCN behavior tree (b) Test case control structure of (a)

Figure 1: Simple TTCN test case and the corresponding test case control structure

Test Case Dynamic Behaviour		
Test Case Name: TC2		
Nr	Label	Behaviour Description
1	LA	A!ConReq
2		B?ConInd
3		B!ConResp
4		A?ConConf
5		GOTO LA
6		A?DisInd



(a) TTCN behavior tree (b) Test case control structure of (a)

Figure 2: TTCN test case with loops and the corresponding test case control structure

Example 3.2 (The meaning of the *nte* relation) Figure 1 shows the behavior tree of a simple TTCN test case (a) and the corresponding test case control structure (b). The graph in (b) is a simple tree. The tree structure is determined by the TTCN test events and the possible sequences of test events. These sequences are described by the *nte* relation. The graph in (b) includes no *attach*, *create* and *cm* edges because the TTCN description includes no parallel test components and no tree attachments.

The use of GOTO and REPEAT statements within a TTCN test case may lead to loops within the corresponding test case control structure. Figure 2 provides an example.

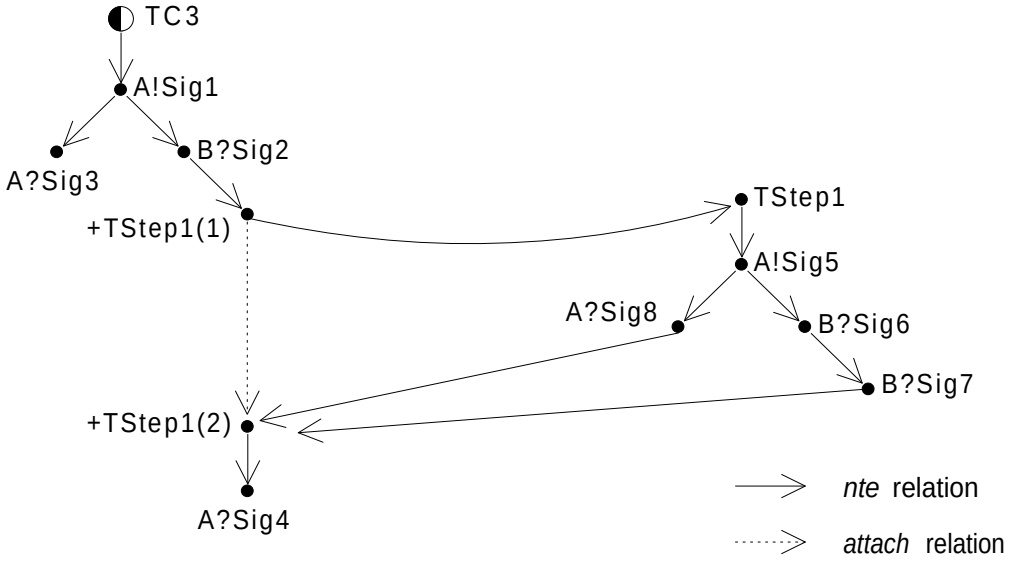
Example 3.3 (The meaning of the *attach* relation) The TTCN test case description in Figure 3 (a) consists of two behavior trees. The main tree *TC3* and the local tree *TStep1*. *TStep1* is attached to *TC3* by means of the attach statement *+TStep1*.

For technical purposes in the corresponding test case control structure an attach statement is represented by two nodes which are related by the *attach* relation, i.e. *+TStep1*

Test Case Dynamic Behaviour		
Test Case Name: TC3		
Nr	Label	Behaviour Description
1		A!Sig1
2		B?Sig2
3		+TStep1
4		A!Sig4
5		A?Sig3

Test Step Dynamic Behaviour		
Test Step Name: TStep1		
Nr	Label	Behaviour Description
1		A!Sig5
2		B?Sig6
3		B?Sig7
4		A?Sig8

(a) Two TTCN behavior trees



(b) Test case control structure of (a)

Figure 3: Using the *attach* relation for modeling the TTCN tree attachment

in (a) is represented by *+TStep1(1)* and *+TStep1(2)* in (b). The *nte* edges describe the transfer of the control flow from the main tree to the test step and back to the main tree.

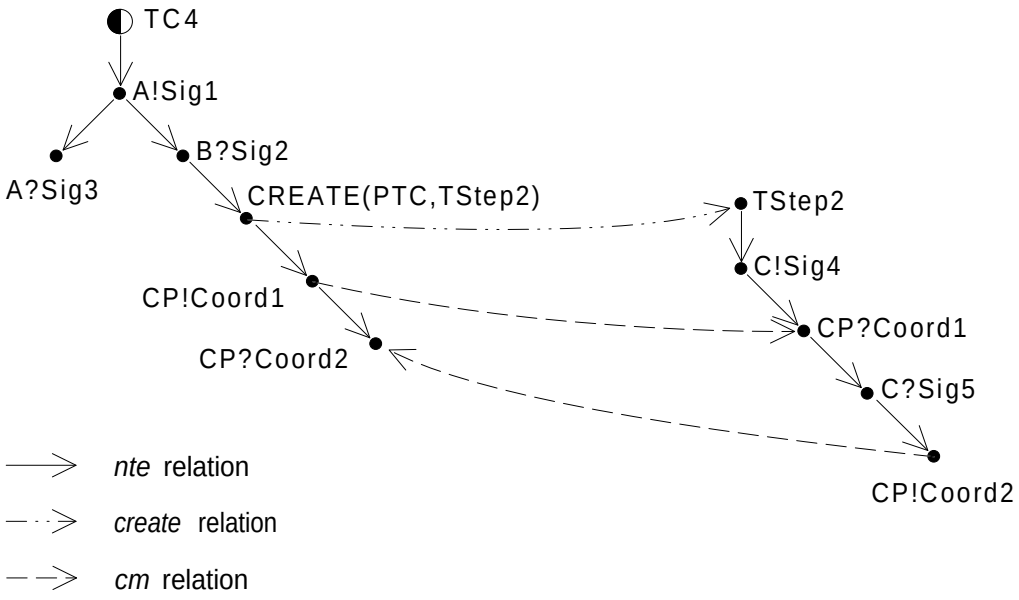
A test step may be called from different places of the main tree. As a consequence each final test event of the attached test step has *nte* edges to different statements of the main tree. Only from the *nte* relation it is not clear to which point of the main tree the control should return. Within our state definition we will use the *attach* relation to remember the correct point of return. It is obvious that such a mechanism is not able to handle recursive tree attachments.

Example 3.4 (The meaning of *create* and *cm* relation) The TTCN test case in Figure 4 (a) comprises a main test component *TC4* which creates the parallel test component *PTC* by using the test step definition *TStep2*. The test components synchronize themselves by exchanging the coordination messages *Coord1* and *Coord2*.

Test Case Dynamic Behaviour		
Test Case Name: TC4		
Comment: This is the MTC		
Nr	Label	Behaviour Description
1		A!Sig1
2		B?Sig2
3		CREATE(PTC,TStep2)
4		CP!Coord1
5		CP?Coord2
6		A?Sig3

Test Step Dynamic Behaviour		
Test Step Name: TStep2		
Comment: This is the PTC		
Nr	Label	Behaviour Description
1		C!Sig4
2		CP?Coord1
3		C?Sig5
4		CP!Coord2

(a) Main test component MTC and a parallel test component PTC



(b) Test case control structure of (a)

Figure 4: Modeling the creation and synchronization of parallel test components

As shown in Figure 4 (b) the *create* and *cm* relation are used to describe the ordering of test events in different test components. For example, the first test event of *TStep2* can only be performed after the execution of the corresponding CREATE event by *TC4*. Analogously, the coordination message *Coord2* can only be received by *TC4* after its sending by *TStep2*.

A test case may consist of several test components. The test events within the corresponding test control structure are related to these test components. For simplification we assume that test steps are local to test components, i.e. within a test case a test step only can be attached by one test component. Furthermore, we assume that different create events refer to different test step definitions, i.e. $\forall te \in TE : \#({}^{\bullet}te_{create} \cup te_{create}^{\bullet}) \leq 1$,

and that a test step which can be instantiated as test component is not used for tree attachment, i.e. $\forall te \in TE : te^{\bullet}_{create} \cap te^{\bullet}_{nte} = \emptyset$.

3.2 The control state of a TTCN test case

It is our aim to simulate the control structure of a TTCN test case. This is done by defining the control state of a TTCN test case and by defining the transition from one state to another when a test event is executed. Before we can define a control state we need some definitions to handle the queues at the coordination points of parallel test components.

Parallel test components synchronize themselves and communicate by exchanging coordination messages at *coordination points*. A coordination point can be seen as a gate between two test components. Coordination points have to be declared in the declarations part of the TTCN test suite. According to the TTCN semantics a coordination point is modeled by two infinite FIFO queues, i.e. one FIFO queue for each direction.

During the simulation of a TTCN test case the execution order of test events is influenced by these FIFO queues. In order to preserve the correct order of test events we also have to model a queue mechanism for test events. Although our queues are used to buffer test events and not to buffer coordination messages we also relate them to the coordination points of a test case.

Definition 3.2 (Test components, coordination points, and queues) *Let TC be a TTCN test case. Then there exist the sets CP, $TECO$, and QU, where:*

1. *$TECO$ is the set of test components of TC.*
2. *CP denotes the set of all coordination points of TC.*
3. *QU denotes the set of all queues which correspond to the coordination points of TC.*
4. *$QS_{QU} = \{qs_q \mid \forall q \in QU : qs_q = qstate(q)\}$ is the set of all queue states of QU.*

Each queue is determined by the coordination point, the sending test component, and the receiving test component. To identify queues and their states uniquely we annotate them with the corresponding names. For example $q_{(cp,s,r)} \in QU$, $s_{(cp,s,r)} = qstate(q_{(cp,s,r)})$ where $cp \in QU$ and $s, r \in TECO$.

A test event may manipulate a queue of QU by sending and receiving coordination messages. Based on the kind of an event and the involved coordination point the used queue can be determined statically. Therefore we define the *tequeue* function.

Definition 3.3 (*tequeue* function) *Let $TCCS = (TE, t0, nte, attach, create, cm)$ be the test case control structure of the TTCN test case TC, and QU the set of all queues which correspond to the coordination points of TC. Then*

$tequeue : \{te \mid te \in TE \wedge {}^{\bullet}te_{cm} \cup te^{\bullet}_{cm} \neq \emptyset\} \rightarrow QU$

is a function which returns the queue which is manipulated by a specific test event.

Now we are able to define the control state of a TTCN test case.

Definition 3.4 (Test case control state) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure and QU the set of FIFO queues of TC. A control state of the test case TC is defined by a tuple $CS = (S_{nte}, S_{attach}, S_{create}, S_{cm})$, where:*

(a) $S_{nte} \subseteq TE$ is a set of test events.

(b) $S_{attach} \subseteq TE$ is a set of test events.

(c) $S_{create} \subseteq TE$ is a set of test events.

(d) $S_{cm} = QS_{QU}$ describe the actual state of all queues $q \in QU$.

Explanation of Definition 3.4. A test case control state comprises the four sets S_{nte}, S_{attach}, S_{create}, and S_{cm}. S_{nte} is used to remember the last executed test event of each test component. In the case of tree attachment a test step or local tree might be attached at different places of the behavior tree. S_{attach} is used to remember the correct point return after the attached behavior tree has been executed. S_{create} is used to initialize the execution of a test component after its creation. S_{cm} is used to preserve the order of send and receive events of coordination messages in different test components.

Later on we will generate traces and global state graphs by simulating the test case control structure. For checking the finiteness of the explored traces and state graphs it is necessary to be able to compare states. Therefore we define *equality* and $<$ for test case control states.

Definition 3.5 (Equality and order relation for control states) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure, QU the set of all queues which correspond to the coordination points of TC, and let $S = (S_{nte}, S_{attach}, S_{create}, S_{cm})$ and $S' = (S'_{nte}, S'_{attach}, S'_{create}, S'_{cm})$ be control states of TC. Let $qs_q \in S_{cm}$ and $qs'_q \in S'_{cm}$ be queue states of $q \in QU$.*

1. *$S = S'$ iff $S_{nte} = S'_{nte} \wedge S_{attach} = S'_{attach} \wedge S_{create} = S'_{create} \wedge S_{cm} = S'_{cm}$*

2. *$S < S'$ iff the following four conditions hold:*

   (a) $S_{nte} = S'_{nte}$

   (b) $S_{attach} = S'_{attach}$

   (c) $S_{create} = S'_{create}$

   (d) $\forall q \in QU \ : \ qs_q \sqsubset qs'_q$

3.3 Control enabling of test events

Several test events may be executable in a control state. They can be determined by using the following enabling condition.

Definition 3.6 (Control enabling of a test event) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure, $CS = (S_{nte}, S_{attach}, S_{create}, S_{cm})$ a control state of TC, and QU the set of FIFO queues of TC. A test event $te \in TE$ is control enabled in CS, written $CS \xrightarrow{te}_c$, if the following four conditions hold:*

(a) ${}^\bullet te_{nte} = \emptyset \vee \#({}^\bullet te_{nte} \cap S_{nte}) = 1$

(b) ${}^\bullet te_{attach} = \emptyset \vee \#({}^\bullet te_{attach} \cap S_{attach}) = 1$

(c) ${}^\bullet te_{create} = \emptyset \vee \#({}^\bullet te_{create} \cap S_{create}) = 1$

(d) ${}^\bullet te_{cm} = \emptyset \vee next(qstate(tequeue(te))) \in {}^\bullet te_{cm}$ $(qstate(tequeue(te)) \in S_{cm})$

The index c in $CS \xrightarrow{te}_c$ should indicate that the enabling condition is based on the information in a test control structure. Later on we will define an enabling condition which is based on data information. We will use the same notation, but change the index to d.

Explanation of Definition 3.6. A test event is control enabled, if it can be executed. We explain this by means of the test case control structure shown in Figure 5. The corresponding test case comprises two test components MTC and PTC. We assume there exist one coordination point cp with the corresponding queues $q_{(cp,MTC,PTC)}$ and $q_{(cp,PTC,MTC)}$. In the following queue states different from $\perp$ are annotated in order to relate them to the corresponding queues, e.g. $h{\cdot}i_{(cp,MTC,PTC)}$ means that $qstate(q_{(cp,MTC,PTC)}) = h{\cdot}i$.

- In the state $S^1 = (\{MTC\}, \emptyset, \emptyset, \{\perp, \perp\})$ only the test event a is control enabled, because ${}^\bullet a_{nte} = \{MTC\}$, ${}^\bullet a_{attach} = \emptyset$, ${}^\bullet a_{create} = \emptyset$, and $a \notin dom(tequeue)$. Event a is the first test event of TC. The example shows how a is control enabled by using the start node MTC. It should be noted that it is only possible to enable test events, the start node is not in the set of test events (cf. Definition 3.1 (b)) and therefore can never be control enabled.

- In the state $S^2 = (\{c\}, \{b\}, \emptyset, \{\perp, \perp\})$ the events d and e are control enabled. They require that $c \in S^2_{nte}$ has been executed before. ${}^\bullet d_{attach}$, ${}^\bullet e_{attach}$, ${}^\bullet d_{create}$, and ${}^\bullet e_{create}$ are for both the empty set and $\{d, e\} \not\subseteq dom(tequeue)$.

- In the state $S^3 = (\{g\}, \emptyset, \{g\}, \{\perp, \perp\})$ the two test events h and PTC are control enabled. The event h requires only the execution of g. Test event PTC is the top node of the parallel test component. Its control enabling requires the execution of the create event g. The creation of PTC by g is indicated by $S^3_{create} = \{g\}$.

- In the state $S^4 = (\{i, m\}, \emptyset, \emptyset, \{h{\cdot}i_{(cp,MTC,PTC)}, \perp\})$ the test events n and p are control enabled. Event n requires the execution of m, i.e. $m \in S^4_{nte}$, and that the next element to be dequeued of $q_{(cp,MTC,PTC)}$ $(= next(tequeue(n)))$ is element of ${}^\bullet n_{cm}$. This is valid because $next(h{\cdot}i_{(cp,MTC,PTC)}) = h$.

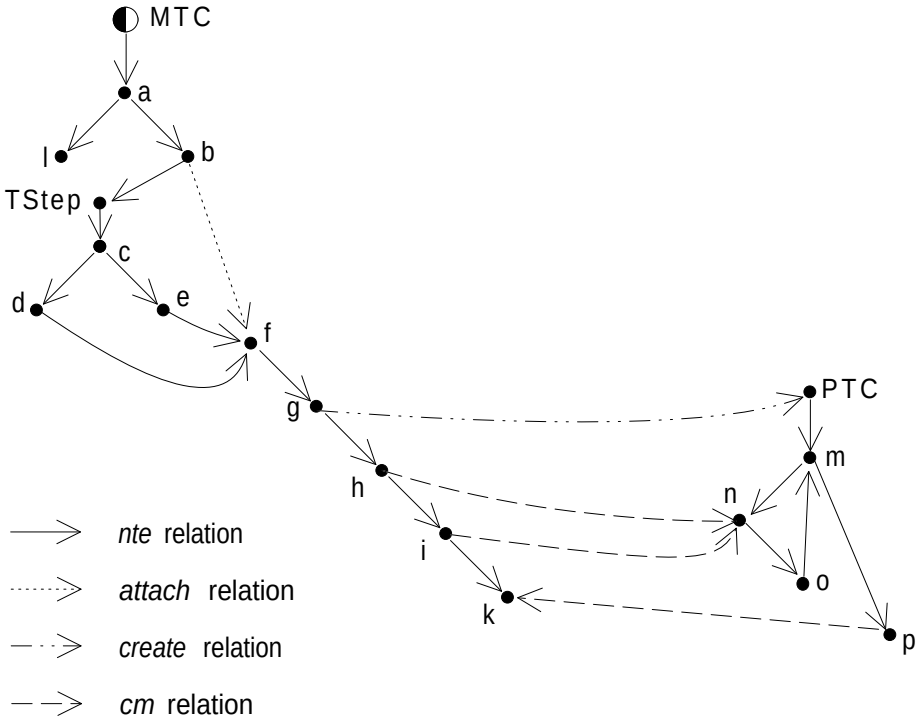


Figure 5: A test case control structure

This example also shows the necessity of using a set of queues of test events instead of a set of test events for modeling S_{cm}. From a set notation like $\{h, i\}_{(cp,MTC,PTC)}$ we cannot distinguish whether h or i has been performed first. For simulation purposes the order of send and receive events of coordination messages may very well be relevant.

The control enabling of p requires the execution of m. The state components S^4_{attach}, S^4_{create}, and S^4_{cm} are not relevant.

3.4 The execution of control enabled test events

After defining the control state of a test case and the control enabling of test events we define the execution of control enabled test events. According to certain rules which are defined afterwards during execution the current control state is altered.

Definition 3.7 (Execution of a control enabled test event) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure, QU the set of FIFO queues of TC, $CS = (S_{nte}, S_{attach}, S_{create}, S_{cm})$ a control state of TC, and $te \in TE$ be control enabled in CS. Furthermore, if $te \in dom(tequeue)$ then let $q_{te} \in QU$ be the queue which is manipulated by te (i.e. $q_{te} = tequeue(te)$).*

The execution of te changes the control state CS into the new control state CS'. We denote this fact by $CS \xrightarrow{te}_c CS'$. The new control state $CS' = (S'_{nte}, S'_{attach}, S'_{create}, S'_{cm})$ is defined by:

$$(a)\ S'_{nte} = S_{nte} \backslash {}^\bullet te_{nte} \cup \begin{cases} \emptyset & : \ te^\bullet_{nte} = \emptyset \\ \{te\} & : \ te^\bullet_{nte} \neq \emptyset \end{cases}$$

$$(b)\ S'_{attach} = S_{attach} \backslash {}^\bullet te_{attach} \cup \begin{cases} \emptyset & : \ te^\bullet_{attach} = \emptyset \\ \{te\} & : \ te^\bullet_{attach} \neq \emptyset \end{cases}$$

$$(c)\ S'_{create} = S_{create} \backslash {}^\bullet te_{create} \cup \begin{cases} \emptyset & : \ te^\bullet_{create} = \emptyset \\ \{te\} & : \ te^\bullet_{create} \neq \emptyset \end{cases}$$

$$(d)\ S'_{cm} = \begin{cases} S_{cm} & : \ te \notin dom(tequeue) \\ perform\ dequeue(q_{te})\ in\ S_{cm} & : \ te \in dom(tequeue) \wedge {}^\bullet te_{cm} \neq \emptyset \\ perform\ enqueue(q_{te}, te)\ in\ S_{cm} & : \ te \in dom(tequeue) \wedge te^\bullet_{cm} \neq \emptyset \end{cases}$$

Explanation of Definition 3.7. We explain the meaning of the execution of a control enabled test event by means of the example in Figure 5 and by using the states S^1, S^2, S^3, and S^4 which we already examined in the explanation of Definition 3.6 (cf. Page 12).

- In state $S^1 = (\{MTC\}, \emptyset, \emptyset, \{\bot, \bot\})$ test event a is control enabled. When a is executed S^1 changes to $S^{11} = (\{a\}, \emptyset, \emptyset, \{\bot, \bot\})$. S^{11} is calculated by

  $S^{11}_{nte} = \{MTC\} \backslash \{MTC\} \cup \{a\} = \{a\}$ because ${}^\bullet a_{nte} = \{MTC\}$ and $a^\bullet_{nte} = \{l, b\}$.
  $S^{11}_{attach} = \emptyset \backslash \emptyset \cup \emptyset = \emptyset$ because $a^\bullet_{attach} = \emptyset$.
  $S^{11}_{create} = \emptyset \backslash \emptyset \cup \emptyset = \emptyset$ because $a^\bullet_{create} = \emptyset$.
  $S^{11}_{cm} = S^1_{cm} = \{\bot, \bot\}$ because $a \notin dom(tequeue)$.

  In the new state S^{11} the test event a is not control enabled anymore, but the events b and l are control enabled newly.

- In the state $S^2 = (\{c\}, \{b\}, \{\bot, \bot\})$ the two events d and e are control enabled. Event d executes to S^{21} and event e executes to S^{22}. $S^{21} = (\{d\}, \{b\}, \emptyset, \{\bot, \bot\})$ is computed by

  $S^{21}_{nte} = \{c\} \backslash \{c\} \cup \{d\} = \{d\}$,
  $S^{21}_{attach} = \{b\} \backslash \emptyset \cup \emptyset = \{b\}$,
  $S^{21}_{create} = \emptyset \backslash \emptyset \cup \emptyset = \emptyset$, and
  $S^{21}_{cm} = S^2_{cm} = \{\bot, \bot\}$ $(d \notin dom(tequeue))$.

  $S^{22} = (\{e\}, \{b\}, \emptyset, \emptyset, \{\bot, \bot\})$ is computed analogously. It should be noted that after the execution of d or e the other event is not control enabled anymore, i.e., the events are control enabled alternatively and not independently.

- In the state $S^3 = (\{g\}, \emptyset, \{g\}, \{\bot, \bot\})$ the two test events h and PTC are control enabled. Event h executes to $S^{31} = (\{h\}, \emptyset, \{g\}, \{h_{(cp,MTC,PTC)}, \bot\})$ and PTC executes to $S^{32} = (\{g, PTC\}, \emptyset, \emptyset, \{\bot, \bot\})$. The computation of S^{31} shows how the queues are manipulated. S^{31} is calculated by

$S^{31}_{nte} = \{g\}\backslash\{g\} \cup \{h\} = \{h\}$.

$S^{31}_{attach} = \emptyset\backslash\emptyset \cup \emptyset = \emptyset$.

$S^{31}_{create} = \{g\}\backslash\emptyset \cup \emptyset = \{g\}$.

$S^{31}_{cm} = \{h_{(cp,MTC,PTC)}, \perp\}$ because we perform an $enqueue(h, tequeue(h))$ operation on an empty queue in S^3_{cm}.

S^{32} is computed by

$S^{32}_{nte} = \{g\}\backslash\emptyset \cup \{PTC\} = \{g, PTC\}$,

$S^{32}_{attach} = \emptyset\backslash\emptyset \cup \emptyset = \emptyset$,

$S^{32}_{create} = \{g\}\backslash\{g\} \cup \emptyset = \emptyset$, and

$S^{32}_{cm} = S^3_{cm} = \{\perp, \perp\}$ ($l \notin dom(tequeue)$).

The calculation of S^{32} shows how the *create* relation is used to enable the test event PTC of the created test component. It should be noted that h and PTC are enabled independently. After the execution of h event PTC is still control enabled and vice versa, i.e. $S^{31} \xrightarrow{PTC}_c$ and $S^{32} \xrightarrow{h}_c$.

- In the state $S^4 = (\{i, m\}, \emptyset, \emptyset, \{h{\cdot}i_{(cp,MTC,PTC)}, \perp\})$ the events n and p are control enabled. Event n executes to $S^{41} = (\{k, n\}, \emptyset, \emptyset, \{i_{(cp,MTC,PTC)}, \perp\})$ and event p executes to $S^{42} = (\{i, p\}, \emptyset, \emptyset, \{h{\cdot}i_{(cp,MTC,PTC)}, \{p_{(cp,PTC,MTC)}\})$. S^{41} is calculated by

  $S^{41}_{nte} = \{i, m\}\backslash\{m\} \cup \{n\}$

  $S^{41}_{attach} = \emptyset\backslash\emptyset \cup \emptyset = \emptyset$

  $S^{41}_{create} = \emptyset\backslash\emptyset \cup \emptyset = \emptyset$

  $S^{41}_{cm} = \{i_{(cp,MTC,PTC)}, \perp\}$ because we perform a $dequeue(tequeue(h))$ operation on a queue with the actual state $h{\cdot}i_{(cp,MTC,PTC)}$.

  S^{42} is computed analogously.

3.5 Control flow based definitions of traces, reachability sets, and global state graphs

Based on the definitions of test case control structure, control state, control enabling of test events, and execution of control enabled test events it is possible to define traces, reachability sets, and global state graphs for the control flow of TTCN test cases.

Definition 3.8 (Initial control state of a TTCN test case) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure, and QU be the set of all queues which correspond to the coordination points of TC. A control state $S^0 = (S^0_{nte}, S^0_{attach}, S^0_{create}, S^0_{cm})$ is called the initial control state of TC if the following holds:*

(a) $S^0_{nte} = \{t0\}$,

(b) $S^0_{attach} = \emptyset$,

(c) $S^0_{create} = \emptyset$, *and*

(d) $S^0_{cm} = \{qs_q \mid \forall q \in QU : qs_q = qstate(q) \wedge qs_q = \perp\}$

Definition 3.9 (Traces and reachability set) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure, $w \in TE^*$ a finite word over TE, M_{CS} be the set of all control state of TC, S^0 the initial control state of TC, and S, S', S_1, S_2, ... , S_n control states of TC.*

1. *The notation $S \xrightarrow{w}_c S'$ is defined by*

   (a) $S \xrightarrow{w}_c S$ *if* $w = \perp$

   (b) $S \xrightarrow{w}_c S'$ *if* $w = t_1 \cdot t_2 \cdot ... \cdot t_n$ $(t_i \in TE)$ *and* $S \xrightarrow{t_1}_c S_1 \xrightarrow{t_2}_c ... \xrightarrow{t_n}_c S_n = S'$

2. *$R(TC) = \{S \mid \exists w \in TE^* : S^0 \xrightarrow{w}_c S \wedge S \in M_{CS}\}$ is the reachability set of TC, i.e. the set of control states which can be reached from the initial state S^0 when TC is executed.*

3. *$TR(TC) = \{w \mid \exists S \in M_{CS} : S^0 \xrightarrow{w}_c S \wedge w \in TE^*\}$ is the set of traces of TC.*

Definition 3.10 (Global state graph) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure, S^0 the initial control state, and $R(TC)$ be the reachability set of TC. The global state graph of TC is defined by $Er(TC) = (N, E, S^0)$, where*

(a) $N = R(N)$ *is the set of nodes,*

(b) $E = \{(S, t, S') \mid S, S' \in R(N) \wedge S \xrightarrow{t}_c S'\}$ *is the set of labeled edges, and*

(c) S^0 *is the start node of the graph.*

According to the TTCN semantics a test run ends in a final state in which a test verdict is assigned. We define the final states of a TTCN test case formally.

Definition 3.11 (Final control states of a TTCN test case) *Let TC be a TTCN test case, $TCCS = (TE, t0, nte, attach, create, cm)$ the corresponding test case control structure, and $R(TC)$ be the reachability set of TC. The final states of TC are defined by the set $FS(TC) = \{S \mid \nexists\, te \in TE : S \xrightarrow{te}_c\}$.*

Example 3.5 (Reachability set, global state graph, final control states) Some examples may clarify the meaning of the definitions.

Nr.	Value	Nr.	Value
S0	$(\{MTC\},\emptyset,\emptyset,\{\bot,\bot\})$	S18	$(\{h\},\emptyset,\emptyset,\{h_{(cp,MTC,PTC)},p_{(cp,PTC,MTC)}\})$
S1	$(\{a\},\emptyset,\emptyset,\{\bot,\bot\})$	S19	$(\{h,n\},\emptyset,\emptyset,\{\bot,\bot\})$
S2	$(\emptyset,\emptyset,\emptyset,\{\bot,\bot\})$	S20	$(\{i,m\},\emptyset,\emptyset,\{h{\cdot}i_{(cp,MTC,PTC)},\bot\})$
S3	$(\{b\},\{b\},\emptyset,\{\bot,\bot\})$	S21	$(\{i\},\emptyset,\emptyset,\{h{\cdot}i_{(cp,MTC,PTC)},p_{(cp,PTC,MTC)}\})$
S4	$(\{TStep\},\{b\},\emptyset,\{\bot,\bot\})$	S22	$(\{h,o\},\emptyset,\emptyset,\{\bot,\bot\})$
S5	$(\{c\},\{b\},\emptyset,\{\bot,\bot\})$	S23	$(\{i,n\},\emptyset,\emptyset,\{i_{(cp,MTC,PTC)},\bot\})$
S6	$(\{d\},\{b\},\emptyset,\{\bot,\bot\})$	S24	$(\emptyset,\emptyset,\emptyset,\{h{\cdot}i_{(cp,MTC,PTC)},\bot\})$
S7	$(\{e\},\{b\},\emptyset,\{\bot,\bot\})$	S25	$(\{h,m\},\emptyset,\emptyset,\{\bot,\bot\})$
S8	$(\{f\},\emptyset,\emptyset,\{\bot,\bot\})$	S26	$(\{i,o\},\emptyset,\emptyset,\{i_{(cp,MTC,PTC)},\bot\})$
S9	$(\{g\},\emptyset,\{g\},\{\bot,\bot\})$	S27	$(\{h\},\emptyset,\emptyset,\{\bot,p_{(cp,PTC,MTC)}\})$
S10	$(\{PTC,g\},\emptyset,\emptyset,\{\bot,\bot\})$	S28	$(\{i,m\},\emptyset,\emptyset,\{i_{(cp,MTC,PTC)},\bot\})$
S11	$(\{h\},\emptyset,\{g\},\{h_{(cp,MTC,PTC)},\bot\})$	S29	$(\{i\},\emptyset,\emptyset,\{i_{(cp,MTC,PTC)},p_{(cp,PTC,MTC)}\})$
S12	$(\{PTCh\},\emptyset,\emptyset,\{h_{(cp,MTC,PTC)},\bot\})$	S30	$(\{i,n\},\emptyset,\emptyset,\{\bot,\bot\})$
S13	$(\{i\},\emptyset,\{g\},\{h{\cdot}i_{(cp,MTC,PTC)},\bot\})$	S31	$(\emptyset,\emptyset,\emptyset,\{i_{(cp,MTC,PTC)},\bot\})$
S14	$(\{g,m\},\emptyset,\emptyset,\{\bot,\bot\})$	S32	$(\{i,o\},\emptyset,\emptyset,\{\bot,\bot\})$
S15	$(\{g\},\emptyset,\emptyset,\{\bot,p_{(cp,PTC,MTC)}\})$	S33	$(\{i,m\},\emptyset,\emptyset,\{\bot,\bot\})$
S16	$(\{PTC,i\},\emptyset,\emptyset,\{h{\cdot}i_{(cp,MTC,PTC)},\bot\})$	S34	$(\{i\},\emptyset,\emptyset,\{\bot,p_{(cp,PTC,MTC)}\})$
S17	$(\{h,m\},\emptyset,\emptyset,\{h_{(cp,MTC,PTC)},\bot\})$		

Table 1: The reachability set of Example 3.5 (a)

(a) Figure 5 shows a test case control structure. We assume there is only one coordination point *cp* with the queues $q_{(cp,MTC,PTC)}$ and $q_{(cp,PTC,MTC)}$. The initial state is given by $S^0 = (\{MTC\},\emptyset,\emptyset,\{\bot,\bot\})$. By executing the control enabled test events continuously, we gain the reachability set which is shown in Table 1. The complete global state graph is shown in Figure 6. The initial state *S0* and final states *S2*, *S24*, and *S31* are emphazised.

(b) A global state graph may include loops. An example is presented in Figure 7. In (a) a test case control structure with a loop is given. The graph consists only of *nte* edges. Therefore within a test case control state $S^i = (S^i_{nte}, S^i_{attach}, S^i_{create}, S^i_{cm})$ only the *nte* part S^i_{nte} is relevant. The other parts are always the empty set. By using the initial control state $S^0 = (\{MTC\},\emptyset,\emptyset,\emptyset)$ as start node we gain the global state graph shown in (b). The initial state is given by $(\{MTC\},\{\},\{\},\{\},\{\})$. The final state is defined by $(\{\},\{\},\{\},\{\},\{\})$.

(c) A reachability set and a global state graph may be infinite. This is due to the infinite queues which are used to model the exchange of coordination messages between parallel test components. An example is shown in Figure 8. (a) presents a a part of a test case control structure which causes an infinite global state graph. We assume the existens of a master test component *MTC* and a parallel test component *PTC*. The master test component *MTC* is able send arbitrary often coordination messages to *PTC*. This causes an infinite global state graph. The structure of the infinite part is shown in Figure 8 (b).

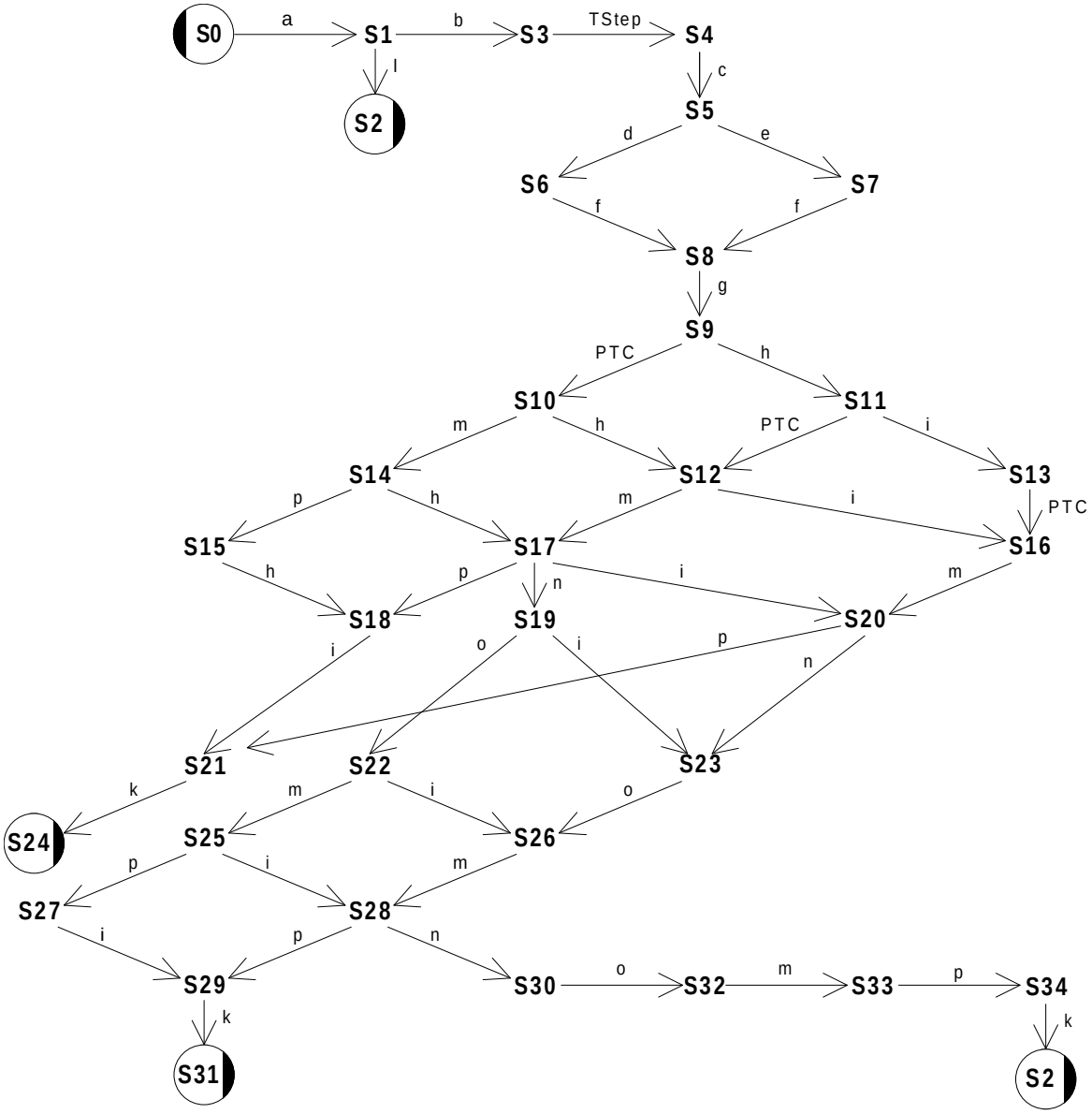
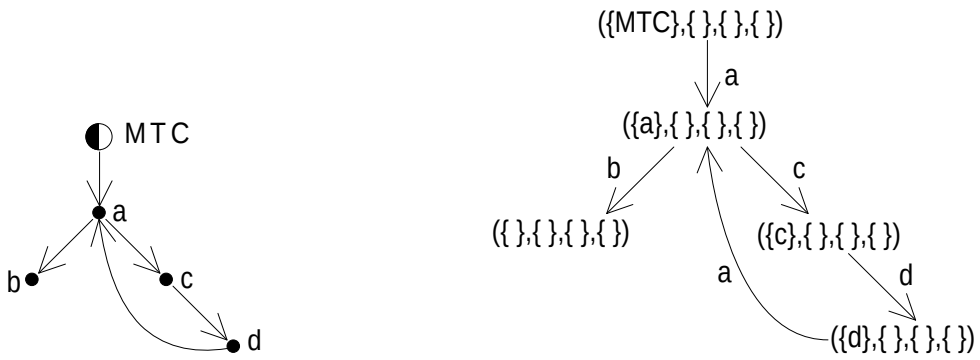


Figure 6: Global state graph

3.6 Some characteristics of the developed control flow model

Example 3.5 has shown some characteristics of the developed model which should be mentioned here.

Pending coordination messages. The test case of Example 3.5 (a) may end in the final states $S24 = (\emptyset, \emptyset, \emptyset, \{h \cdot i_{(cp,MTC,PTC)}, \perp\})$ and $S31 = (\emptyset, \emptyset, \emptyset, \{i_{(cp,MTC,PTC)}, \perp\})$. In both states the queue $q_{(cp,MTC,PTC)}$ of the coordination point cp is not empty, i.e. there exist pending coordination messages. For the definitions this is no problem, but we define a semantical model for TTCN. Often it is desired to concatenate test cases in order to facilitate the test campaign. In such cases pending coordination messages may have to



(a) Test case control structure (b) Global state graph of (a)

Figure 7: Global state graph with a loop

be destroyed. However, the TTCN language definition also does not care about pending coordination messages after test case termination.

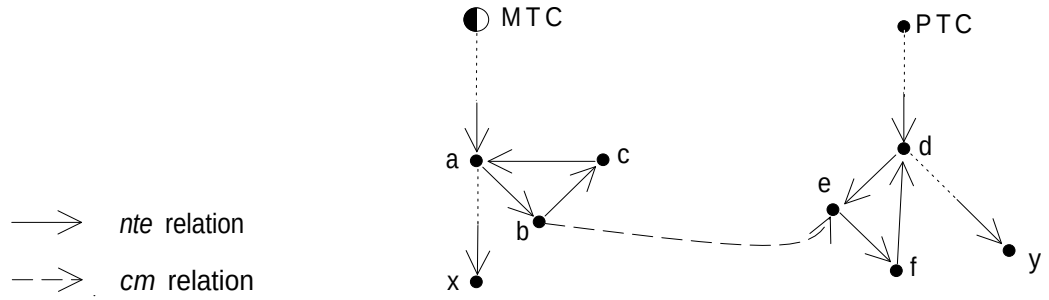
Finite and infinite global state graphs. We already have seen that there exist finite and infinite global state graphs, resp. reachability sets. This is due to the existence of infinite queues which are used for communication and synchronization purposes between parallel test components. For *normal* TTCN, i.e. without parallel test components, reachability sets and global state graphs are finite. In this case an upper bound for the number of states can be calculated. For a control state $S := (S_{nte}, S_{attach}, S_{create}, S_{cm})$ the maximal number of values for S_{nte}, S_{attach}, and S_{create} is equal to $\#(TE)$, i.e., the number of test events of the test case control structure. Since no parallel test components exist the only value of S_{cm} is $\emptyset$. Therefore an upper bound for the number of control states is given by $(\#(TE))^3$.

Finite and infinite firing sequences. A global state graph may be finite, infinite, and may contain loops. A test case has only finite firing sequences if the corresponding global state graph is finite and does not include loops. Loops and an infinite state space may lead to infinite firing sequences.

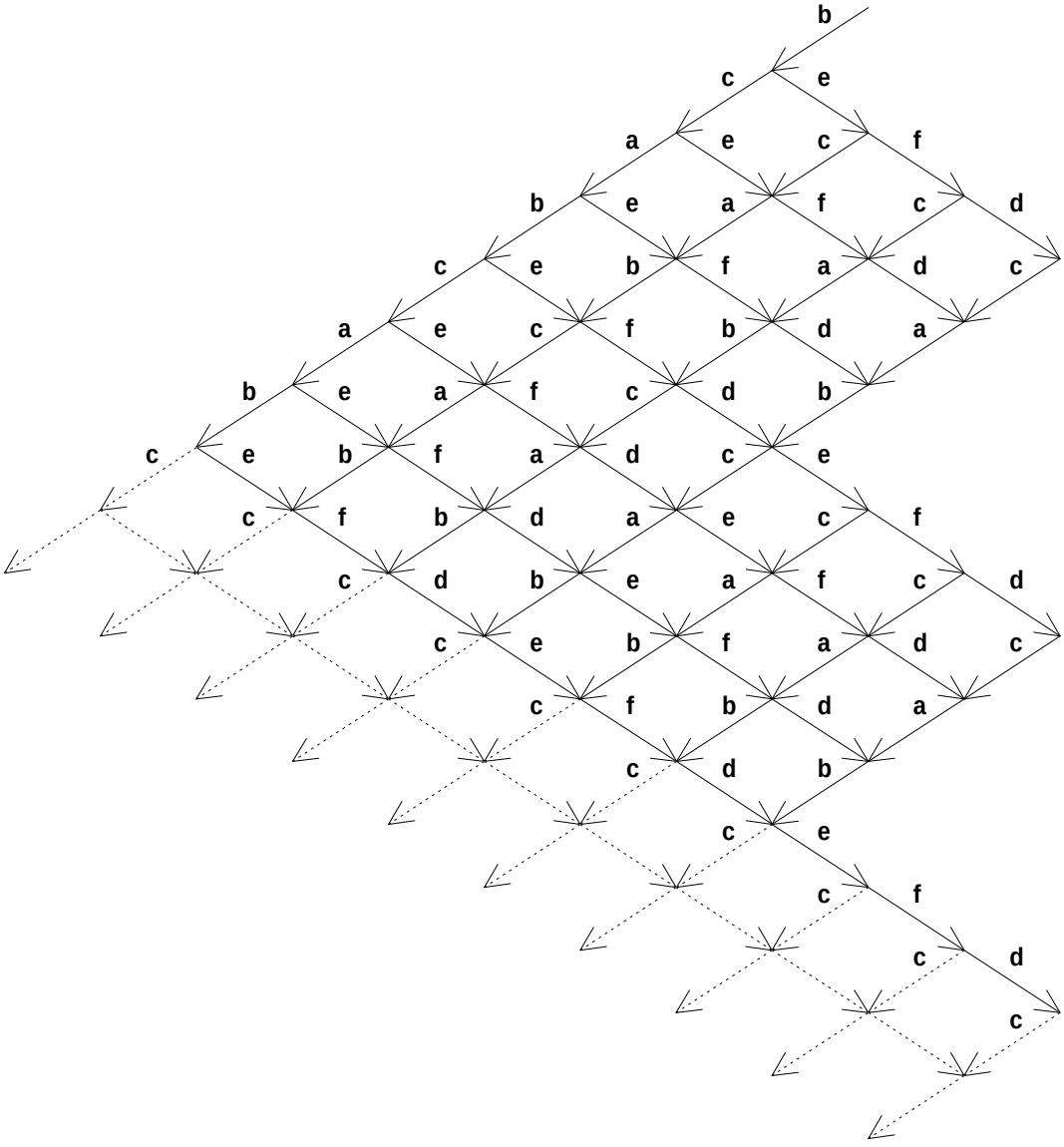
Checking for finiteness. By comparing each new state with already reached states it is possible to detect loops or an infinite state space. The comparison can be done by using Definition 3.5.

3.7 On the mapping of test cases on our control flow model

The introduced model allows to describe the control flow of TTCN test cases. The whole model is based on a directed graph with four sorts of edges which we called test case control structure. The mapping of a TTCN test case description onto a test case control structure has been described only informally. For providing a complete model of the semantics of the test case control flow it is necessary to define a bijection which maps



(a) Part of a test case control structure



(b) Infinite part of a global state graph

Figure 8: Infinite global state graph

a TTCN test case onto a test case control structure. The definition of this function is complicated because for the sake of simplicity we mapped some TTCN events onto more than one node of the test case control structure and introduce some some extra nodes into the test case control structure which do not correspond to behavior lines within the represented TTCN behavior description.

However, it is not our aim to define a complete TTCN semantics. Therefore we leave the mapping informal but explain the exceptional cases, i.e., the additional nodes, and a special case, i.e., the handling of default behavior descriptions. Before doing this we describe the principles followed during the development of the model.

Principles

The provided model follows some principles which should be mentioned here. The knowledge of these principles may also help to motivate the model.

The first principle was not to define the control enabling condition and execution rule for each type of TTCN event, i.e. the events CREATE, SEND, or RECEIVE are treated with the same control enabling condition and execution rule. The drawback of this modeling is that the influence on the control flow of the different event types is described by relations.

The second principle was to keep the definitions as simple as possible in order to allow an efficient implementation. We think that the result, i.e., the Definitions 3.4, 3.6, and 3.7, is relative simple. Simple set operations are used mainly. Only the treatment of the queues at the coordination points need some special treatment.

The third principle was to keep the mapping of the TTCN behavior description onto the test case control structure simple. We tried to follow this principle by mapping each TTCN event, i.e., each behavior line, onto one node of the test case control structure. This is only possible by violating the first and/or the second principle, i.e., we either define control enabling conditions and execution rules for the different types of TTCN events, or we provide very complicated definitions. We decided not to violate the first and second principle, but to map some TTCN events onto more than one node of the test case control structure and to introduce some some extra nodes which do not correspond to a behavior line within the represented TTCN behavior description.

Additional nodes in the test case control structure

We already have seen some additional nodes in the previous examples. Top nodes have been introduced and the ATTACH statement is modeled by two nodes. There is a third exception, UNTIL statements of REPEAT-UNTIL loops also is modeled by two nodes.

Top nodes. The top nodes are necessary to model the creation of test components correctly. Problems occur if the first specified TTCN event of a created component is part of a loop. In this case our control enabling condition in Definition 3.6 would not work without the top nodes. An example is shown in Figure 9. The graph in (a) shows our modeling of test component creation. An alternative modeling is shown in (b). The *create* arrow relates the create event *CREATE(PTC)* directly the test event *B!S2* (*LA*

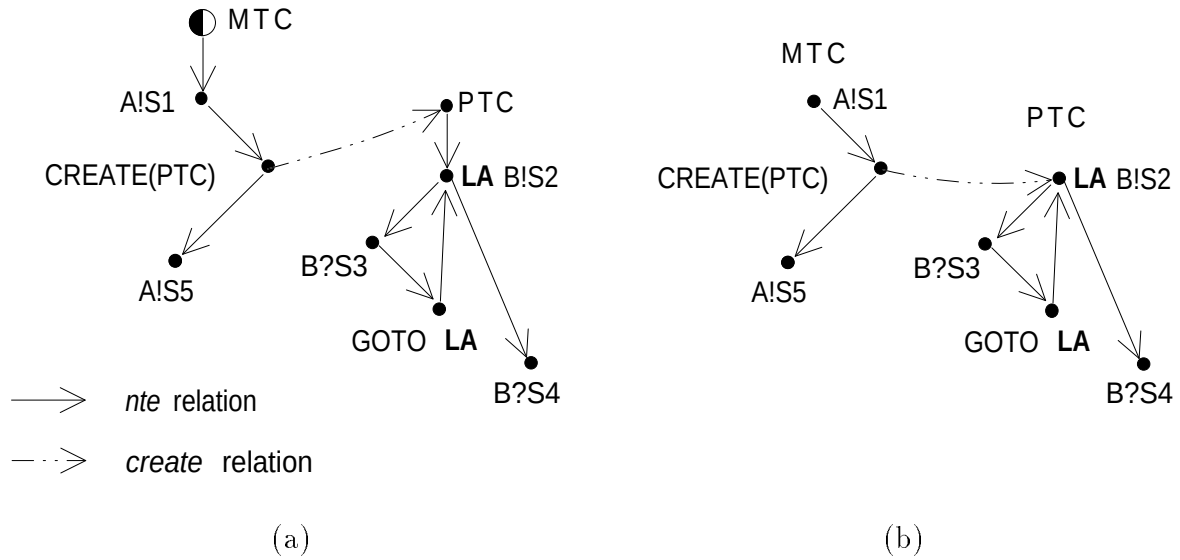


Figure 9: The necessity of top nodes

denotes the label which is used in the *GOTO* statement). But $B!S2$ is part of a loop and therefore ${}^\bullet B!S2_{nte} \neq \emptyset$. According to our control enabling definition the execution of the create event *CREATE(PTC)* is not sufficient to enable $B!S2$. As shown in (a) an additional top node PTC is necessary to model the test component creation correctly.

Modeling of the ATTACH construct. The reason to model an attach statement by two nodes is the possibility that the statement following the attachment is part of a loop. Figure 10 presents our representation (a) and an alternative modeling (b). In (b) the *attach* arrow relates the attachment directly with the following TTCN statement $A!S1$. According to our enabling condition $A!S1$ is only control enabled if an element of ${}^\bullet A!S1_{attach}$ and an element of ${}^\bullet A!S1_{create}$ is part of the actual state. According to Definition 3.7 $A!S1$ can only be control enabled once. Since $B?S2$ is the predecessor of $A!S1$ within the loop and since $B?S2^\bullet_{attach} = \emptyset$ it is not possible to control enable $A!S1$ during a second execution of the loop. To avoid such situations we model an attach statement by means of two nodes.

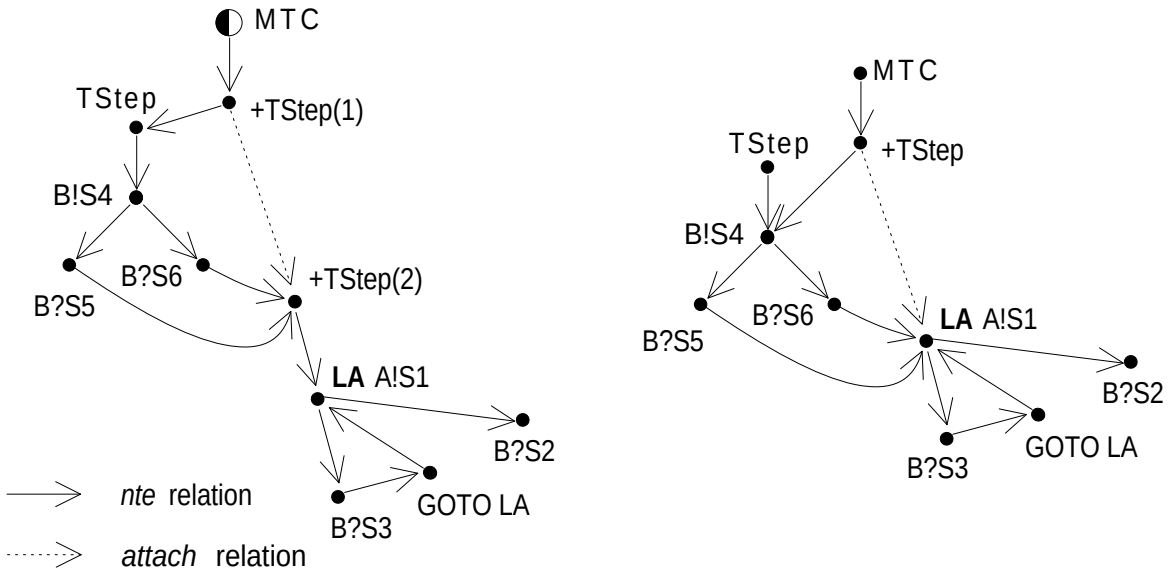
Modeling the Repeat construct. The BNF syntax of the Repeat construct is:

$$Repeat ::= \textbf{REPEAT}\ TreeReference[ActualParList]\ \textbf{UNTIL}\ Qualifier$$

The Repeat construct is modeled by three nodes within a test case control structure. One node represents the beginning of the loop, i.e., the keyword REPEAT. The UNTIL statement is modeled by two nodes. The REPEAT node is related to the UNTIL nodes by means of *attach* relations (cf. Figure 11 (b)).

The reason for the *attach* relations is the fact that a Repeat may use a normal test step as body of the loop. The test step may also be referred to by normal attach statements. In this case we need to remember the point from which the test step is called. Analogous to tree attachments therefore we use *attach* relations.

The reason to model the UNTIL statement by two nodes is the following. Depending



(a) (b)

Figure 10: The necessity to model attach statements by two nodes

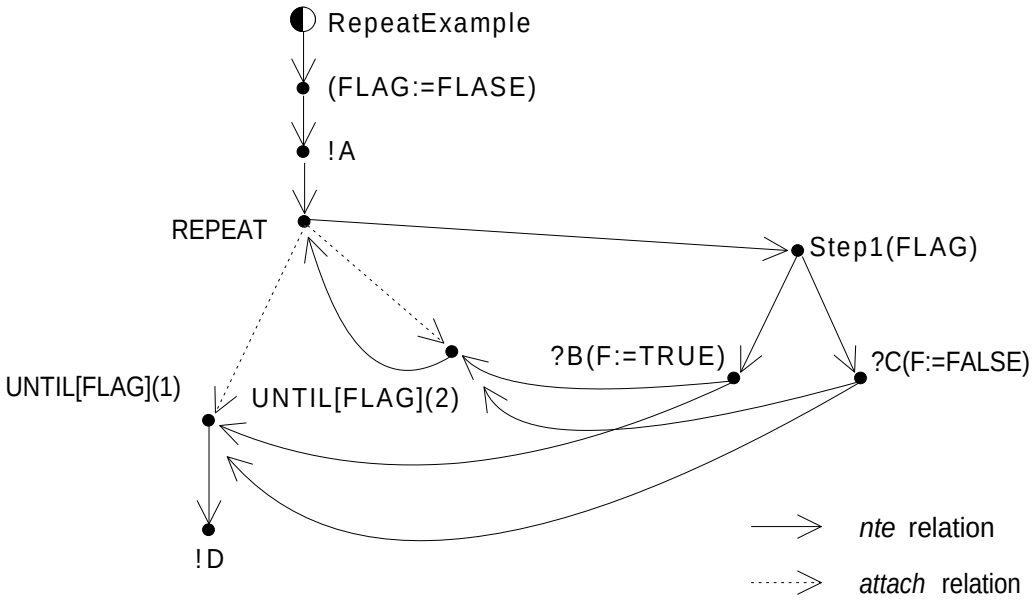
on the evaluation of the qualifier the UNTIL statement may lead to different actions. If the qualifier evaluates to *true* the Repeat construct is completed, i.e., the loop is left. If not, the tree $TreeReference$ is executed again, followed by another evaluation of the qualifier. This process is repeated until the qualifier evaluates to *true*. We decided to model the two actions of the UNTIL by two nodes. Later on we will introduce control data then the expressions $Qualifier$ and $\neg Qualifier$ will be used as enabling conditions for the different actions. Figure 11 shows an example of how the Repeat construct is modeled. The example is taken from [10]. It also shows the representation of TTCN Pseudo events, i.e., assignments, qualifiers, and timer operations without any associated test event (cf. line 1 in Figure 11 (a)). It should be noted that assignments, qualifiers and timer operations with associated test events are not evaluated in a test case control structure.

The handling of default behavior descriptions

TTCN introduces default behavior descriptions as a special sort of tree attachment which is done implicitly. In normal TTCN the test case terminates in the default behavior tree. The TTCN extensions also allow the return of the control from the default back to the main tree by means of RETURN statements, and the enabling and disabling of different default behaviors during the test run by means of ACTIVATE and DEACTIVATE statements. By treating default behavior descriptions as normal tree attachments our model is able to deal with defaults. ACTIVATE and DEACTIVATE statements describe which defaults where have to be attached.

Test Case Dynamic Behaviour		
Test Case Name: RepeatExample		
Nr	Label	Behaviour Description
1		(FLAG:=FALSE)
2		!A
3		REPEAT STEP1(FLAG) UNTIL[FLAG]
4		!D
		STEP1(F:BOOLEAN)
5		?B(F:=TRUE)
6		?C(F:=FALSE)

(a) TTCN behavior tree



(b) Test case control structure of (a)

Figure 11: Modeling the Repeat construct

3.8 Restrictions

The provided semantical model is able to handle most of the TTCN language constructs. But, there are three exceptions. These are recursive tree attachments, handling of timer and termination of parallel test components.

Recursive tree attachments

For the sake of simplicity the effect of recursive tree attachments has not been modeled. However, one way to model recursive tree attachments is to introduce queues (for each

test component one) for the treatment of *attach* relations within test case control states (cf. Definition 3.4). Queues are necessary to identify the correct origin of an attachment, i.e., the point to which the control returns when the attached tree is completed. The introduction of such queues leads to a more complicated control enabling condition (cf. Definition 3.6) and execution rule (cf. Definition 3.7).

The introduction of additional queues is feasible, but the influence of recursive tree attachments on data is very complicated. Data states, i.e., variable values, have to be duplicated and stored (in a queue) when the control is given to an attached tree and restored when the tree is completed. To avoid the definition of such a complex procedure we restrict ourselves to test cases without recursive tree attachments.

Timer handling

The timer handling in TTCN is modeled by the language constructs READTIMER, START, CANCEL, and TIMEOUT. A READTIMER operation has no direct influence on the control flow of a TTCN test case. It can be used to read the amount of time that has passed for a currently running timer.

The constructs START, TIMEOUT, and CANCEL have direct influence on the control flow of a test case. A timeout operation can only be executed if a corresponding timer was started, and a cancel operation may be used prevent the execution of timeout operations.

The modeling of the timer handling in our semantical model needs the introduction of two new relations into the test case control structure. A *timer* relation which is necessary to model the dependency between timer start and timeout operations, and a *cancel* relation which defines the relation between start and cancel operations. The two relations are necessary because the effect of cancel and timeout operations on the control flow of the test case are different. A cancel operation can always be executed, whereas a timeout always needs a corresponding timer start operation. Figure 12 shows schematically the modeling of the timer handling within test case control structures.

Subsequently, the definitions of test case control states (cf. Definition 3.4), control enabling of test events (cf. Definition 3.6), and execution of control enabled test events (cf. Definition 3.7) have to be extended.

Definition 3.4a (Test case control state with timer handling) *A test case control state is defined by a tuple* $CS = (S_{nte}, S_{attach}, S_{create}, S_{cm}, S_{timer})$*, where preconditions, (a), (b), (c), and (d) are the same as in Definition 3.4 and for* S_{timer} *the following holds:*

(e) $S_{timer} \subseteq TE$ *is a set of test events.*

Definition 3.6a (Control enabling of a test event with timer handling) *Let* $CS = (S_{nte}, S_{attach}, S_{create}, S_{cm}, S_{timer})$ *be a test case control state according to Definition 3.4a. A test event* $te \in TE$ *is control enabled in* CS*, written* $CS \xrightarrow{te}$*, if the preconditions, (a), (b), (c), and (d) of Definition 3.6 and the following condition holds:*

(e) ${}^\bullet te_{timer} = \emptyset \vee {}^\bullet te_{timer} \cap S_{timer} \neq \emptyset$

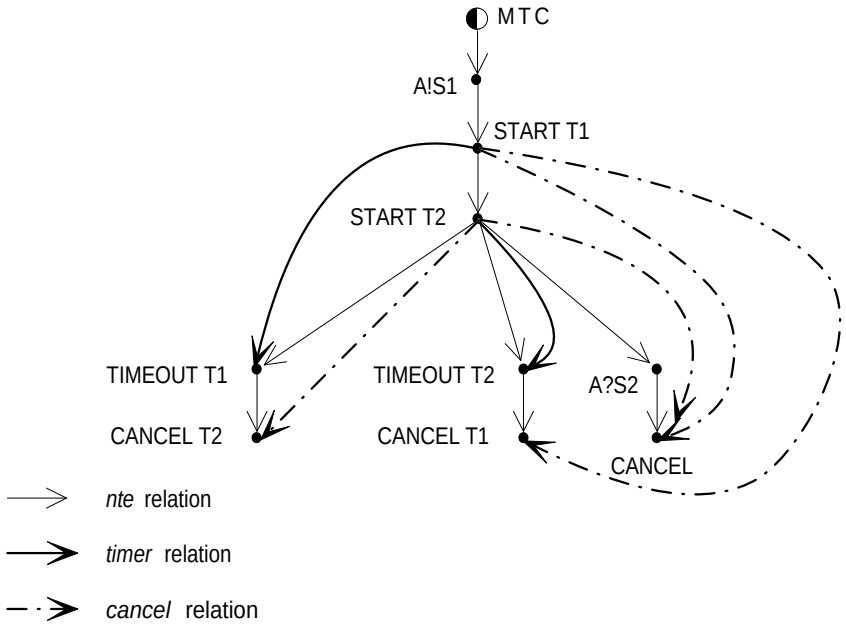


Figure 12: Modeling the timer handling within test case control structures

Definition 3.7a (Execution of a control enabled test event with timer handling) *Let $te \in TE$ be enabled in $CS = (S_{nte}, S_{attach}, S_{create}, S_{cm}, S_{timer})$. The execution of te changes the control CS into the new control state CS'. The new control state $CS' = (S'_{nte}, S'_{attach}, S'_{create}, S'_{cm}, S'_{timer})$ is defined by (a), (b), (c), and (d) of Definition 3.7 and the following equation:*

$$Let\ T_{element} = \begin{cases} \emptyset & : \ {}^{\bullet}te_{timer} = \emptyset \\ \{t \mid t \in S_{timer}\ is\ one\ arbitrary\ element\ of\ {}^{\bullet}te_{timer}\} & : \ {}^{\bullet}te_{timer} \neq \emptyset \end{cases}$$

$$(e)\ S'_{timer} = S_{timer} \backslash \{T_{element} \cup {}^{\bullet}te_{cancel}\} \cup \begin{cases} \emptyset & : \ te^{\bullet}_{timer} = \emptyset \\ \{te\} & : \ te^{\bullet}_{timer} \neq \emptyset \end{cases}$$

The previous definitions should give an idea of how the timer handling can be implemented in our semantical model. The proposed modeling seems to be easy, but it should be noted that it includes at least one problematic point. The condition (e) of Definition 3.7 introduces indeterminism on the execution level. Several start timer operations may enable the same timeout event. In such a case an arbitrary start event is chosen when the timeout is executed. This means, depending on the chosen start operation the execution of the timeout event may lead to different control states.

However, it is our aim to generate traces from TTCN test case descriptions. The modeling of the TTCN timer constructs will prevent only the generation of traces which include timeout operations without a corresponding running timer. We believe that it is easier and faster to exclude such traces after their generation than to implement the entire timer mechanism within our TTCN simulator.

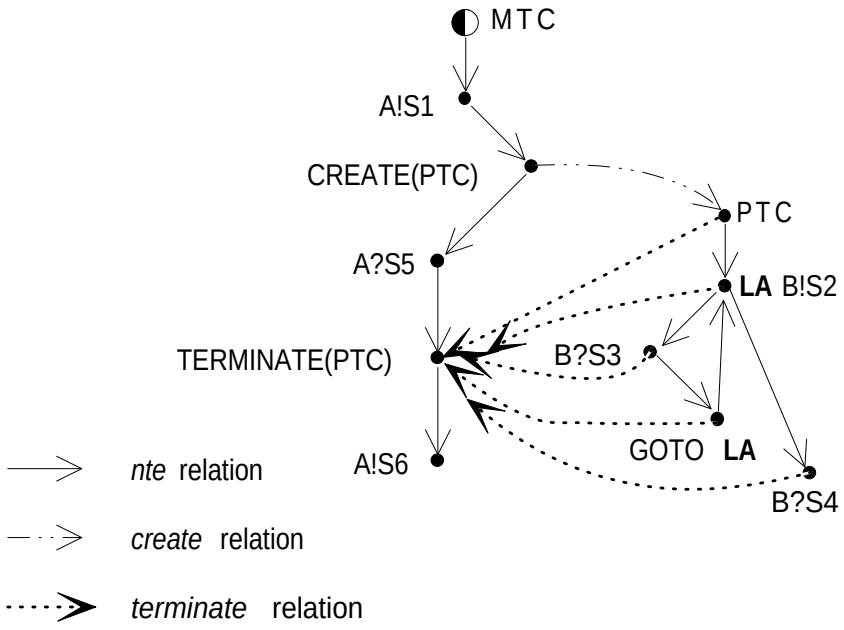


Figure 13: Modeling the terminate construct within test case control structures

Termination of parallel test components

The second problematic point of our semantical model for the control flow of TTCN test cases is the explicit termination of parallel test components. In TTCN this is done by means of the TERMINATE construct.

The modeling of the terminate operation needs the introduction of a new *terminate* relation into the test case control structure. All test events of parallel test component which may be terminated are related with the possible terminate test events of the main test component. Figure 13 describes this schematically. For simulation purposes only Definition 3.7 has to be extended.

Definition 3.7b (Execution of a control enabled test event (including TERMINATE event)) *Let the preconditions be the same as in Definition 3.7. The execution of the test event te changes the control CS into the new control state CS′. The new control state* $CS' = (S'_{nte}, S'_{attach}, S'_{create}, S'_{cm})$ *is defined by (b), (c), and (d) of Definition 3.7 and the following new equation (a):*

$$(a)\ S'_{nte} = S_{nte} \backslash \{ {}^{\bullet}te_{nte} \cup {}^{\bullet}te_{terminate} \} \cup \begin{cases} \emptyset & : \ te^{\bullet}_{nte} = \emptyset \\ \{te\} & : \ te^{\bullet}_{nte} \neq \emptyset \end{cases}$$

It is not very complicated to model the effect of the terminate construct within our semantical model. The problem is that complex test cases with several terminate operations may lead to very complicate test case control structures with an enormous number of terminate relations. In order to avoid the handling of complex graphs we do not model the terminate construct in our simulator. It should be noted that the modeling of the terminate construct will prevent the generation of traces which include events of already

terminated test components. Such traces can also be excluded easily after their generation.

4 Modeling data in TTCN test cases

In the previous section we defined the influence of the control structure on the dynamic behavior of a TTCN test case. In this section we define the influence of data on the behavior. The procedure corresponds to the procedure used in the previous section. We define a *'data structure of a TTCN test case'* (Section 4.1), a *'data state of a TTCN test case'* (Section 4.2), a *'data enabling condition for test events'* (Section 4.3), and the *'execution rules'* for data enabled test events (Section 4.4).

4.1 The data structure of a TTCN test case

Definition 4.1 (Data structure of a TTCN test case) *The data structure of a test case TC is defined by the tuple $TCDS = (TE; SV, IV; G, VT; \phi, \psi)$, where:*

(a) $TE \neq \emptyset$ is a finite set of test events.

(b) SV and IV are finite sets of typed variables.

- *The elements in $SV = \{v_1{:}D_1, \ \ldots \ , v_n{:}D_n\}$ are called 'state variables'. Each state variable $v_i{:}D_i$ has a name v_i and a type D_i. We assume that all variable names in SV are distinct.*
- *The elements in $IV = \{iv_1{:}D_1^{iv}, \ \ldots \ , iv_l{:}D_l^{iv}\}$ are called 'intermediate variables'. Each intermediate variable $iv_i{:}D_i^{iv}$ has a name iv_i and a type D_i^{iv}. We assume that all variable names in IV are distinct.*

(c) G and VT are finite sets of expressions.

- *The elements in G are called 'guards'. A guard $g \in G$ may contain elements of $SV = \{v_1{:}D_1, \ \ldots \ , v_n{:}D_n\}$ and $IV = \{iv_1{:}D_1^{iv}, \ \ldots \ , iv_l{:}D_l^{iv}\}$ as free variables. We assume that the lambda expression $\lambda(v_1, \ \ldots \ , v_n, iv_1, \ \ldots \ , iv_l).g$ defines a mapping from $D_1 \times \ldots \times D_n \times D_1^{iv} \times \ldots \times D_l^{iv}$ into $\{true, false\}$.*
- *The elements in VT are called variable transitions. A variable transition $vt \in VT$ may contain elements of $SV = \{v_1{:}D_1, \ \ldots \ , v_n{:}D_n\}$ and $IV = \{iv_1{:}D_1^{iv}, \ \ldots \ , iv_l{:}D_l^{iv}\}$ as free variables. We assume that the lambda expression $\lambda(v_1, \ \ldots \ , v_n, iv_1, \ \ldots \ , iv_l).vt$ defines a mapping from $D_1 \times \ldots \times D_n \times D_1^{iv} \times \ldots \times D_l^{iv}$ into $\{ran(v) \mid v \in SV\}$.*

(d) ϕ and ψ are functions which relate test events and expressions in G and VT.

- *$\phi : TE \rightarrow G$ is a function which relates each test event to a guard.*

- $\psi : TE \rightarrow VT^{\#(SV)}$ *is a function which relates a tuple* $(vt_1, \ldots, vt_{\#(SV)})$ *of expressions in* VT *to each test event. For each test event* te *a tuple element* vt_i *corresponds to a variable* $v_i{:}D_i \in SV$ *in such a way that the lambda expression* $\lambda(v_1, \ldots, v_{\#(SV)}, iv_1, \ldots, iv_{\#(IV)}).exp_i$ $(iv_i{:}D_i^{iv} \in IV)$ *defines a mapping from* $D_1 \times \ldots \times D_{\#(SV)} \times D_1^{iv} \times \ldots \times D_{\#(IV)}^{iv}$ *into* $ran(v_i)$. *Within the lambda expression each state and intermediate variable is referred to exactly once*

Explanation of Definition 4.1. The data structure of a TTCN test case is a tuple $(TE; SV, IV; G, VT; \phi, \psi)$ which contains several sets and functions. The meaning and use of these sets and functions may need some explanation.

- **Test events.** The set TE describes the test events of the TTCN test case. This set is identical to the set of test events in the test case control structure (cf. Definition 3.1).

- **State variables.** State variables are the elements in the set SV. They describe all variables and constants (a constant is meant to be a variable which does not change its value during the test run) which are declared for the whole test suite or just for the test case. This also includes variables declared implicitly, e.g., the variable R which during the test run holds preliminary test verdicts, and parameters.

- **Intermediate variables.** Intermediate variables are the elements in the set IV. Their meaning is related to test events describing the reception of PDUs, ASPs, and coordination messages. State variables may depend on parameters of PDUs, ASPs, or coordination messages. In order to describe this influence in an adequate manner, we use intermediate variables, i.e., within expressions intermediate variables represent message parameters.

- **Guards.** TTCN test events can be guarded by qualifiers. A qualifier is a boolean expression. During a test run a test event can only be executed if its qualifier evaluates to *true*. The *guards* in our definition, i.e., elements of G, are used for the same purpose. They represent TTCN qualifiers. Since the value of a qualifier may depend on message parameters our definition considers intermediate variables.

- **Variable transitions.** The elements of the set VT are called *variable transition*. They are used to describe the change of state variables during the test run. A variable transition is an expression. If it is evaluated during the test run the resulting value is assigned to a state variable. Since message parameter values may have an influence on the value of state variables our definition considers intermediate variables.

  In a TTCN test case variable transitions are provided within assignment operations, i.e., in the form $x_i = vt_i$ where x_i denotes the variable which is changed and vt_i the expression which defines how the new value of x_i will be calculated. In order to treat guards and variable transitions in the same way, in our definition we do not refer to the complete assignment operations, but to their components. For example, let te be a test event which is annotated with the assignment operation $x_i = vt_i$. In

Test Case Dynamic Behaviour				
Test Case Name: DataExample				
Nr	Label	Behaviour Description	Constraints Ref	Verdict
1		?CONind		
2		!CONresp (Count:=1)		
3		REPEAT LOCAL UNTIL [Count=3]		
4		?DISind [DISind.err="Wrong ACK"]		
5		!DISresp		PASS
6		?DISind		
7		!DISresp		FAIL
		LOCAL		
8		?DAT (S:=DAT.SeNr, Count:=Count+1)		(PASS)
9		!AK (AK.SeNr:=S-1)		

Figure 14: Behaviour description of a TTCN test case

this case vt_i defines the variable transition and x_i refers to the place of vt_i within the tuple $\psi(te)$.

It is obvious that guards and variable transitions will not always use all state and intermediate variables, although we bind all variables to all expressions. This is only for facilitating the definitions of an *data enabling condition* (Definition 4.3) and the *execution of data enabled* test events (Definition 4.4). During the evaluation of an expression the values of unused variables have to be ignored. It should also be noted, that the expressions in VT will only be used for the assignment of values to state variables. For the moment, we do not care about how the values of IM will be determined.

- **The function ϕ.** The function ϕ assigns a guard to each test event. However, within a TTCN test case not all test events are guarded by qualifiers. In such a case we assign the guard *true*, i.e., the execution of test event does not depend on the data state of the test case.

- **The function ψ.** The function *psi* assigns a tuple of expressions to each test event. Each expression describes how one state variable changes its value if the test event is executed. If a variable does not change its value the expression for this variable is the *identity function*.

Example 4.1 (Example of a test case data structure) We explain the meaning of Definition 4.1 by using a simple TTCN test case. The behavior description of the example is shown in Figure 14. The test case has no parallel test components, no tree attachments, but includes one local tree which is used within a REPEAT-UNTIL loop.

The test case data structure is given by the tuple $TCDS = (TE; SV, IV; G, VT; \phi, \psi)$. The set TE includes all test events. These are given by the TTCN statements in the

behavior lines and some additional nodes which we introduced when we modeled the control flow of a TTCN test case (cf. Section 3). In our case the set TE is:

$$TE = \{?CONind_{Nr.1}, !CONresp_{Nr.2}, ?DISind_{Nr.4}, ?DISresp_{Nr.5}, REPEAT_{Nr.3}, UNTIL(1)_{Nr.3}, UNTIL(2)_{Nr.3}, ?DISind_{Nr.6}, ?DISresp_{Nr.7}, LOCAL_{top}, ?DAT_{Nr.8}, !AK_{Nr.9}\}$$

The indices $Nr.i$ describe the line number of the corresponding statements in the TTCN behavior description (cf. Figure 14). The additional nodes in TE are the top node $LOCAL_{top}$, and the two nodes $UNTIL(1)_{Nr.3}$ and $UNTIL(2)_{Nr.3}$ describing the two alternative behaviors of the UNTIL statement within the REPEAT-UNTIL loop. It should be noted that the top node of the test case which is introduced in the test case control structure (cf. Definition 3.1) is not an element of TE. The state variables SV are given by:

$$SV = \{Count{:}Integer, S{:}Integer, R{:}Verdicts\}$$

State variables have to be defined explicitly or are defined implicitly by the TTCN semantics. The definition of the variables $Count$ and S is shown in Figure 18. The variable R is defined implicitly. During the execution of a test case R holds the actual test verdict. Possible values are $none$, $PASS$, $(PASS)$, $INCONC$, $(INCONC)$, $FAIL$, and $(FAIL)$[3]. Intermediate variables IV are:

$$IV = \{DISind.err{:}CharString, DAT.SeNr{:}Integer\}$$

As indicated by the names they are used to store parameter values of the messages $DISind$ and DAT. Guards are given by the qualifiers within the test case description. Additionally, we add the expression $true$. It will be used for test events which are not qualified. G is given by:

$$G = \{true, Count{\neq}3, Count{=}3, DISind.err{=}"Wrong\ AK"\}$$

According to the explanation of Definition 4.1 variable transitions are expressions which have to be extracted from the assignment operations in the behavior lines of the test case. These operations are $Count{:=}1$, $S{:=}DAT.SeNr$, and $Count{:=}Count{+}1$. The variable transitions within these assignments are 1, $DAT.SeNr$,and $Count{+}1$. Although it may look a little bit strange, the constant 1 is a valid expression. Furthermore, the test case description includes some implicit assignments to R. R is set to $none$ when the test case starts, to $PASS$ in behavior line Nr 5 (cf. Figure 14), to $FAIL$ in line Nr 7, and to $(PASS)$ in line Nr 8. In order to cope with $unknown$ values (cf. Definition 2.9) and to facilitate the handling of variables which are not treated by the assignment operations in a behavior line, we add the value $unknown$ $\propto$ and identity functions for all state variables. The set VT is given by:

$$VT = \{1, DAT.SeNr, Count{+}1, none, PASS, FAIL, (PASS), \propto, id(Count), id(S), id(R)\}$$

The functions ϕ and ψ for our example are defined in the table in Figure 15. The table shows that for ϕ a lot of $true$ predicates and for ψ a lot of identity functions have

[3]TTCN allows the abbreviation of $PASS$, $(PASS)$, $INCONC$, $(INCONC)$, $FAIL$, and $(FAIL)$ by P, (P), I, (I), (F), and F.

test event (te)	$\phi(te)$	$\psi(te)$ tuples are meant to be assignments for: $(Count, S, R) \in Integer \times Integer \times Verdicts$
$?CONind_{Nr.1}$	$true$	$(id(Count), id(S), id(R))$
$!CONresp_{Nr.2}$	$true$	$(1, id(S), id(R))$
$REPEAT_{Nr.3}$	$true$	$(id(Count), id(S), id(R))$
$UNTIL(1)_{Nr.3}$	$Count \neq 3$	$(id(Count), id(S), id(R))$
$UNTIL(2)_{Nr.3}$	$Count=3$	$(id(Count), id(S), id(R))$
$?DISind_{Nr.4}$	$DISind.err=$"$Wrong\ AK$"	$(id(Count), id(S), id(R))$
$?DISresp_{Nr.5}$	$true$	$(id(Count), id(S), PASS)$
$?DISind_{Nr.6}$	$true$	$(id(Count), id(S), id(R))$
$?DISresp_{Nr.7}$	$true$	$(id(Count), id(S), FAIL)$
$LOCAL_{top}$	$true$	$(id(Count), id(S), id(R))$
$?DAT_{Nr.8}$	$true$	$(Count{+}1, DAT.SeNr, (PASS))$
$!AK_{Nr.9}$	$true$	$(id(Count), id(S), id(R))$

Figure 15: The function ϕ and ψ of Example 4.1

to be assigned.

The functions ϕ and ψ also assign expressions to the top node $LOCAL_{top}$. The guard is *true* and the variable transitions are identity functions. The variable transitions need not to be identity functions. Variable transitions associated with top nodes are also used to assign values to parameters, e.g., in the case of parameterized test steps or test components.

By comparing the table with the TTCN description (cf. Figure 14) we see that the assignment in behavior line Nr 9 ($AK.SeNr := S - 1$) is not considered by ψ. This is due to the fact that this is an assignment to the intermediate variable $AK.SeNr$. In our definition of ψ we only take care of state variables.

4.2 The data state of a TTCN test case

The data state of a TTCN test case is a tuple of values. Each element of the tuple corresponds to the value of a state variable when the test case is in this data state.

Definition 4.2 (Data state of a TTCN test case) *Let TC be a TTCN test case and $TCDS = (TE; SV, IV; G, VT; \phi, \psi)$ be the corresponding data structure where $SV = \{v_1{:}D_1, \ \ldots \ , v_n{:}D_n\}$ denotes the set of state variables. A data state of TC test case is a tuple $DS = (x_1, \ \ldots \ , x_n) \in D_1 \times \ldots \times D_n$. Each element of DS holds the actual value of a state variable when the test case is in the data state DS.*

Example 4.2 (Data states for Example 4.1) Some examples of data states for Example 4.1 may help to understand the meaning of the definition. The state variables of the example are $SV = \{Count{:}Integer, S{:}Integer, R{:}Verdicts\}$. Therefore data states have to be elements of $Integer \times Integer \times Verdicts$. Some examples:

- $DS_1 = (\propto, \propto, \propto)$ denotes that the values of all state variables are *unknown*.

- $DS_2 = (1, \propto, none)$ states that the value of $Count$ is 1, the value of S is *unkown*, and the value of R is *none*.
- $DS_3 = (3, 5, (PASS))$ denotes that $Count = 3$, $S = 5$, and $R = (PASS)$.

4.3 The data enabling of test events

The data enabling of a test event is defined via the predicates which are associated with the test events. The guard is evaluated by using the variable values of the actual state and a tuple containing the values of intermediate variables. If the guard evaluates to *true* the test event is *data enabled.*

Definition 4.3 (Data enabling of a test event) *Let TC be a TTCN test case,*

- *$TCDS = (TE; SV, IV; G, VT; \phi, \psi)$ be the corresponding data structure, where*
  - *$SV = \{v_1{:}D_1, \ \ldots \ , v_n{:}D_n\}$ denotes the set of state variables,*
  - *$IV = \{iv_1{:}D_1^{iv}, \ \ldots \ , iv_l{:}D_l^{iv}\}$ denotes the set of intermediate variables;*
- *$DS = (x_1, \ \ldots \ , x_n) \in D_1 \times \ldots \times D_n$ be a data state of TC;*
- *$IS = (x_1^{iv}, \ \ldots \ , x_l^{iv}) \in D_1^{iv} \times \ldots \times D_l^{iv}$ be a tuple which includes values for all intermediate variables;*
- *$DSIS = (x_1, \ \ldots \ , x_n, x_1^{iv}, \ \ldots \ , x_l^{iv}) \in D_1 \times \ldots \times D_n \times D_1^{iv} \times \ldots \times D_l^{iv}$ be a tuple which is gained by the concatenation of DS and IS;*
- *$te \in TE$ be a test event;*
- *$g = \phi(te) \in G$ be the guard related to te; and let*
- *$fg : D_1 \times \ldots \times D_n \times D_1^{iv} \times \ldots \times D_l^{iv} \rightarrow \{true, false\}$ be the function which is defined by the lambda expression $\lambda(v_1, \ \ldots \ , v_n, iv_1, \ \ldots \ , iv_l).g$.*

The test event te is data enabled in DS, written $DS \xrightarrow{te,IS}_D$, if $fp(DSIS) = true$.

The index D in $DS \xrightarrow{te,IS}_D$ indicates that the enabling condition is based on the information in the data structure of a test case. For the enabling condition based on the control structure we used a similar notation, but used the index c (cf. Definition 3.6). The IS above the arrow refers to the tuple of values for intermediate variables. IS describes the influence of PDU, ASP, and coordination message parameter values on the enabling of test events. Later on we will determine the values within IS from *receive* events within test logs and send events of coordination messages.

Example 4.3 (Data enabling) In Example 4.2 we defined some data states for data structure in Example 4.1. Here we describe which test events are data enabled if the test case is in the data state DS_1, DS_2, and DS_3.

test event (te)	$DS_i \xrightarrow{te,IS}_D (IS = (\propto, \propto))$		
	$DS_1 = (\propto, \propto, \propto)$	$DS_2 = (1, \propto, none)$	$DS_3 = (3, 5, (PASS))$
$?CONind_{Nr.1}$	enabled	enabled	enabled
$!CONresp_{Nr.2}$	enabled	enabled	enabled
$REPEAT_{Nr.3}$	enabled	enabled	enabled
$UNTIL(1)_{Nr.3}$	enabled	enabled	not enabled
$UNTIL(2)_{Nr.3}$	enabled	not enabled	enabled
$?DISind_{Nr.4}$	enabled	enabled	enabled
$?DISresp_{Nr.5}$	enabled	enabled	enabled
$?DISind_{Nr.6}$	enabled	enabled	enabled
$?DISresp_{Nr.7}$	enabled	enabled	enabled
$LOCAL_{top}$	enabled	enabled	enabled
$?DAT_{Nr.8}$	enabled	enabled	enabled
$!AK_{Nr.9}$	enabled	enabled	enabled

Figure 16: Data enabled events for the states DS_1, DS_2, and DS_3

For checking the enabling condition a tuple $IS = (x_1^{iv}, \ldots, x_l^{iv}) \in D_1^{iv} \times \ldots \times D_l^{iv}$ which includes values for all intermediate variables is needed. In our case the intermediate variables $DISind.err$:$CharString$ and $DAT.SeNr$:$Integer$ are used to describe the influence of the parameter values of the messages $DISind$ and DAT on the execution of the test case. We assume that we do not know anything about the parameter values $DISind.err$ and $DAT.SeNr$. This means $IS = (\propto, \propto)$ for the states DS_1, DS_2, and DS_3.

Which test events are data enabled in which states is shown in the table in Figure 16. It is obvious that test events with the guard $true$, i.e., $?CONind_{Nr.1}$, $!CONresp_{Nr.2}$, $REPEAT_{Nr.3}$, $?DISresp_{Nr.5}$, $?DISind_{Nr.6}$, $?DISresp_{Nr.7}$, $LOCAL_{top}$, $?DAT_{Nr.8}$, and $!AK_{Nr.9}$, are always enabled. Furthermore $?DISind_{Nr.4}$ is also data enabled in DS_1, DS_2, and DS_3, because its enabling depends on the value of an intermediate variable which is not known. According to Definition 2.9 predicates which depend on $unkown$ values always evaluate to $true$.

In our example only the test events $UNTIL(1)_{Nr.3}$ and $UNTIL(2)_{Nr.3}$ need special attention. Their data enabling depends on the actual value of the state variable $Count$. Subsequently, $UNTIL(1)_{Nr.3}$ is not data enabled in DS_3 and $UNTIL(2)_{Nr.3}$ is not data enabled in DS_2.

4.4 The execution of data enabled test events

Based on the knowledge of test case data structures, data states for TTCN test cases, and an data enabling condition for test events we are able to define an execution rule for data enabled test events. The execution rule defines how the data state changes when a test event is executed. Our execution rule makes use of the variable transitions, i.e., the set VT, which are assigned to the test events by means of the function ψ. The variable transitions are evaluated by using the values of the actual data state and the values of intermediate variables. The results of these calculations form the new data state.

Definition 4.4 (Execution of data enabled test events) *Let TC be a TTCN test case,*

- *$TCDS = (TE; SV, IV; G, VT; \phi, \psi)$ be the corresponding data structure, where*
  - *$SV = \{v_1{:}D_1, \ \ldots \ , v_n{:}D_n\}$ denotes the set of state variables,*
  - *$IV = \{iv_1{:}D_1^{iv}, \ \ldots \ , iv_l{:}D_l^{iv}\}$ denotes the set of intermediate variables;*
- *$DS = (x_1, \ \ldots \ , x_n) \in D_1 \times \ldots \times D_n$ be a data state of TC;*
- *$IS = (x_1^{iv}, \ \ldots \ , x_l^{iv}) \in D_1^{iv} \times \ldots \times D_l^{iv}$ be a tuple which includes values for all intermediate variables;*
- *$DSIS = (x_1, \ \ldots \ , x_n, x_1^{iv}, \ \ldots \ , x_l^{iv}) \in D_1 \times \ldots \times D_n \times D_1^{iv} \times \ldots \times D_l^{iv}$ be a tuple which is gained by the concatenation of DS and IS;*
- *$te \in TE$ be a test event which is data enabled in DS, i.e., $DS \xrightarrow{te,IS}_D$;*
- *$\psi(te)$ be the tuple $(vt_1, \ \ldots \ , vt_n)$;*
- *$vt_i \in VT$ $(1 \leq i \leq n)$ be the i.th expression in $\psi(te)$; and let*
- *$fvt_i : D_1 \times \ldots \times D_n \times D_1^{iv} \times \ldots \times D_l^{iv} \rightarrow \bigcup_{v \in SV} ran(v)$ be the function which is defined by the lambda expression $\lambda(v_1, \ \ldots \ , v_n, iv_1, \ \ldots \ , iv_l).vt_i$.*

The execution of te changes the data state $DS = (x_1, \ \ldots \ , x_n)$ into the new data state $DS' = (x'_1, \ \ldots \ , x'_n) \in D_1 \times \ldots \times D_n$. We denote this fact by $DS \xrightarrow{te,IS}_D DS'$. The values x'_i of the elements of new data state DS' are calculated by: $x'_i = ft^i(DSIS)$.

Example 4.4 (Execution of data enabled test events) The idea of Definition 4.4 should be clear intuitively. The actual data state is transformed into the new data state by evaluating the variable transitions which are associated to each test event. Information concerning ASPs, PDU, or coordination message parameter values can be considered by using intermediate variables.

For our example Figure 16 shows which test events are enabled in the data states $DS_1 = (\propto, \propto, \propto)$, $DS_2 = (1, \propto, none)$, and $DS_3 = (3, 5, (PASS))$. The table in Figure 17 shows the new states when the data enabled test events in Figure 16 are executed.

Test events which only perform *identityfunction* to the tuple elements of the actual state, i.e., $?CONind_{Nr.1}$, $REPEAT_{Nr.3}$, $UNTIL(1)_{Nr.3}$, $UNTIL(2)_{Nr.3}$, $?DISind_{Nr.4}$, $?DISind_{Nr.6}$, $LOCAL_{top}$, and $!AK_{Nr.9}$ (cf. Figure 15), need no further explanation.

During the execution of $!CONresp_{Nr.2}$ $Count$ is set to 1. Performing $?DISresp_{Nr.5}$ or $?DISresp_{Nr.7}$ leads to new values of the verdict variable R.

The calculation of the new states when $?DAT_{Nr.8}$ is executed is a little bit more tricky. Starting with the actual state DS_1 leads to *unknown* values for all state variables. For the variable $Count$ this is due to the rules for the evaluation of functions with *unknown* values. Independent from the starting data state the execution of $?DAT_{Nr.8}$ leads to an *unknown* value for S and the value $(PASS)$ for R. For S this is because of the unknown the actual value of the intermediate variable $DAT.SeNr$. For R a simple assignment of $(PASS)$ is performed.

test event (te)	for all cases $IS = (\propto, \propto)$)		
	$DS_1 \overset{te,IS}{\longrightarrow}_D DS'_1$	$DS_2 \overset{te,IS}{\longrightarrow}_D DS'_2$	$DS_3 \overset{te,IS}{\longrightarrow}_D DS'_3$
$?CONind_{Nr.1}$	$DS'_1 = DS_1$	$DS'_2 = DS_2$	$DS'_3 = DS_3$
$!CONresp_{Nr.2}$	$DS'_1 = (1, \propto, \propto)$	$DS'_2 = (1, \propto, none)$	$DS'_3 = (1, 5, (PASS))$
$REPEAT_{Nr.3}$	$DS'_1 = DS_1$	$DS'_2 = DS_2$	$DS'_3 = DS_3$
$UNTIL(1)_{Nr.3}$	$DS'_1 = DS_1$	$DS'_2 = DS_2$	–
$UNTIL(2)_{Nr.3}$	$DS'_1 = DS_1$	–	$DS'_3 = DS_3$
$?DISind_{Nr.4}$	$DS'_1 = DS_1$	$DS'_2 = DS_2$	$DS'_3 = DS_3$
$?DISresp_{Nr.5}$	$DS'_1 = (\propto, \propto, PASS)$	$DS'_2 = (1, \propto, PASS)$	$DS'_3 = (3, 5, PASS)$
$?DISind_{Nr.6}$	$DS'_1 = DS_1$	$DS'_2 = DS_2$	$DS'_3 = DS_3$
$?DISresp_{Nr.7}$	$DS'_1 = (\propto, \propto, FAIL)$	$DS'_2 = (1, \propto, FAIL)$	$DS'_3 = (3, 5, FAIL)$
$LOCAL_{top}$	$DS'_1 = DS_1$	$DS'_2 = DS_2$	$DS'_3 = DS_3$
$?DAT_{Nr.8}$	$DS'_1 = (\propto, \propto, (PASS))$	$DS'_2 = (2, \propto, (PASS))$	$DS'_3 = (4, \propto, (PASS))$
$!AK_{Nr.9}$	$DS'_1 = DS_1$	$DS'_2 = DS_2$	$DS'_3 = DS_3$

Figure 17: Executing the data enabled test events in Figure 16

4.5 Intermediate variables

Until now we did not have taken care about the calculation of values for intermediate variables. We have taken out their calculation from the test case data structure, because we want to be able to incorporate external information during the simulation of TTCN test cases. External information might come from test logs or from formal specifications. Beside this, we will also use intermediate variables to tackle the problem of exchanging information between parallel test components via coordination messages. To do this in a formal way we define the notion of an intermediate data structure.

Definition 4.5 (Intermediate data structure) *An intermediate data structure is a tuple $IDS = (E; EV, IV; IA, \theta)$, where:*

(a) $E \neq \emptyset$ is a finite set of events.

(b) IV and EV are finite sets of typed variables.

- *The elements in $EV = \{ev_1{:}D_1^{ev}, \ldots, ev_n{:}D_n^{ev}\}$ are called 'external variables'. Each external variable $ev_i{:}D_i^{ev}$ has a name ev_i and a type D_i^{ev}. We assume that all variable names in EV are distinct.*
- *The elements in $IV = \{iv_1{:}D_1^{iv}, \ldots, iv_l{:}D_l^{iv}\}$ are called 'intermediate variables'. Each intermediate variable $iv_i{:}D_i^{iv}$ has a name iv_i and a type D_i^{iv}. We assume that all variable names in IV are distinct.*

(c) IA is a finite set of expressions. The elements in IA are called 'intermediate assignments'. An intermediate assignment $ia \in IA$ may contain elements of $EV = \{ev_1{:}D_1^{ev}, \ldots, ev_n{:}D_n^{ev}\}$ as free variables. We assume that the lambda expression $\lambda(ev_1, \ldots, ev_n).ia$ defines a function $\alpha(ia)$ from $D_1^{ev} \times \ldots \times D_n^{ev}$ into $\bigcup_{iv:D \in IV} D$. Within the lambda expression each intermediate variable is referred to exactly once.

(d) $\theta : E \rightarrow IA^{\#(IV)}$ *is a function which relates a tuple* $(ia_1, \ldots, ia_{\#(IV)})$ *of expressions of* IA *to each event in* E*. For each event a tuple element* ia_i *corresponds to a variable* $iv_i{:}D_i^{iv} \in IV$ *in such a way that* $ran(\alpha(ia_i)) = ran(iv)$*.*

The definition of the intermediate data structure is independent from data structure of a TTCN test case. This allows us to relate an intermediate data structure to test logs and other system descriptions. We only need to identify the events which provide the values of intermediate variables. For doing this the intermediate variables in Definition 4.1 have to correspond to the intermediate variables in Definition 4.5.

However, we also want to use the intermediate data structure to tackle a problem which we did not have covered yet. It is the problem of exchanging information between parallel test components via coordination messages. Storing parameter values of coordination messages in intermediate variables only is not sufficient, because the information exchange is meant to be asynchronous.

The value of the message parameter has to be stored in an intermediate variable when the send event takes place. A problem occurs if a second message of the same type is sent before the receive event of the first send event has been performed. In this case the variable would be overwritten. One possibility to avoid this is to declare intermediate variables for all possible instances of a coordination message type. This may lead to infinite sets of variables.

We use another approach. We bind the events in Definition 4.5 to the send events of coordination messages. These events will be stored within the control state of a TTCN test case (cf. Definition 3.4) and will be available when the receive event of a coordination message is performed (cf. Definition 3.7). We only have to assume that we can determine the correct parameter value from the send event only. For this it might be necessary to evaluate the intermediate assignments before the send event is stored in the control state.

In order to facilitate the handling of intermediate data structures we define an $eval_{ids}$ function. The $eval_{ids}$ function calculates for a given event e and a given tuple of values for external variables a tuple of values for intermediate variables.

Definition 4.6 (The $eval_{ids}$ function)

- *Let the tuple* $IDS = (E; EV, IV; IA, \theta)$ *be an intermediate data structure, where* $EV = \{ev_1{:}D_1^{ev}, \ldots, ev_n{:}D_n^{ev}\}$ *and* $IV = \{iv_1{:}D_1^{iv}, \ldots, iv_l{:}D_l^{iv}\}$*.*
- *Let* $e \in E$ *be an event,* $\theta(e) = (exp_1, \ldots, exp_n) \in IA^{\#(IV)}$ *be the tuple of expressions assigned to* e *by* θ*, and let* $\alpha_i^{\theta(e)} : D_1^{ev} \times \ldots \times D_n^{ev} \rightarrow \bigcup_{iv:D \in IV} D$ *be the function which for each* $exp_i \in \theta(e)$ *is defined by the lambda expression* $\lambda(ev_1, \ldots, ev_n).exp_i$*.*
- *Furthermore, let* $DS = (x_1, \ldots, x_n) \in D_1^{ev} \times \ldots \times D_n^{ev}$ *be a tuple of values.*

The function $eval_{ids} : E \times (D_1^{ev} \times \ldots \times D_n^{ev}) \rightarrow D_1^{iv} \times \ldots \times D_l^{iv}$ *is defined by:*

$$eval_{ids}(e, DS) = (\alpha_1^{\theta(e)}(DS), \ldots \alpha_l^{\theta(e)}(DS))$$

Example 4.5 (The handling of intermediate data structures) A simple example may help to understand the meaning and handling of intermediate data structures. We

only consider the TTCN statement '$!Signal(Signal.Nr{:=}2+x, Signal.Info{:=}R)$' which should be bound to an intermediate data structure $IDS = (E; EV, IV; IA; \theta)$. There is only one event in E, i.e.,

$E = \{!Signal\}$

We assume there are two external and two internal variables only, i.e., EV and IV are given by:

$EV = \{x{:}Integer, R{:}Verdicts\}$

$IV = \{Signal.Nr{:}Integer, Signal.Info{:}Verdicts\}$

The intermediate assignments IA and $\theta(!Signal)$ are:

$IA = \{2+x, R\}$

$\theta(!Signal) = (2+x, R)$

For applying the function $eval_{ids}$ to $!Signal$ we need tuples $DS_i \in Integer \times Verdicts$. Example of such tuples are $DS_1 = (3, (PASS))$, $DS_2 = (\propto, FAIL)$, or $DS_3 = (5, \propto)$. Applying $eval_{ids}$ with these tuples leads to:

$eval_{ids}(!Signal, DS_1) = (5, (PASS))$

$eval_{ids}(!Signal, DS_2) = (\propto, FAIL)$

$eval_{ids}(!Signal, DS_3) = (7, \propto)$

4.6 A remark on the developed model for data handling

The goal for developing a model for the data handling in TTCN test cases was to provide a simple basis for the implementation of data aspects within a TTCN simulator. The developed model should reach this goal. It is very simple and compact. But, it does not provide a complete formal semanctics for TTCN data types, TTCN matching mechanisms, parameterization, or the use of ASN.1 within TTCN.

5 The simulation of TTCN test cases

Within the previous sections we covered control flow and data aspects of a TTCN test case. The aim of this section is to combine both aspects in order to provide a comprehensive and formal basis for the implementation of a TTCN simulator. Similarly, to the definitions for control flow and data aspects we start with a *state* definition which is followed by the definitions of an *enabling condition* and an *execution rule*. Furthermore we provide definitions for traces, reachability sets, and global state graphs. We only provide the definitions with some short explanations. The way how the definitions work will be described by two examples which will be presented in Section 6.

5.1 The state of a TTCN test case

This section provides definitions for the state of a TTCN test case and a $\leq$ relation for test case states. The $\leq$ relation is necessary of the definition of traces, reachability sets,

and global state graphs.

Definition 5.1 (State of a TTCN test case) *Let TC be a TTCN test case. A tuple $ST = (CS, DS)$ is a state of TC if CS is a valid test case control state of TC (cf. Definition 3.4) and DS is a valid data state of TC (cf. Definition 4.2).*

Definition 5.2 (Equality and order relation for TTCN test case states) *Let TC be a TTCN test case, and $ST = (CS, DS)$, $ST' = (CS', DS')$ be two states of TC.*

1. *$ST = ST'$ if $CS = CS' \wedge DS = DS'$*
2. *$ST < ST'$ if $CS < CS' \wedge DS = DS'$ (the $<$ relation for control states is defined in Definition 3.5)*

5.2 Enabling of test events

The enabling of a test event is defined by combining *control enabling* and *data enabling.*

Definition 5.3 (Enabling condition for a test event)

- *Let TC be a TTCN test case, where*
  - *$TCCS = (TE, t0, nte, attach, create, cm)$ is the corresponding test case control structure (cf. Definition 3.1), and*
  - *$TCDS = (TE; SV, IV; G, VT; \phi, \psi)$ be the corresponding test case data structure (cf. Definition 4.1).*
- *If TC includes parallel test components, let $IDS = (dom(cm); SV, IV; IA, \theta)$ be an intermediate data structure, (cf. Definition 4.5). The set $dom(cm) \subseteq TE$ denotes send events of coordination messages. We assume that the function $eval_{ids} : E \times (D_1 \times \ldots \times D_n) \rightarrow D_1^{iv} \times \ldots \times D_l^{iv}$ is defined for IDS.*
- *Let $te \in TE$ be a test event.*
- *Let $ST = (CS, DS)$ be a state of TC, where*
  - *$CS = (S_{nte}, S_{attach}, S_{create}, S_{cm})$ is a control of TC state and*
  - *$DS = (x_1, \ \ldots \ , x_n)$ denotes a data state of TC.*
- *Let $IS \in D_1^{iv} \times \ldots \times D_l^{iv}$ be a tuple of values for intermediate variables.*

The test event te is enabled in ST, written $ST \xrightarrow{te}_{c,d}$, if the following three conditions hold:

1. *$CS \xrightarrow{te}_c$ (cf. Definition 3.6),*
2. *$DS \xrightarrow{te,IS}_d$ (cf. Definition 4.3), and*

3. $IS = \begin{cases} (\propto_1, \ldots, \propto_l) & : \quad te \notin ran(cm) \\ eval_{ids}(next(qstate(tequeue(te))), DS) & : \quad else \end{cases}$

Condition *3.* includes a rule for the calculation of IS. Therefore, in contrast to the short notation for the data enabling condition, i.e., $\xrightarrow{te,IS}_d$, the short notation for the enabling condition above ,i.e., $\xrightarrow{te}_{c,d}$, includes no reference to IS.

However, as already mentioned, the enabling condition for a test event is defined by combining control enabling and data enabling. This is denoted in the conditions *1.* and *2.* of the definition above. Only condition *3.* needs some further explanation.

In the case of parallel TTCN information may be exchanged by using parameters of coordination messages. This information has to be considered during the simulation of the test case. We do this by treating message parameters as intermediate variables and by deriving their values from the send event of a coordination message when the enabling condition of the corresponding receive event is checked. Later on we will also use this mechanism when normal receive events are executed. But, then we need to derive message parameter values from external information, e.g., test logs. We need no values for intermediate variables when we check the enabling condition of an event $te \notin ran(cm)$, i.e., te is not a send event of a coordination message. In this case IS is set to a tuple of unknown values, i.e., $(\propto_1, \ldots, \propto_l)$.

5.3 Execution of enabled test events

Based on the definition of an enabling condition we are able to define an execution rule for enabled test events.

Definition 5.4 (The Execution of an enabled test event) *Let TC be a TTCN test case and let $ST = (CS, DS)$ be a state of TC. Let te be a test event which is enabled in ST, i.e., $ST \xrightarrow{te}_{c,d}$. We assume that $IS \in D_1^{iv} \times \ldots \times D_l^{iv}$ is the tuple of values for intermediate variables which is calculated according to Condition 3. in Definition 4.3 and used in Condition 2. of Definition 4.3. The execution of te changes the actual state $ST = (CS, DS)$ into the new state $ST' = (CS', DS')$, where*

1. *CS' is determined by $CS \xrightarrow{te}_c CS'$ (cf. Definition 3.7), and*

2. *DS' is calculated by $DS \xrightarrow{te,IS}_d DS'$.*

We denote the execution of te by $ST \xrightarrow{te}_{c,d} ST'$.

The definition should be clear intuitively. The new state reached by the execution of a test event is calculated by the rules for the execution of control enabled and data enabled test events. It should be noted that for the calculation of IS the execution rule above refers to Definition 4.3. If we want to calculate IS in another way, we only need to modify the enabling condition.

Test Case Variable Declarations			
Variable Name	**Type**	**Value**	**Comments**
Count	Integer	1	Counter of a counter loop
S	Integer		

Figure 18: Declaration of TTCN test case variables

5.4 Traces, reachability sets, and global state graphs

In Section 3.5 we defined the notion of traces, reachability sets, and global state graphs already. All definitions were based on the control flow of a TTCN test case. We can modify these definitions slightly in order to make them applicable for complete TTCN test cases.

Definition 5.5 (Initial state of a TTCN test case) *Let TC be a TTCN test case and $ST^0 = (CS, DS)$ be a state of TC. ST^0 is called the initial state of TC, if*

1. *CS is the initial control state of TC (cf. Definition 3.8), and*
2. *DS is a tuple which holds the initial values of all state variables.*

The tuple DS in the definition above is defined informally. The initial values for the state variables have to be extracted from the declarations part of the test suite. For example, the declaration of the state variables for the test case in Figure 14 may look like the table in Figure 18. This leads to the initial data state $DS = (1, \propto, none)$, i.e., $Count$ is set to 1, S has no initial value, and the initial value of R is according to the TTCN definition $none$.

Definition 5.6 (Traces and reachability set of a TTCN test case) *Let TC be a TTCN test case, TE be the set of test events of TC, $w \in TE^*$ a finite word over TE, M_{TC} be the set of all possible states of TC, ST^0 the initial state of TC, and ST, ST', ST_1, ST_2, ..., ST_n states of TC.*

1. *The notation $ST \xrightarrow{w}_{c,d} ST'$ is defined by*
   (a) *$ST \xrightarrow{w}_{c,d} S$ if $w = \perp$*
   (b) *$ST \xrightarrow{w}_{c,d} ST'$ if*
   - *$w = t_1 \cdot t_2 \cdot \ldots \cdot t_n$ $(t_i \in TE)$ and*
   - *$ST \xrightarrow{t_1}_{c,d} ST_1 \xrightarrow{t_2}_{c,d} \ldots \xrightarrow{t_n}_{c,d} ST_n = ST'$*
2. *$R(TC) = \{ST \mid \exists w \in TE^* : ST^0 \xrightarrow{w}_{c,d} ST \wedge ST \in M_{TC}\}$ is the reachability set of TC, i.e. the set of states which can be reached from the initial state ST^0 when TC is executed.*
3. *$TR(TC) = \{w \mid \exists ST \in R(TC) : ST^0 \xrightarrow{w}_{c,d} ST \wedge w \in TE^*\}$ is the set of traces of TC.*

Definition 5.7 (Global state graph of a TTCN test case) *Let TC be a TTCN test case, ST^0 the initial state of TC, and $R(TC)$ be the reachability set of TC. The global state graph of TC is defined by $Er(TC) = (N, E, ST^0)$, where*

(a) $N = R(N)$ is the set of nodes,

(b) $E = \{(ST, t, ST') \mid ST, ST' \in R(N) \wedge ST \xrightarrow{t}_{c,d} ST'\}$ is the set of edges, and

(c) ST^0 is the start node of the graph.

According to the TTCN semantics a test run ends in a final state. Such a final state is a state in which a final test verdict is assigned, i.e., $PASS$, $FAIL$, or $INCONCLUSIVE$. This can be done explicitly within the verdict column of a TTCN behavior description, or implicitly, i.e., if no test event can be executed anymore the actual preliminary test verdict will become the final test verdict. We define the notion of final control states formally.

Definition 5.8 (Final states of a TTCN test case) *Let TC be a TTCN test case, TE be the set of test events, R the state variable which holds preliminary and final test verdicts during a test run, and $R(TC)$ be the reachability set of TC. The final states of TC are defined by the set:*

$$FS(TC) = \{ST \mid ST \in R(TC) \wedge \nexists\, te \in TE : ST \xrightarrow{te}_{c,d} \vee\ R \text{ holds a final test verdict}\}.$$

Based on the definition of the final states of a TTCN test case, we are able to define the notion of *complete test runs* of a TTCN test case.

Definition 5.9 (Complete test runs of a TTCN test case) *Let TC be a TTCN test case, TE be the set of test events, $R(TC)$ be the reachability set of TC, and ST^0 the initial state of TC. The complete test runs of TC are defined by the set:*

$$TR_{comp}(TC) = \{w \mid \exists ST \in FS(TC) : ST^0 \xrightarrow{w}_{c,d} ST \wedge w \in TE^*\}.$$

6 Two complete TTCN examples

In this section we introduce two small, but complete TTCN test case examples. Both examples should explain, how the provided model for the TTCN semantics works. We provide control structure, data structure, reachability set, and global state graph for both examples.

6.1 The TTCN test case DataExample

The first example is the TTCN test case *DataExample* which we already used for explaining the data structure of a TTCN test case (cf. Section 4.1). The information relevant for our purposes is described within the TTCN tables in Figure 19, 20, and 21.[4]

[4]Figure 19 and Figure 20 are identical to Figure 14 and Figure 18. For the sake of completeness and readability they are presented here again.

Test Case Dynamic Behaviour				
Test Case Name: DataExample				
Nr	Label	Behaviour Description	Constraints Ref	Verdict
1		?CONind		
2		!CONresp (Count:=1)		
3		REPEAT LOCAL UNTIL [Count=3]		
4		?DISind [DISind.err="Wrong ACK"]		
5		!DISresp		PASS
6		?DISind		
7		!DISresp		FAIL
		LOCAL		
8		?DAT (S:=DAT.SeNr, Count:=Count+1)		(PASS)
9		!AK (AK.SeNr:=S-1)		

Figure 19: Behavior description of the TTCN test case *DataExample*

Test Case Variable Declarations			
Variable Name	**Type**	**Value**	**Comments**
Count	Integer	1	Counter of a counter loop
S	Integer		

Figure 20: Declaration of state variables for the TTCN test case *DataExample*

The test case control structure of DataExample. The control structure for our example is described by a tuple $TCCS = (TE, t0, nte, attach, create, cm)$. The components of the tuple have the following meaning.

$$TE = \{?CONind_{Nr.1}, !CONresp_{Nr.2}, ?DISind_{Nr.4}, ?DISresp_{Nr.5}, REPEAT_{Nr.3}, UNTIL(1)_{Nr.3}, UNTIL(2)_{Nr.3}, ?DISind_{Nr.6}, ?DISresp_{Nr.7}, LOCAL_{top}, ?DAT_{Nr.8}, !AK_{Nr.9}\}$$

denotes the set of test events. TE includes some additional test events. These are the top node $LOCAL_{top}$ of the local tree $LOCAL$ and the events $UNTIL(1)_{Nr.3}$ and $UNTIL(2)_{Nr.3}$ which are necessary to model the control flow of the REPEAT loop correctly.

$$t0 = DataExample_{top}$$

is top node of the test case. The top node is no test event, i.e., $DataExample_{top} \notin TE$.

ASP Type Definition		
ASP Name: DISind (Disconnection Indication) **PCO Type:** **Comments:**		
Parameter Name	**Parameter Type**	**Comments**
err (Error Message)	CharString	Describes the kind of an error

ASP Type Definition		
ASP Name: DAT (Data) **PCO Type:** **Comments:**		
Parameter Name	**Parameter Type**	**Comments**
S (Sequence Number) Item (Data Item)	Integer InfoType	Sequence Number of data item Transported data

Figure 21: TTCN ASP definitions referred to within *DataExample*

nte $\subseteq TE \times TE$ is the *next-test-event* relation,

$attach$ $\subseteq TE \times TE$ is the *attach* relation,

$create$ $\subseteq TE \times TE$ is the create relation, and

cm $\subseteq TE \times TE$ is the *coordination-message* relation.

In Figure 22 the nte relation and the $attach$ relation are presented in a graphical form. The test case *DataExample* includes no parallel test components. Therefore there exist no $create$ and *coordination-message* relations, i.e., $create = \emptyset$ and $cm = \emptyset$.

Simulating the control flow of DataExample. Based on the control structure of *DataExample*, we are able to simulate the control flow of the test case. The initial control state $S0$ is

$$S0 = \quad (\{DataExample_{top}\}, \emptyset, \emptyset, \emptyset)$$

Applying the Definitions 3.6 and 3.7 repeatedly leads to the reachability set shown in Figure 23. The rightmost column of the table presents the test events which are enabled in a given control state. From the reachability set we can construct the global state graph shown in Figure 24.

The data structure of DataExample. The data structure of *DataExample* has been explained thoroughly in Example 4.1 on Page 30. For the sake of completeness we summarize this explanation. The data structure of *DataExample* is given by a tuple $TCDS = (TE; SV, IV; G, VT; \phi, \psi)$.

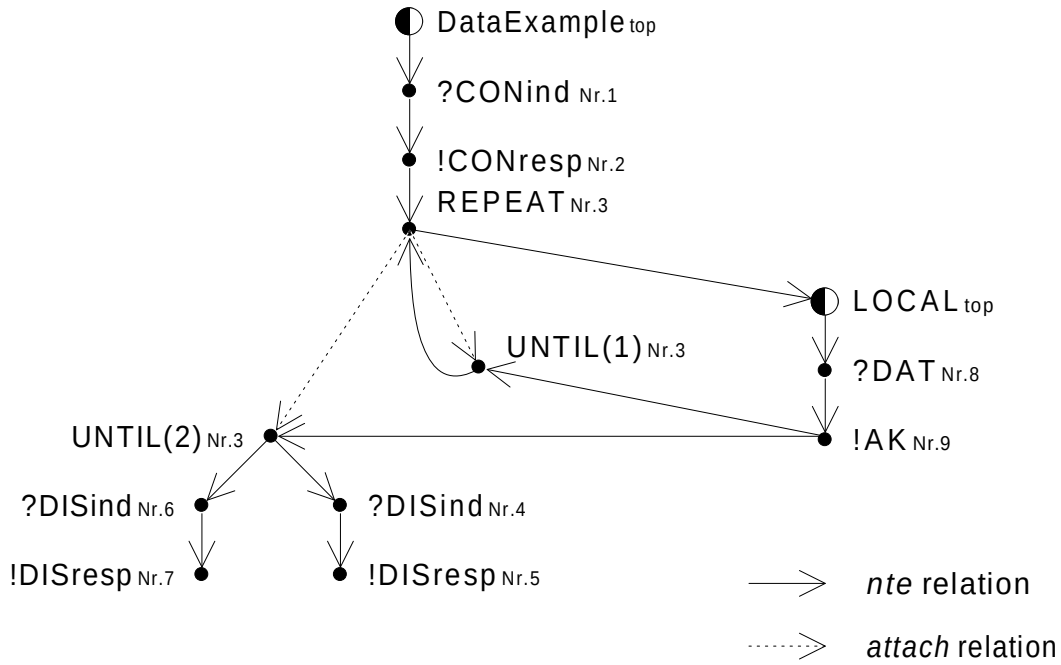


Figure 22: Control structure of the TTCN test case *DataExample*

State	Value	enabled events
S0	$(\{?DataExample_{top}\},\emptyset,\emptyset,\emptyset)$	$?CONind_{Nr.1}$
S1	$(\{?CONind_{Nr.1}\},\emptyset,\emptyset,\emptyset)$	$!CONresp_{Nr.2}$
S2	$(\{!CONresp_{Nr.2}\},\emptyset,\emptyset,\emptyset)$	$REPEAT_{Nr.3}$
S3	$(\{REPEAT_{Nr.3}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset)$	$?LOCAL_{top}$
S4	$(\{?LOCAL_{top}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset)$	$?DAT_{Nr.8}$
S5	$(\{?DAT_{Nr.8}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset)$	$!AK_{Nr.9}$
S6	$(\{!AK_{Nr.9}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset)$	$UNTIL(1)_{Nr.3}, UNTIL(2)_{Nr.3}$
S7	$(\{UNTIL(1)_{Nr.3}\},\emptyset,\emptyset,\emptyset)$	$REPEAT_{Nr.3}$
S8	$(\{UNTIL(2)_{Nr.3}\},\emptyset,\emptyset,\emptyset)$	$?DISind_{Nr.4}, ?DISind_{Nr.6}$
S9	$(\{?DISind_{Nr.4}\},\emptyset,\emptyset,\emptyset)$	$!DISresp_{Nr.5}$
S10	$(\emptyset,\emptyset,\emptyset,\emptyset)$	*none*
S11	$(\{?DISind_{Nr.6}\},\emptyset,\emptyset,\emptyset)$	$!DISresp_{Nr.7}$

Figure 23: Control flow based reachability set of *DataExample*

$$TE = \{?CONind_{Nr.1}, !CONresp_{Nr.2}, ?DISind_{Nr.4}, ?DISresp_{Nr.5}, REPEAT_{Nr.3}, UNTIL(1)_{Nr.3}, UNTIL(2)_{Nr.3}, ?DISind_{Nr.6}, ?DISresp_{Nr.7}, LOCAL_{top}, ?DAT_{Nr.8}, !AK_{Nr.9}\}$$

is the set of test events. It is the same set as the set of test events in the test case control structure explained in the previous paragraphs.

$$SV = \{Count{:}Integer, S{:}Integer, R{:}Verdicts\}$$

is the set of state variables. The set denotes all control variables declared in the declarations part of the test suite and variables declared implicitly, like in our case the variable R

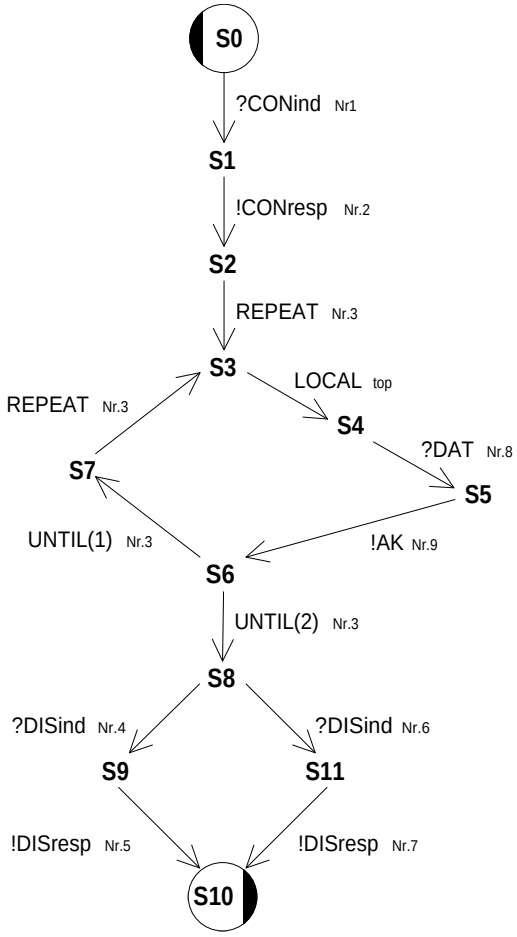


Figure 24: Control flow based global state graph of $DataExample$

which during a test run holds the actual test verdict. The state variables $Count$:$Integer$ and S:$Integer$ are extracted from the TTCN table in Figure 20.

$$IV = \{DISind.err\text{:}CharString, DAT.SeNr\text{:}Integer\}$$

denotes the set of intermediate variables. These variables are used to describe the influence of ASP, PDU, and coordination message parameters on the values of state variables during the test run. We extracted the names and types of intermediate variables from the TTCN tables in Figure 21.

$$G = \{true, Count{\neq}3, Count{=}3, DISind.err{=}"Wrong\ AK"\}$$

is the set of guards. Guards represent TTCN qualifiers. TTCN qualifiers are used to specify a data dependent enabling of test events. We extracted the guards above from the behavior description of $DataExample$ (cf. Figure 19).

$$VT = \{1, DAT.SeNr, Count{+}1, none, PASS, FAIL, (PASS), \propto, id(Count), id(S), id(R)\}$$

denotes the set of variable transitions. The expressions in VT describe how the state variables change their values during a test run. Variable transitions are gained from

test event (te)	$\phi(te)$	$\psi(te)$ tuples are meant to be assignments for: $(Count, S, R) \in Integer \times Integer \times Verdicts$
$?CONind_{Nr.1}$	$true$	$(id(Count), id(S), id(R))$
$!CONresp_{Nr.2}$	$true$	$(1, id(S), id(R))$
$REPEAT_{Nr.3}$	$true$	$(id(Count), id(S), id(R))$
$UNTIL(1)_{Nr.3}$	$Count \neq 3$	$(id(Count), id(S), id(R))$
$UNTIL(2)_{Nr.3}$	$Count=3$	$(id(Count), id(S), id(R))$
$?DISind_{Nr.4}$	$DISind.err=$"$Wrong\ AK$"	$(id(Count), id(S), id(R))$
$?DISresp_{Nr.5}$	$true$	$(id(Count), id(S), PASS)$
$?DISind_{Nr.6}$	$true$	$(id(Count), id(S), id(R))$
$?DISresp_{Nr.7}$	$true$	$(id(Count), id(S), FAIL)$
$LOCAL_{top}$	$true$	$(id(Count), id(S), id(R))$
$?DAT_{Nr.8}$	$true$	$(Count+1, DAT.SeNr, (PASS))$
$!AK_{Nr.9}$	$true$	$(id(Count), id(S), id(R))$

Figure 25: The functions ϕ and ψ of the data structure of $DataExample$

assignment operations within the behavior description of the test case.

The functions ϕ and ψ relate guards and variable transitions to test events. We define them in the table in Figure 25.

The simulation of DataExample. Based on the control structure and the data structure of $DataExample$ we are able to simulate the whole test case. Note, we need no additional intermediate data structure, because $DataExample$ includes no parallel test components. The simulation is done by applying enabling condition, i.e., Definition 4.3, and execution rule, i.e., Definition 4.4, repeatedly. For the start of the simulation procedure, we need an initial state $ST0$ (cf. Definition 5.5). This state is given by

$$ST0 = \quad ((\{?DataExample_{top}\}, \emptyset, \emptyset, \emptyset), (1, \propto, none))$$

Within the initial state, the initial value 1 of the state variable $Counter$ is extracted from its declaration in Figure 20. The initial value of the state variable S is undefined and the initial value of R is by default $none$.

By starting the simulation in $ST0$ the exploration of all states leads to the reachability set shown in Figure 26. The rightmost column shows which test events are enabled in a given state of the reachability set. Based on Figure 26 we can construct the global state graph shown in Figure 27.

By comparing the graphs in Figure 24 and Figure 27 we make some observations.

The global state graph which only is based on the control flow (Figure 24) includes a loop and consists of 13 nodes only. The loop may lead to infinite test runs. In contrast to this, the global state graph in Figure 27 has more nodes, i.e., 17, and includes no loops, i.e., only two traces are possible. The additional loop is due to the possible values of the state variable $Counter$. $Counter$ is the counter variable of the REPEAT loop. Therefore the loop in Figure 24 can be seen as an abstraction from the values of $Counter$.

Both graphs include nodes which represent final states, i.e., no test event is enabled

State	(Control-State,	Data-State)	enabled events
ST0	$((\{?DataExample_{top}\},\emptyset,\emptyset,\emptyset),$	$(1,\propto,none))$	$?CONind_{Nr.1}$
ST1	$((\{?CONind_{Nr.1}\},\emptyset,\emptyset,\emptyset),$	$(1,\propto,none))$	$!CONresp_{Nr.2}$
ST2	$((\{!CONresp_{Nr.2}\},\emptyset,\emptyset,\emptyset),$	$(1,\propto,none))$	$REPEAT_{Nr.3}$
ST3	$((\{REPEAT_{Nr.3}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(1,\propto,none))$	$?LOCAL_{top}$
ST4	$((\{?LOCAL_{top}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(1,\propto,none))$	$?DAT_{Nr.8}$
ST5	$((\{?DAT_{Nr.8}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(2,\propto,(PASS)))$	$!AK_{Nr.9}$
ST6	$((\{!AK_{Nr.9}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(2,\propto,(PASS)))$	$UNTIL(1)_{Nr.3}$
ST7	$((\{UNTIL(1)_{Nr.3}\},\emptyset,\emptyset,\emptyset),$	$(2,\propto,(PASS)))$	$REPEAT_{Nr.3}$
ST8	$((\{REPEAT_{Nr.3}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(2,\propto,(PASS)))$	$?LOCAL_{top}$
ST9	$((\{?LOCAL_{top}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(2,\propto,(PASS)))$	$?DAT_{Nr.8}$
ST10	$((\{?DAT_{Nr.8}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(3,\propto,(PASS)))$	$!AK_{Nr.9}$
ST11	$((\{!AK_{Nr.9}\},\{REPEAT_{Nr.3}\},\emptyset,\emptyset),$	$(3,\propto,(PASS)))$	$UNTIL(2)_{Nr.3}$
ST12	$((\{UNTIL(2)_{Nr.3}\},\emptyset,\emptyset,\emptyset),$	$(3,\propto,(PASS)))$	$?DISind_{Nr.4}$, $?DISind_{Nr.6}$
ST13	$((\{?DISind_{Nr.4}\},\emptyset,\emptyset,\emptyset),$	$(3,\propto,(PASS)))$	$!DISresp_{Nr.5}$
ST14	$((\emptyset,\emptyset,\emptyset,\emptyset),$	$(3,\propto,PASS))$	$none$
ST15	$((\{?DISind_{Nr.6}\},\emptyset,\emptyset,\emptyset),$	$(3,\propto,(PASS)))$	$!DISresp_{Nr.7}$
ST16	$((\emptyset,\emptyset,\emptyset,\emptyset),$	$(3,\propto,FAIL))$	$none$

Figure 26: Reachability set of the TTCN test case *DataExample*

in a final state. But, the graph in Figure 24 has one final state only, i.e., $S10$, whereas the graph in Figure 27 includes two final states. The reason for this is that the global state graph which is based on the control flow only does not consider the possible test verdicts, i.e., the values of the state variable R. Our test case may end with the final verdicts $PASS$ and $FAIL$. These possible verdicts are reflected by the two final states $ST14$ and $ST16$ in Figure 27.

6.2 A parallel TTCN test case based on Inres

The second example includes parallel test components. The system for which the test case is defined, is called Inres. Details about Inres can be found in [6, 7]. We intend to analyze a test case for testing the Initiator part of Inres.

A test architecture in which our test case is applicable is shown in Figure 28. The system under test (SUT) consists of the implementation under test (IUT), i.e., the Initiator process, and a Medium service which is used by the Initiator process in order to provide the Initiator part of the Inres service to its user. The tester processes are the lower tester LT and the upper tester UT. The main test component is the upper tester. LT and UT communicate via the coordination point $CP1$. For the communication with the SUT the tester processes use the PCOs ISAP and MSAP. The exchange of ASPs and PDUs which should be observed in order to obtain the test verdict $PASS$ is shown in the MSC in Figure 29.

The intention of the test case is the following. After connection establishment we send two data request messages $DATreq$. The second message is sent after the DT of the first $DATreq$ is received, but before it is acknowledged. The acknowledgement AK

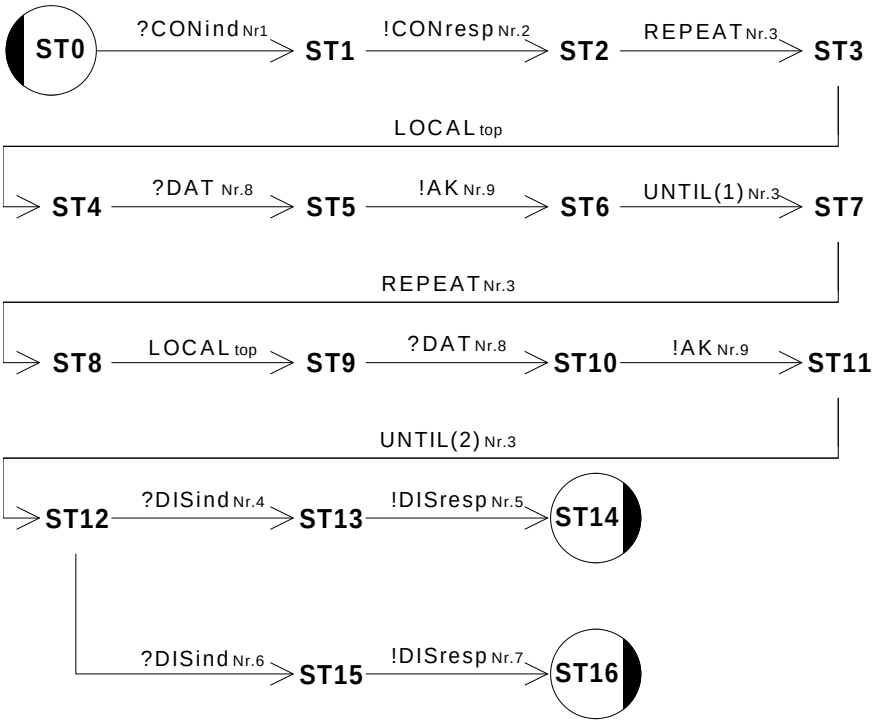


Figure 27: Global state graph of the TTCN test case *DataExample*

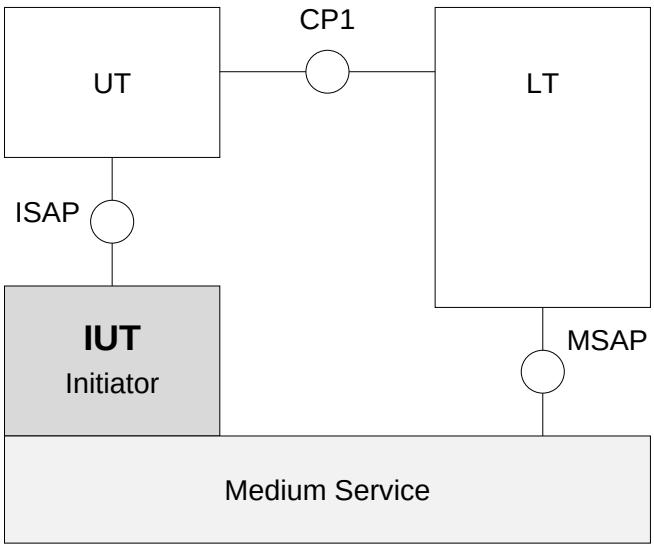


Figure 28: Test architecture

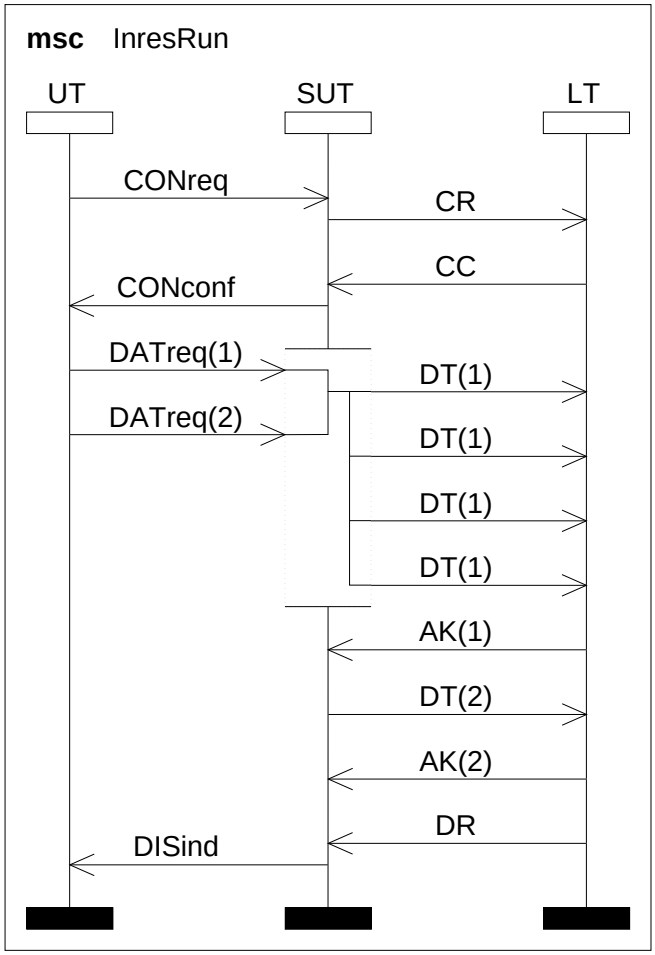


Figure 29: Message flow between LT, UT and SUT

is performed after the third retransmission of DT. After the acknowledgement we wait for the DT of the second $DATreq$ and check if it is transmitted correctly. The correct DT is acknowledged and a normal disconnection is performed. The relationship between $DATreq$, AK, and DT messages is indicated by the numbers *1* and *2* in the message parameters. They can be interpreted as sequence numbers of the $DATreq$ messages.

However, the MSC in Figure 29 does not describe the whole message flow which is performed during the test run. The UT as main test component has to create the LT and for synchronization purposes LT and UT have to exchange the coordination messages $RecDAT$, $SendDAT$, and $CorrDAT$.

Figure 30 describe the whole message flow including the creation of LT and the exchange of coordination messages. The coordination message $RecDat$ is sent by the LT after the reception of the first $DATreq$. The reception of $RecDat$ by the UT leads to the transmission of the second $DATreq$ which in return is indicated by the $SendDAT$ message to the LT. The coordination message $CorDAT$ acknowledges the reception of the correct DT for the second $DATreq$ message.

For defining a TTCN test case which includes parallel test components we need to specify a configuration. The TTCN table including the configuration for our test case

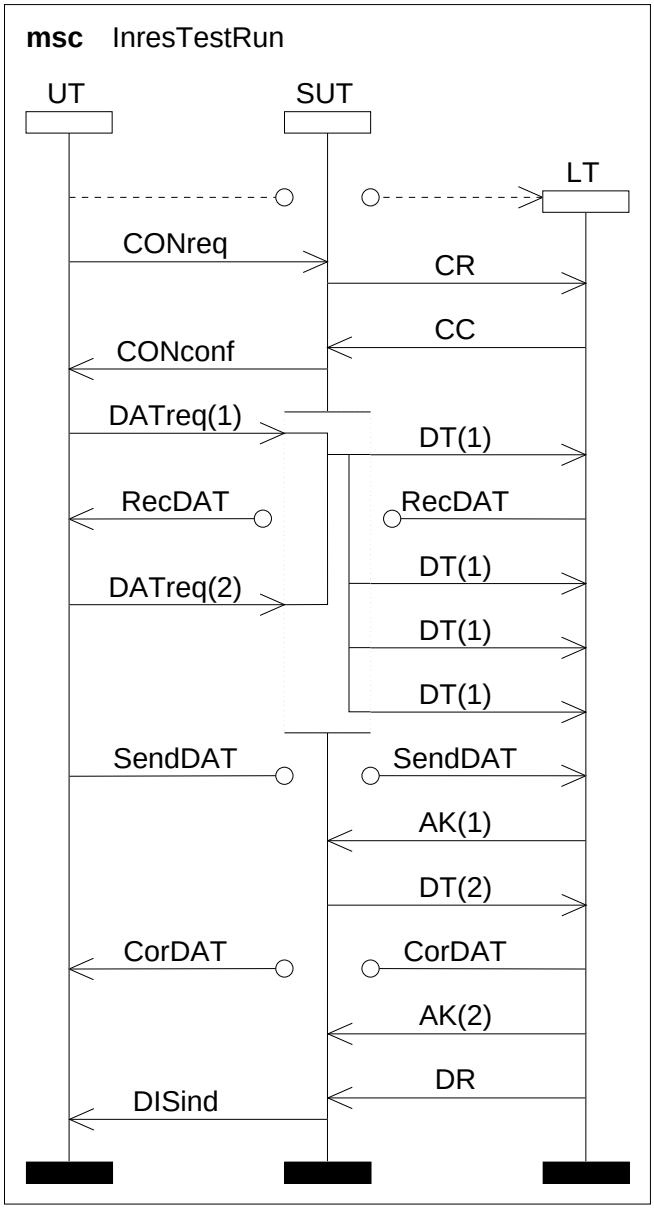


Figure 30: Message flow among LT and UT, and between LT, UT and SUT

example is shown in Figure 31. It describes the used test components (TCs), points of control and observation (PCOs), and coordination points (CPs). The coordination point declaration in Figure 32 defines the role of a coordination point. In our case $CP1$ is used for the communication between the main test component UT and the parallel test component LT.

The complete TTCN behavior description of the test case is presented in Figure 33 and Figure 34. The behavior lines 1 - 9 in Figure 33 and 1 - 12 in Figure 34 describe the behavior of UT and LT as shown in Figure 30. The other lines describe the handling of unexpected situations, i.e., if the SUT behaves not according to the test specification. All these cases will lead to $FAIL$ verdicts. Both behavior descriptions include no

Test Component Configuration Declarations			
Config Name: ConfigOne **Comments:**			
TCs Used	**PCOs Used**	**CPs Used**	**Comments**
UT LT	ISAP MSAP	CP1 CP1	

Figure 31: Configuration for the Inres test case example

CP Declarations		
CP Name	**CP Role**	**Comments**
CP1	UT <-> LT	

Figure 32: Coordination point declaration for the Inres test case example

REPEAT loops. The behavior description of the parallel test component include seven tree attachments. All attachments refer to the local tree *LOCAL*.

The test case control structure of the Inres test case example. The test case control structure of the example is given by $TCCS_{Inres} = (TE, t0, nte, attach, create, cm)$. The components of the tuple have the following meaning.

TE is the set of test events.

For our example there exist 46 test events. They are listed in the *Event Description* columns of the tables in the Figures 35 and 36. In order to facilitate the following descriptions we introduce abbreviations for the different test events. These abbreviations can be found in the *Short Id* columns within the Figures 35 and 36.

$t0 =$ MTC

is top node of the test case. The top node is no test event, i.e., $MTC \notin TE$.

nte $\subseteq TE \times TE$ is the *next-test-event* relation,

$attach$ $\subseteq TE \times TE$ is the *attach* relation,

$create$ $\subseteq TE \times TE$ is the *create* relation, and

cm $\subseteq TE \times TE$ is the *coordination-message* relation.

In Figure 22 the *next-test-event*, the *attach*, the *create*, and the *coordination-message* relation are presented in a graphical form. Although the TTCN test case descriptions looks quite simple the graphical form of the test case control structure is a complex graph with 4 sorts of edges. It should be noted that the graph includes no loops. Since we disallow recursive tree attachment the simulation of the test case will lead to final global state graphs without loops, i.e., all traces are finite.

Test Case Dynamic Behaviour				
Test Case Name: InresTestCaseExample **Configuration:** ConfigOne **Comments:** This is the main test component				
Nr	Label	Behaviour Description	Constraints Ref	Verdict
1		CREATE(LowerTester,PTCDescription)		
2		ISAP!CONreq		
3		ISAP?CONconf		
4		ISAP!DATreq (DATreq.S := 1)	DATreqdef	
5		CP1?RecDAT		
6		ISAP!DATreq (DATreq.S := 2)	DATreqdef	
7		CP1!SendDAT (SendDAT.S := 2)	SendDATdef	
8		CP1?CorDAT		(P)
9		ISAP?DISind		
10		CP1?FailDAT		(F)
11		ISAP?DISind		
12		ISAP?DISind		F
13		CP1?FailDAT		(F)
14		ISAP?DISind		
15		ISAP?DISind		F

Figure 33: TTCN behavior description of the main test component

Simulating the control flow of the Inres test case example. Based on the test case control structure the control flow aspects of the Inres test case example can be simulated. Like in the previous example (cf. Page 44) this is done by applying the definitions 3.6 and 3.7 repeatedly. The initial control state is

$$S0 = \quad (\{MTC\}, \emptyset, \emptyset, \{\perp, \perp\}).$$

The set $\{\perp, \perp\}$ describes the empty queues through which the main test component and the parallel test component will exchange coordination messages. In the following we assume that the first element of the coordination message queue parts of all following control states describe the queue in which the main test component sends coordination messages, i.e., through which the parallel test component receives coordination messages. The second element of the set describes the queue for the other direction.

By simulating the control structure we gain a reachability set which consists of 374 control states. The global state graph will have 374 nodes and 713 state transitions. There are 5 end states and there are 488272 different traces leading from the initial state to an end state.

The global state graph cannot be presented here graphically. But, for the sake of completeness the reachability set and all possible state transitions are listed at the end of this report (cf. Figures 40-47 and 48-51).

The data structure of the Inres test case example. The data structure of our example test case is given by a tuple $TCDS_{Inres} = (TE; SV, IV; G, VT; \phi, \psi)$. The components have the following meaning.

Test Step Dynamic Behaviour				
Test Step Name: PTCDescription				
Nr	Label	Behaviour Description	Constraints Ref	Verdict
1		MSAP?CR		
2		MSAP!CC		
3		MSAP?DT (S := DT.S)	DTdef	
4		CP1!RecDAT		
5		MSAP?DT [DT.S = S]	DTdef	
6		MSAP?DT [DT.S = S]	DTdef	
7		MSAP?DT [DT.S = S]	DTdef	
8		CP1?SendDAT (SuccS := SendDAT.S)	SendDATdef	
9		MSAP!AK (AK.Nr := S)	AKdef	
10		MSAP?DT [DT.S = SuccS]	DTdef	
11		CP1!CorDAT		
12		MSAP!DR		
13		+LOCAL		
14		+LOCAL		
15		+LOCAL		
16		+LOCAL		
17		+LOCAL		
18		+LOCAL		
19		+LOCAL		
		LOCAL		
20		MSAP?OTHERWISE		
21		CP1!FailDAT		
22		MSAP!DR		

Figure 34: TTCN behavior description of the parallel test component

Short Id	**Line Nr. in TTCN**	**Event Description**	**Short Id**	**Line Nr. in TTCN**	**Event Description**
M1	1	CREATE	M9	9	ISAP?DISind
M2	2	ISAP!CONreq	M10	10	CP1?FailDAT
M3	3	ISAP?CONconf	M11	11	ISAP?DISind
M4	4	ISAP!DATreq	M12	12	ISAP?DISind
M5	5	CP1?RecDAT	M13	13	CP1?FailDAT
M6	6	ISAP!DATreq	M14	14	ISAP?DISind
M7	7	CP1!SendDAT	M15	15	ISAP?DISind
M8	8	CP1?CorDAT			

Figure 35: Test events related to the main test component

Node Id	Line Nr. in TTCN	Event Description	Node Id	Line Nr. in TTCN	Event Description
PTC		*Top node of PTC*	P14(2)	14	+LOCAL
P1	1	MSAP?CR	P15(1)	15	+LOCAL
P2	2	MSAP!CC	P15(2)	15	+LOCAL
P3	3	MSAP?DT	P16(1)	16	+LOCAL
P4	4	CP1!RecDAT	P16(2)	16	+LOCAL
P5	5	MSAP?DT	P17(1)	17	+LOCAL
P6	6	MSAP?DT	P17(2)	17	+LOCAL
P7	7	MSAP?DT	P18(1)	18	+LOCAL
P8	8	CP1?SendDAT	P18(2)	18	+LOCAL
P9	9	MSAP!AK	P19(1)	19	+LOCAL
P10	10	MSAP?DT	P19(2)	19	+LOCAL
P11	11	CP1!CorDAT	TopL		LOCAL
P12	12	MSAP!DR	P20	20	MSAP?OTHERWISE
P13(1)	13	+LOCAL	P21	21	CP1!FailDAT
P13(2)	13	+LOCAL	P22	22	MSAP!DR
P14(1)	14	+LOCAL			

Figure 36: Test events related to the parallel test component

TE is the set of test events.

It is the same set as used for the test case control structure, i.e., for our example they can be found in the Figures 35 and 36.

$SV = \{S{:}Integer, SuccS{:}Integer, R{:}Verdicts\}$

is the set of state variables. We have only three state variables. The variables S and $SuccS$ are used in the parallel test component only. In Figure 38 they are declared as test component variables. The variable R is defined implicitly as the variable which holds the actual test verdict during a test run.

$IV = \{SendDAT.S{:}Integer, DT.S{:}Integer\}$

is the set of intermediate variables. They are declared as parameters of ASPs, PDUs, or coordinations messages in the same manner as described in the previous example. The concrete TTCN definitions for the Inres test case example are not presented here.

$G = \{true, DT.S{=}S\}$

is the set of guards. It should be noted that during simulation the expression $DT.S{=}S$ always evaluates to $true$. This is due to the fact, that $DT.S$ is an intermediate variable. It represents a parameter value of the ASP DT which is received from the IUT during the test run. We assume that we have no knowledge about the IUT. Therefore the value of $DT.S$ is unknown.

$VT = \{DT.S, SendDat.S, fin(R), id(S), id(SuccS), id(R)\}$

denotes the set of variable transitions. The expressions in VT describe how the state variables change their values during a test run. Variable transitions are gained from

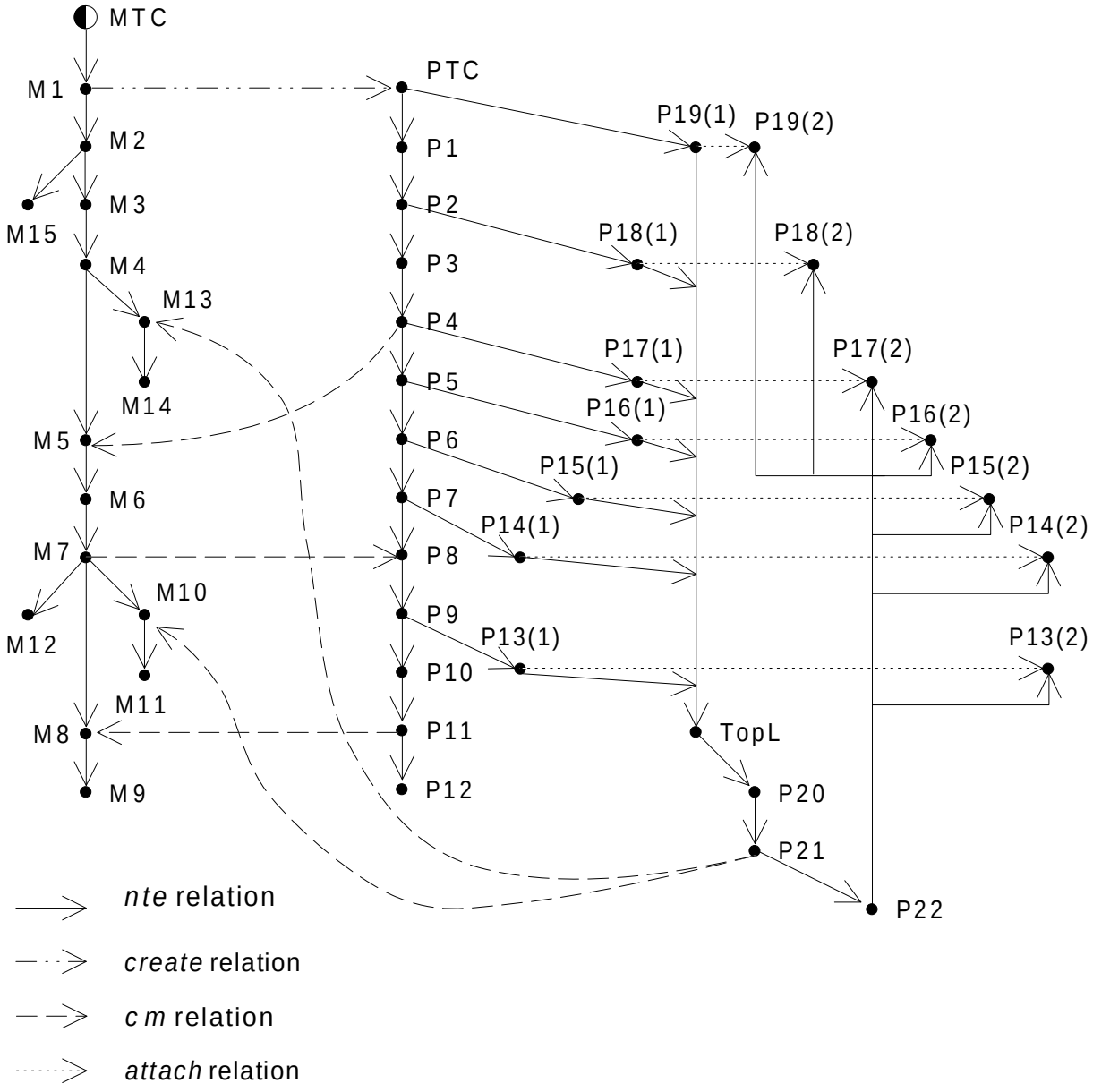


Figure 37: Test case control structure of the Inres test case example

Test Component Variable Declarations			
Variable Name	**Type**	**Value**	**Comments**
S	Integer		
SuccS	Integer		

Figure 38: Variable declarations for the Inres test case example

event te	$\phi(te)$	$\psi(te)$ $(S, SuccS, R)$	event te	$\phi(te)$	$\psi(te)$ $(S, SuccS, R)$
M1	$true$	$(id(S), id(SuccS), id(R))$	M9	$true$	$(id(S), id(SuccS), fin(R))$
M2	$true$	$(id(S), id(SuccS), id(R))$	M10	$true$	$(id(S), id(SuccS), (F))$
M3	$true$	$(id(S), id(SuccS), id(R))$	M11	$true$	$(id(S), id(SuccS), fin(R))$
M4	$true$	$(id(S), id(SuccS), id(R))$	M12	$true$	$(id(S), id(SuccS), F)$
M5	$true$	$(id(S), id(SuccS), id(R))$	M13	$true$	$(id(S), id(SuccS), (F))$
M6	$true$	$(id(S), id(SuccS), id(R))$	M14	$true$	$(id(S), id(SuccS), fin(R))$
M7	$true$	$(id(S), id(SuccS), id(R))$	M15	$true$	$(id(S), id(SuccS), F)$
M8	$true$	$(id(S), id(SuccS), (P))$			
PTC	$true$	$(id(S), id(SuccS), id(R))$	P14(2)	$true$	$(id(S), id(SuccS), id(R))$
P1	$true$	$(id(S), id(SuccS), id(R))$	P15(1)	$true$	$(id(S), id(SuccS), id(R))$
P2	$true$	$(id(S), id(SuccS), id(R))$	P15(2)	$true$	$(id(S), id(SuccS), id(R))$
P3	$true$	$(DT.S, id(SuccS), id(R))$	P16(1)	$true$	$(id(S), id(SuccS), id(R))$
P4	$true$	$(id(S), id(SuccS), id(R))$	P16(2)	$true$	$(id(S), id(SuccS), id(R))$
P5	$DT.S{=}S$ (always $true$)	$(id(S), id(SuccS), id(R))$	P17(1)	$true$	$(id(S), id(SuccS), id(R))$
P6	$DT.S{=}S$ (always $true$)	$(id(S), id(SuccS), id(R))$	P17(2)	$true$	$(id(S), id(SuccS), id(R))$
P7	$DT.S{=}S$ (always $true$)	$(id(S), id(SuccS), id(R))$	P18(1)	$true$	$(id(S), id(SuccS), id(R))$
P8	$true$	$(id(S), SendDAT.S, id(R))$	P18(2)	$true$	$(id(S), id(SuccS), id(R))$
P9	$true$	$(id(S), id(SuccS), id(R))$	P19(1)	$true$	$(id(S), id(SuccS), id(R))$
P10	$DT.S{=}SuccS$ (always $true$)	$(id(S), id(SuccS), id(R))$	P19(2)	$true$	$(id(S), id(SuccS), id(R))$
P11	$true$	$(id(S), id(SuccS), id(R))$	TopL	$true$	$(id(S), id(SuccS), id(R))$
P12	$true$	$(id(S), id(SuccS), id(R))$	P20	$true$	$(id(S), id(SuccS), id(R))$
P13(1)	$true$	$(id(S), id(SuccS), id(R))$	P21	$true$	$(id(S), id(SuccS), id(R))$
P13(2)	$true$	$(id(S), id(SuccS), id(R))$	P22	$true$	$(id(S), id(SuccS), id(R))$
P14(1)	$true$	$(id(S), id(SuccS), id(R))$			

Figure 39: The functions ϕ and ψ for the data structure of the Inres test case example

assignment operations within the behavior description of the test case. Only the variable transition $fin(R)$ may need an additional explanation. The TTCN standard states that the actual value of R will be final test verdict if the test case ends without an explicit assignment of a final test verdict. In our case the main test component may end in several situations where a preliminary $PASS$ or $FAIL$ is assigned. The variable transition $fin(R)$ states explicitly that based on the actual value of R a final test verdict should be calculated. Later on we will describe the simulation of the Inres test case example. The simulation will always be terminated when a final test verdict is assigned. This criterion is independent from the actual state of the parallel test component.

The functions ϕ and ψ relate guards and variable transitions to test events. We define them in the table in Figure 39.

An intermediate data structure for the Inres test case example. Our test case example includes parallel test components which communicate by exchanging coordination messages at coordination points. The information which is contained in message parameters may influence the simulation run. Therefore it has to be considered. We do this by treating message parameters as intermediate variables and by deriving their values from the send event of a coordination message when the enabling condition of the corresponding receive event is checked and executed. However, we have to provide an intermediate data structure for this purpose. It is given by a tuple $IDS_{Inres} = (E; EV, IV; IA, \theta)$. The tuple components have the following meaning.

$E = \{CP1!SendDAT, CP1!RecDAT, CP1!CorDAT, CP1!FailDAT\}$

is the set of events. E is a subset of the test events of the test case control structure and test case data structure. The elements of E denote the send events of coordination messages. In the following we use the short identifiers of these events mainly (cf. Figures 35 and 36), i.e., $E = \{M7, P4, P11, P21\}$.

$EV = \{S{:}Integer, SuccS{:}Integer, R{:}Verdicts\}$ is the set of external variables. In our case it is identical to the set of state variables of the Inres test case example.

$IV = \{SendDAT.S{:}Integer, DT.S{:}Integer\}$

is the set of intermediate variables. IV is identical to the set of intermediate variables which is used in the data structure of the test case.

$IA = \{1, \propto\}$

is the set of intermediate assignments. The expressions in IV describe how values are assigned to intermediate variables.

$\theta : \quad E \rightarrow IA^{\#(IV)}$

relates a tuple $(ia_1, \ldots, ia_{\#(IV)})$ of expressions of IA to each event of E. For our example *theta* is defined by:

$\theta(M7) := \quad (2, \propto)$

$\theta(P4) := \quad (\propto, \propto)$

$\theta(P11) := \quad (\propto, \propto)$

$\theta(P21) := \quad (\propto, \propto)$

In cases where an intermediate variable is unused, e.g., $SendDAT.S$ and $DT.S$ for $P4$, $P11$, and $P21$, we assign the expression unknown, i.e., $\propto$, to the corresponding tuple element.

For our example, the function $eval_{ids}$ (cf. Definition 4.6) is identical to θ for all events in E. This is due to the fact that external variables are not used in the expressions of IA.

The simulation of the Inres test case example. Based on the test case control structure $TCCS_{Inres}$, the test case data structure $TCDS_{Inres}$, and the intermediate data structure IDS_{Inres} it is possible to simulate the complete test case. The simulation should start in the initial state

$$ST0 = \quad (\{MTC\}, \emptyset, \emptyset, \{\perp, \perp\}), (\propto, \propto, none)).$$

By applying the definitions 4.3 and 4.4 repeatedly we obtain a reachabilty set with 310 states. 33 states are final states. Compared with the results of the control flow simulation we obtain 28 more final states. This due to the fact that we terminate the simulation always when a final test verdict is calculated, independently from the actual (control) state of the parallel test component. The additional final states reflect the different states in which the parallel test component may end.

The construction of the global state graph will lead to a graph with 310 nodes and 608 arrows, i.e., state transitions. This graph cannot be presented graphically. But, for the sake of completeness the reachabililty set and the state transitions are listed at the end of this section (cf. Figures 52-58 and 59-63).

The graph is finite and includes no loops. We can distinguish 478406 traces leading from the initial state to a final state. 4767 traces end with a $PASS$ verdict and 473639 traces with a $FAIL$ verdict.

We did not provide a complete example for the execution of an enabled test event (cf. Section 5.3). Especially, the way of how the intermediate data structure works. Therefore we look at one state transition in more detail. In state

$$ST204 = \quad (\{M7, P7\}, \emptyset, \emptyset, \{M7, \perp\}), (\propto, \propto, none))$$

the events $M12$, $P8$, and $P14(1)$ are enabled (cf. Figure 56). The execution of $P8$ changes $ST204$ to

$$ST227 = \quad (\{M7, P8\}, \emptyset, \emptyset, \{\perp, \perp\}), (\propto, 2, none)).$$

The calculation of $CS_{ST227} = (\{M7, P8\}, \emptyset, \emptyset, \{\perp, \perp\})$ follows the explanation in Section 3.4. The calculation starts from the control state $CS_{ST204} = (\{M7, P7\}, \emptyset, \emptyset, \{M7, \perp\})$.

$$S_{nte}^{ST227} = \quad \{M7, P7\} \backslash P7 \cup P8 = \{M7, P8\}$$
$$S_{attach}^{ST227} = \quad \emptyset \backslash \emptyset \cup \emptyset = \emptyset$$
$$S_{create}^{ST227} = \quad \emptyset \backslash \emptyset \cup \emptyset = \emptyset$$
$$S_{cm}^{ST227} = \quad \{dequeue(tequeue(P8)), \perp\} = \{\perp, \perp\}$$

For the calculation of the data state $DS_{ST227} = (\propto, 2, none))$ the intermediate data structure IDS_{Inres} and the function $eval_{ids}$ are relevant. The calculation of DS_{ST227} starts from $DS_{ST204} = (\propto, \propto, none))$. The tuple IS which is needed for the execution of $P7$, i.e., $DS_{ST204} \stackrel{P8,IS}{\longrightarrow}_d DS_{ST227}$, is calculated by

$$IS = eval_{ids}(next(qstate(tequeue(P8))), DS_{ST204}) = eval_{ids}(M7, (\propto, \propto, none)) = (2, \propto)$$

The details concerning the function $eval_{ids}$ for our example have been explained in the previous paragraph. According to Definition 4.4 we need 3 functions f_{P8}^{S}, f_{P8}^{SuccS}, and f_{P8}^{R} for the calculation of $DS_{ST227} = (d_1, d_2, d_3) = (\propto, 2, none))$. The functions include the variable transitions within the test case data structure $TCDS_{Inres}$ which are related to $P8$. They are defined by

$$f^{S}_{P8}(S, SuccS, R, SendDAT.S, DT.S) = \quad id(S)$$
$$f^{SuccS}_{P8}(S, SuccS, R, SendDAT.S, DT.S) = \quad SendDAT.S$$
$$f^{R}_{P8}(S, SuccS, R, SendDAT.S, DT.S) = \quad id(R)$$

Applying these functions leads to

$$d_1 = f^{S}_{P8}(DS_{ST204}, IS) = f^{S}_{P8}((\propto, \propto, none), (2, \propto)) = \propto$$
$$d_1 = f^{SuccS}_{P8}(DS_{ST204}, IS) = f^{S}_{P8}((\propto, \propto, none), (2, \propto)) = 2$$
$$d_3 = f^{R}_{P8}(DS_{ST204}, IS) = f^{S}_{P8}((\propto, \propto, none), (2, \propto)) = none$$

7 Simulation and Complexity

Within the previous sections we only provided the definitions on which the implementation of an TTCN simulator can be based. We discussed some problems, but did not prove any property of the developed model.

However, the last example (cf. Section 6.2) presents the effect of one property in an intuitive manner, it is the complexity. In general, the behavior tree[5] of a TTCN test case with parallel test components grows exponentially.[6]

Dealing with behavior descriptions which grow exponentially is also problem for test case generation based on formal descriptions, i.e., system which are described by using for example LOTOS[9] or SDL [14]. At the University of Berne we tackled the problem for SDL by using *heuristics*, *partial order simulation methods*, and *optimization strategies*. Our discussions on this topic are summarized in [21] and [5]. In principle the techniques presented in both papers can be adapted to our needs, i.e., be used for dealing with the complexity of TTCN descriptions.

8 Comparison with other approaches

There are some other approaches which provide a formal semantics for TTCN or at least the possibility for simulating TTCN test cases. We would like to describe 3 approaches briefly.

- Direct generation of executable code from TTCN descriptions.
- Translation of TTCN into SDL.
- Common semantical representation for SDL and TTCN.

[5] A behavior tree can be seen as a mapping of the global state graph onto a, possibly infinite, tree structure.

[6] We provide no formal proof of this fact. But, the formal proof can be done by relating our approach to Petri nets. The generation of the behavior tree for TTCN can be related to the reachability problem for Petri nets which is NP-complete.

8.1 Direct generation of executable code from TTCN

Some modern TTCN tools like ITEX 3.0 [20] provide the possibility to generate executable C code directly from TTCN descriptions. It is also possible to generate some sort of TTCN simulator.

However, we received the new ITEX 3.0 version when we already had finished the presented work and the implementation of our TTCN simulator. Therefore, we could not check if the features of, for example, ITEX 3.0 may also be suitable for solving some of our problems. But, for the sake of completeness we would like to mention that there exist commercial tools which provide simulation facilities. It should also be mentioned that these commercial tools provide no complete formal model on which their simulation is based. The tool developers interpreted the informal TTCN standard and implemented their understanding of the standard.

8.2 The translation of TTCN into SDL

In [22] concurrent TTCN test cases are transformed into SDL [14]. The aim of this procedure is to validate TTCN test cases by using validation techniques for SDL specifications.

The problem of such an approach is the mapping of TTCN constructs onto SDL constructs. TTCN and SDL have been developed for different purposes and therefore an adequate modeling of TTCN concepts by using SDL constructs is not always possible or trivial. For example, in TTCN a parallel test component may be connected with several queues each of which represent a coordination point (CP) or a point of control and observation (PCO). Contrary to this, in SDL each process is connected to one queue only. Subsequently, it is not trivial to model a parallel test component by means of an SDL process. At least the possible interleaving of ASPs, PDUs, and coordination messages waiting in different CPs and PCOs have to be considered. To avoid such problems, we decided to base our simulation directly on TTCN.

8.3 A common semantical representation for SDL and TTCN

Over recent years, the European Telecommunications Standards Institute has recognized that an integrated methodology has to be established covering all aspects of protocol engineering. As part of this broader context is has been considered reasonable to define a semantical relationship between SDL and TTCN.

The term *common semantical representation* refers to a representation, able to represent the (formal) semantics of objects from different domains in a common model. Such a common model enables the investigation of semantical relations between objects of the different domains. Hence, a common semantical representation for SDL and TTCN enables the definition of formal semantical relations between SDL and TTCN specifications. Such relations may act as a basis when developing a theory and tools for test generation, test validation, etc.

In [23, 24] a common semantical representation is described that is an operational model consistent with the semantics defined in ITU Recommendation Z.100 for SDL and in ISO 9646 Part 3 for TTCN. In [25] an extension to the common semantical representation is described so that concurrent TTCN is also covered.

The main characteristic of the common semantical representation is that it is a compositional hierarchical model, which enables reasoning about the dynamic behavior of SDL and concurrent TTCN specifications at different levels of observability. The common semantical representation structures an SDL or concurrent TTCN specification into a set of hierarchically ordered components. The dynamic semantics of each component is defined in terms of the dynamic semantics of the components at the next lower level (if the next lower level exists) or directly for all basic components that do not have lower levels. The semantics of a components is defined by a labeled transition system as described in [19]. The approach has been proven convenient for the definition of compositional structures and allows for formal reasoning about temporal properties.

From a more general point of view our automata based model and the basis of the common semantical representation are identical. The problem which we had with the on the common semantical representation is that the semantics cannot be implemented directly and efficiently, i.e., it is not convenient for simulation purposes. Therefore, we preferred to develop a model which can be implemented directly, i.e., the test case control structure is implemented as a graph and the data structure is realized in form of a set of functions which are associated with the nodes of the graph.

9 Summary

We presented a model for the semantics of TTCN. It should serve as the base for the implementation of a TTCN simulator and, therefore, the definitions are close to implementation. The model is able to deal with most TTCN constructs, including parallel TTCN. We implemented the model and started to experiment with the TTCN simulator. The first trials showed that the simulator is suitable for our purposes, i.e., test case visualization and test result analysis. In a further step we intend to include mechanisms for dealing with the complexity of TTCN descriptions, i.e., heuristics and partial order simulation methods.

State	Value	enabled events
S0	({MTC},∅,∅,{⊥,⊥})	M1
S1	({M1},∅,M1,{⊥,⊥})	M2,PTC
S2	({M2},∅,M1,{⊥,⊥})	M3,M15,PTC
S3	({M1,PTC},∅,∅,{⊥,⊥})	M2,P1,P19(1)
S4	({M3},∅,M1,{⊥,⊥})	M4,PTC
S5	(∅,∅,M1,{⊥,⊥})	PTC
S6	({M2,PTC},∅,∅,{⊥,⊥})	M3,M15,P1,P19(1)
S7	({M1,P1},∅,∅,{⊥,⊥})	M2,P2
S8	({M1,P19(1)},P19(1),∅,{⊥,⊥})	M2,TopL
S9	({M4},∅,M1,{⊥,⊥})	PTC
S10	({M3,PTC},∅,∅,{⊥,⊥})	M4,P1,P19(1)
S11	({0,PTC},∅,∅,{⊥,⊥})	P1,P19(1)
S12	({M2,P1},∅,∅,{⊥,⊥})	M3,M15,P2
S13	({M2,P19(1)},P19(1),∅,{⊥,⊥})	M3,M15,TopL
S14	({M1,P2},∅,∅,{⊥,⊥})	M2,P3,P18(1)
S15	({M1,TopL},P19(1),∅,{⊥,⊥})	M2,P20
S16	({M4,PTC},∅,∅,{⊥,⊥})	P1,P19(1)
S17	({M3,P1},∅,∅,{⊥,⊥})	M4,P2
S18	({M3,P19(1)},P19(1),∅,{⊥,⊥})	M4,TopL
S19	({0,P1},∅,∅,{⊥,⊥})	P2
S20	({0,P19(1)},P19(1),∅,{⊥,⊥})	TopL
S21	({M2,P2},∅,∅,{⊥,⊥})	M3,M15,P3,P18(1)
S22	({M2,TopL},P19(1),∅,{⊥,⊥})	M3,M15,P20
S23	({M1,P3},∅,∅,{⊥,⊥})	M2,P4
S24	({M1,P18(1)},P18(1),∅,{⊥,⊥})	M2,TopL
S25	({M1,P20},P19(1),∅,{⊥,⊥})	M2,P21
S26	({M4,P1},∅,∅,{⊥,⊥})	P2
S27	({M4,P19(1)},P19(1),∅,{⊥,⊥})	TopL
S28	({M3,P2},∅,∅,{⊥,⊥})	M4,P3,P18(1)
S29	({M3,TopL},P19(1),∅,{⊥,⊥})	M4,P20
S30	({0,P2},∅,∅,{⊥,⊥})	P3,P18(1)
S31	({0,TopL},P19(1),∅,{⊥,⊥})	P20
S32	({M2,P3},∅,∅,{⊥,⊥})	M3,M15,P4
S33	({M2,P18(1)},P18(1),∅,{⊥,⊥})	M3,M15,TopL
S34	({M2,P20},P19(1),∅,{⊥,⊥})	M3,M15,P21
S35	({M1,P4},∅,∅,{⊥,P4})	M2,P5,P17(1)
S36	({M1,TopL},P18(1),∅,{⊥,⊥})	M2,P20
S37	({M1,P21},P19(1),∅,{⊥,P21})	M2,P22
S38	({M4,P2},∅,∅,{⊥,⊥})	P3,P18(1)
S39	({M4,TopL},P19(1),∅,{⊥,⊥})	P20
S40	({M3,P3},∅,∅,{⊥,⊥})	M4,P4
S41	({M3,P18(1)},P18(1),∅,{⊥,⊥})	M4,TopL
S42	({M3,P20},P19(1),∅,{⊥,⊥})	M4,P21
S43	({0,P3},∅,∅,{⊥,⊥})	P4
S44	({0,P18(1)},P18(1),∅,{⊥,⊥})	TopL
S45	({0,P20},P19(1),∅,{⊥,⊥})	P21
S46	({M2,P4},∅,∅,{⊥,P4})	M3,M15,P5,P17(1)

Figure 40: Part I of control states for the Inres test case example

State	Value	enabled events
S47	({M2,TopL},P18(1),∅,{⊥,⊥})	M3,M15,P20
S48	({M2,P21},P19(1),∅,{⊥,P21})	M3,M15,P22
S49	({M1,P5},∅,∅,{⊥,P4})	M2,P6,P16(1)
S50	({M1,P17(1)},P17(1),∅,{⊥,P4})	M2,TopL
S51	({M1,P20},P18(1),∅,{⊥,⊥})	M2,P21
S52	({M1,P22},P19(1),∅,{⊥,P21})	M2,P19(2)
S53	({M4,P3},∅,∅,{⊥,⊥})	P4
S54	({M4,P18(1)},P18(1),∅,{⊥,⊥})	TopL
S55	({M4,P20},P19(1),∅,{⊥,⊥})	P21
S56	({M3,P4},∅,∅,{⊥,P4})	M4,P5,P17(1)
S57	({M3,TopL},P18(1),∅,{⊥,⊥})	M4,P20
S58	({M3,P21},P19(1),∅,{⊥,P21})	M4,P22
S59	({0,P4},∅,∅,{⊥,P4})	P5,P17(1)
S60	({0,TopL},P18(1),∅,{⊥,⊥})	P20
S61	({0,P21},P19(1),∅,{⊥,P21})	P22
S62	({M2,P5},∅,∅,{⊥,P4})	M3,M15,P6,P16(1)
S63	({M2,P17(1)},P17(1),∅,{⊥,P4})	M3,M15,TopL
S64	({M2,P20},P18(1),∅,{⊥,⊥})	M3,M15,P21
S65	({M2,P22},P19(1),∅,{⊥,P21})	M3,M15,P19(2)
S66	({M1,P6},∅,∅,{⊥,P4})	M2,P7,P15(2)
S67	({M1,P16(1)},P16(1),∅,{⊥,P4})	M2,TopL
S68	({M1,TopL},P17(1),∅,{⊥,P4})	M2,P20
S69	({M1,P21},P18(1),∅,{⊥,P21})	M2,P22
S70	({M1},∅,∅,{⊥,P21})	M2
S71	({M4,P4},∅,∅,{⊥,P4})	M5,P5,P17(1)
S72	({M4,TopL},P18(1),∅,{⊥,⊥})	P20
S73	({M4,P21},P19(1),∅,{⊥,P21})	M13,P22
S74	({M3,P5},∅,∅,{⊥,P4})	M4,P6,P16(1)
S75	({M3,P17(1)},P17(1),∅,{⊥,P4})	M4,TopL
S76	({M3,P20},P18(1),∅,{⊥,⊥})	M4,P21
S77	({M3,P22},P19(1),∅,{⊥,P21})	M4,P19(2)
S78	({0,P5},∅,∅,{⊥,P4})	P6,P16(1)
S79	({0,P17(1)},P17(1),∅,{⊥,P4})	TopL
S80	({0,P20},P18(1),∅,{⊥,⊥})	P21
S81	({0,P22},P19(1),∅,{⊥,P21})	P19(2)
S82	({M2,P6},∅,∅,{⊥,P4})	M3,M15,P7,P15(2)
S83	({M2,P16(1)},P16(1),∅,{⊥,P4})	M3,M15,TopL
S84	({M2,TopL},P17(1),∅,{⊥,P4})	M3,M15,P20
S85	({M2,P21},P18(1),∅,{⊥,P21})	M3,M15,P22
S86	({M2},∅,∅,{⊥,P21})	M3,M15
S87	({M1,P7},∅,∅,{⊥,P4})	M2,P14(2)
S88	({M1,P15(2)},P15(2),∅,{⊥,P4})	M2,TopL
S89	({M1,TopL},P16(1),∅,{⊥,P4})	M2,P20
S90	({M1,P20},P17(1),∅,{⊥,P4})	M2,P21
S91	({M1,P22},P18(1),∅,{⊥,P21})	M2,P18(2)
S92	({M5,P4},∅,∅,{⊥,⊥})	M6,P5,P17(1)
S93	({M4,P5},∅,∅,{⊥,P4})	M5,P6,P16(1)

Figure 41: Part II of control states for the Inres test case example

State	Value	enabled events
S94	({M4,P17(1)},P17(1),∅,{⊥,P4})	M5,TopL
S95	({M4,P20},P18(1),∅,{⊥,⊥})	P21
S96	({M13,P21},P19(1),∅,{⊥,⊥})	M14,P22
S97	({M4,P22},P19(1),∅,{⊥,P21})	M13,P19(2)
S98	({M3,P6},∅,∅,{⊥,P4})	M4,P7,P15(2)
S99	({M3,P16(1)},P16(1),∅,{⊥,P4})	M4,TopL
S100	({M3,TopL},P17(1),∅,{⊥,P4})	M4,P20
S101	({M3,P21},P18(1),∅,{⊥,P21})	M4,P22
S102	({M3},∅,∅,{⊥,P21})	M4
S103	({0,P6},∅,∅,{⊥,P4})	P7,P15(2)
S104	({0,P16(1)},P16(1),∅,{⊥,P4})	TopL
S105	({0,TopL},P17(1),∅,{⊥,P4})	P20
S106	({0,P21},P18(1),∅,{⊥,P21})	P22
S107	(∅,∅,∅,{⊥,P21})	
S108	({M2,P7},∅,∅,{⊥,P4})	M3,M15,P14(2)
S109	({M2,P15(2)},P15(2),∅,{⊥,P4})	M3,M15,TopL
S110	({M2,TopL},P16(1),∅,{⊥,P4})	M3,M15,P20
S111	({M2,P20},P17(1),∅,{⊥,P4})	M3,M15,P21
S112	({M2,P22},P18(1),∅,{⊥,P21})	M3,M15,P18(2)
S113	({M1,P14(2)},P14(2),∅,{⊥,P4})	M2,TopL
S114	({M1,TopL},P15(2),∅,{⊥,P4})	M2,P20
S115	({M1,P20},P16(1),∅,{⊥,P4})	M2,P21
S116	({M1,P21},P17(1),∅,{⊥,P21})	M2,P22
S117	({M6,P4},∅,∅,{⊥,⊥})	M7,P5,P17(1)
S118	({M5,P5},∅,∅,{⊥,⊥})	M6,P6,P16(1)
S119	({M5,P17(1)},P17(1),∅,{⊥,⊥})	M6,TopL
S120	({M4,P6},∅,∅,{⊥,P4})	M5,P7,P15(2)
S121	({M4,P16(1)},P16(1),∅,{⊥,P4})	M5,TopL
S122	({M4,TopL},P17(1),∅,{⊥,P4})	M5,P20
S123	({M4,P21},P18(1),∅,{⊥,P21})	M13,P22
S124	({0,P21},P19(1),∅,{⊥,⊥})	P22
S125	({M13,P22},P19(1),∅,{⊥,⊥})	M14,P19(2)
S126	({M4},∅,∅,{⊥,P21})	M13
S127	({M3,P7},∅,∅,{⊥,P4})	M4,P14(2)
S128	({M3,P15(2)},P15(2),∅,{⊥,P4})	M4,TopL
S129	({M3,TopL},P16(1),∅,{⊥,P4})	M4,P20
S130	({M3,P20},P17(1),∅,{⊥,P4})	M4,P21
S131	({M3,P22},P18(1),∅,{⊥,P21})	M4,P18(2)
S132	({0,P7},∅,∅,{⊥,P4})	P14(2)
S133	({0,P15(2)},P15(2),∅,{⊥,P4})	TopL
S134	({0,TopL},P16(1),∅,{⊥,P4})	P20
S135	({0,P20},P17(1),∅,{⊥,P4})	P21
S136	({0,P22},P18(1),∅,{⊥,P21})	P18(2)
S137	({M2,P14(2)},P14(2),∅,{⊥,P4})	M3,M15,TopL
S138	({M2,TopL},P15(2),∅,{⊥,P4})	M3,M15,P20
S139	({M2,P20},P16(1),∅,{⊥,P4})	M3,M15,P21
S140	({M2,P21},P17(1),∅,{⊥,P21})	M3,M15,P22

Figure 42: Part III of control states for the Inres test case example

State	Value	enabled events
S141	({M1,TopL},P14(2),∅,{⊥,P4})	M2,P20
S142	({M1,P20},P15(2),∅,{⊥,P4})	M2,P21
S143	({M1,P21},P16(1),∅,{⊥,P21})	M2,P22
S144	({M1,P22},P17(1),∅,{⊥,P21})	M2,P17(2)
S145	({M7,P4},∅,∅,{M7,⊥})	M12,P5,P17(1)
S146	({M6,P5},∅,∅,{⊥,⊥})	M7,P6,P16(1)
S147	({M6,P17(1)},P17(1),∅,{⊥,⊥})	M7,TopL
S148	({M5,P6},∅,∅,{⊥,⊥})	M6,P7,P15(2)
S149	({M5,P16(1)},P16(1),∅,{⊥,⊥})	M6,TopL
S150	({M5,TopL},P17(1),∅,{⊥,⊥})	M6,P20
S151	({M4,P7},∅,∅,{⊥,P4})	M5,P14(2)
S152	({M4,P15(2)},P15(2),∅,{⊥,P4})	M5,TopL
S153	({M4,TopL},P16(1),∅,{⊥,P4})	M5,P20
S154	({M4,P20},P17(1),∅,{⊥,P4})	M5,P21
S155	({M13,P21},P18(1),∅,{⊥,⊥})	M14,P22
S156	({M4,P22},P18(1),∅,{⊥,P21})	M13,P18(2)
S157	({0,P22},P19(1),∅,{⊥,⊥})	P19(2)
S158	({M13},∅,∅,{⊥,⊥})	M14
S159	({M3,P14(2)},P14(2),∅,{⊥,P4})	M4,TopL
S160	({M3,TopL},P15(2),∅,{⊥,P4})	M4,P20
S161	({M3,P20},P16(1),∅,{⊥,P4})	M4,P21
S162	({M3,P21},P17(1),∅,{⊥,P21})	M4,P22
S163	({0,P14(2)},P14(2),∅,{⊥,P4})	TopL
S164	({0,TopL},P15(2),∅,{⊥,P4})	P20
S165	({0,P20},P16(1),∅,{⊥,P4})	P21
S166	({0,P21},P17(1),∅,{⊥,P21})	P22
S167	({M2,TopL},P14(2),∅,{⊥,P4})	M3,M15,P20
S168	({M2,P20},P15(2),∅,{⊥,P4})	M3,M15,P21
S169	({M2,P21},P16(1),∅,{⊥,P21})	M3,M15,P22
S170	({M2,P22},P17(1),∅,{⊥,P21})	M3,M15,P17(2)
S171	({M1,P20},P14(2),∅,{⊥,P4})	M2,P21
S172	({M1,P21},P15(2),∅,{⊥,P21})	M2,P22
S173	({M1,P22},P16(1),∅,{⊥,P21})	M2,P16(2)
S174	({0,P4},∅,∅,{M7,⊥})	P5,P17(1)
S175	({M7,P5},∅,∅,{M7,⊥})	M12,P6,P16(1)
S176	({M7,P17(1)},P17(1),∅,{M7,⊥})	M12,TopL
S177	({M6,P6},∅,∅,{⊥,⊥})	M7,P7,P15(2)
S178	({M6,P16(1)},P16(1),∅,{⊥,⊥})	M7,TopL
S179	({M6,TopL},P17(1),∅,{⊥,⊥})	M7,P20
S180	({M5,P7},∅,∅,{⊥,⊥})	M6,P14(2)
S181	({M5,P15(2)},P15(2),∅,{⊥,⊥})	M6,TopL
S182	({M5,TopL},P16(1),∅,{⊥,⊥})	M6,P20
S183	({M5,P20},P17(1),∅,{⊥,⊥})	M6,P21
S184	({M4,P14(2)},P14(2),∅,{⊥,P4})	M5,TopL
S185	({M4,TopL},P15(2),∅,{⊥,P4})	M5,P20
S186	({M4,P20},P16(1),∅,{⊥,P4})	M5,P21
S187	({M4,P21},P17(1),∅,{⊥,P21})	M13,P22

Figure 43: Part IV of control states for the Inres test case example

State	Value	enabled events
S188	({0,P21},P18(1),∅,{⊥,⊥})	P22
S189	({M13,P22},P18(1),∅,{⊥,⊥})	M14,P18(2)
S190	(∅,∅,∅,{⊥,⊥})	
S191	({M3,TopL},P14(2),∅,{⊥,P4})	M4,P20
S192	({M3,P20},P15(2),∅,{⊥,P4})	M4,P21
S193	({M3,P21},P16(1),∅,{⊥,P21})	M4,P22
S194	({M3,P22},P17(1),∅,{⊥,P21})	M4,P17(2)
S195	({0,TopL},P14(2),∅,{⊥,P4})	P20
S196	({0,P20},P15(2),∅,{⊥,P4})	P21
S197	({0,P21},P16(1),∅,{⊥,P21})	P22
S198	({0,P22},P17(1),∅,{⊥,P21})	P17(2)
S199	({M2,P20},P14(2),∅,{⊥,P4})	M3,M15,P21
S200	({M2,P21},P15(2),∅,{⊥,P21})	M3,M15,P22
S201	({M2,P22},P16(1),∅,{⊥,P21})	M3,M15,P16(2)
S202	({M1,P21},P14(2),∅,{⊥,P21})	M2,P22
S203	({M1,P22},P15(2),∅,{⊥,P21})	M2,P152
S204	({0,P5},∅,∅,{M7,⊥})	P6,P16(1)
S205	({0,P17(1)},P17(1),∅,{M7,⊥})	TopL
S206	({M7,P6},∅,∅,{M7,⊥})	M12,P7,P15(2)
S207	({M7,P16(1)},P16(1),∅,{M7,⊥})	M12,TopL
S208	({M7,TopL},P17(1),∅,{M7,⊥})	M12,P20
S209	({M6,P7},∅,∅,{⊥,⊥})	M7,P14(2)
S210	({M6,P15(2)},P15(2),∅,{⊥,⊥})	M7,TopL
S211	({M6,TopL},P16(1),∅,{⊥,⊥})	M7,P20
S212	({M6,P20},P17(1),∅,{⊥,⊥})	M7,P21
S213	({M5,P14(2)},P14(2),∅,{⊥,⊥})	M6,TopL
S214	({M5,TopL},P15(2),∅,{⊥,⊥})	M6,P20
S215	({M5,P20},P16(1),∅,{⊥,⊥})	M6,P21
S216	({M5,P21},P17(1),∅,{⊥,P21})	M6,P22
S217	({M4,TopL},P14(2),∅,{⊥,P4})	M5,P20
S218	({M4,P20},P15(2),∅,{⊥,P4})	M5,P21
S219	({M4,P21},P16(1),∅,{⊥,P21})	M13,P22
S220	({M13,P21},P17(1),∅,{⊥,⊥})	M14,P22
S221	({M4,P22},P17(1),∅,{⊥,P21})	M13,P17(2)
S222	({0,P22},P18(1),∅,{⊥,⊥})	P18(2)
S223	({M3,P20},P14(2),∅,{⊥,P4})	M4,P21
S224	({M3,P21},P15(2),∅,{⊥,P21})	M4,P22
S225	({M3,P22},P16(1),∅,{⊥,P21})	M4,P16(2)
S226	({0,P20},P14(2),∅,{⊥,P4})	P21
S227	({0,P21},P15(2),∅,{⊥,P21})	P22
S228	({0,P22},P16(1),∅,{⊥,P21})	P16(2)
S229	({M2,P21},P14(2),∅,{⊥,P21})	M3,M15,P22
S230	({M2,P22},P15(2),∅,{⊥,P21})	M3,M15,P152
S231	({M1,P22},P14(2),∅,{⊥,P21})	M2,P142
S232	({0,P6},∅,∅,{M7,⊥})	P7,P15(2)
S233	({0,P16(1)},P16(1),∅,{M7,⊥})	TopL
S234	({0,TopL},P17(1),∅,{M7,⊥})	P20

Figure 44: Part V of control states for the Inres test case example

State	Value	enabled events
S235	({M7,P7},∅,∅,{M7,⊥})	M12,P8,P14(2)
S236	({M7,P15(2)},P15(2),∅,{M7,⊥})	M12,TopL
S237	({M7,TopL},P16(1),∅,{M7,⊥})	M12,P20
S238	({M7,P20},P17(1),∅,{M7,⊥})	M12,P21
S239	({M6,P14(2)},P14(2),∅,{⊥,⊥})	M7,TopL
S240	({M6,TopL},P15(2),∅,{⊥,⊥})	M7,P20
S241	({M6,P20},P16(1),∅,{⊥,⊥})	M7,P21
S242	({M6,P21},P17(1),∅,{⊥,P21})	M7,P22
S243	({M5,TopL},P14(2),∅,{⊥,⊥})	M6,P20
S244	({M5,P20},P15(2),∅,{⊥,⊥})	M6,P21
S245	({M5,P21},P16(1),∅,{⊥,P21})	M6,P22
S246	({M5,P22},P17(1),∅,{⊥,P21})	M6,P17(2)
S247	({M4,P20},P14(2),∅,{⊥,P4})	M5,P21
S248	({M4,P21},P15(2),∅,{⊥,P21})	M13,P22
S249	({M13,P21},P16(1),∅,{⊥,⊥})	M14,P22
S250	({M4,P22},P16(1),∅,{⊥,P21})	M13,P16(2)
S251	({0,P21},P17(1),∅,{⊥,⊥})	P22
S252	({M13,P22},P17(1),∅,{⊥,⊥})	M14,P17(2)
S253	({M3,P21},P14(2),∅,{⊥,P21})	M4,P22
S254	({M3,P22},P15(2),∅,{⊥,P21})	M4,P152
S255	({0,P21},P14(2),∅,{⊥,P21})	P22
S256	({0,P22},P15(2),∅,{⊥,P21})	P152
S257	({M2,P22},P14(2),∅,{⊥,P21})	M3,M15,P142
S258	({0,P7},∅,∅,{M7,⊥})	P8,P14(2)
S259	({0,P15(2)},P15(2),∅,{M7,⊥})	TopL
S260	({0,TopL},P16(1),∅,{M7,⊥})	P20
S261	({0,P20},P17(1),∅,{M7,⊥})	P21
S262	({M7,P8},∅,∅,{⊥,⊥})	M12,P9
S263	({M7,P14(2)},P14(2),∅,{M7,⊥})	M12,TopL
S264	({M7,TopL},P15(2),∅,{M7,⊥})	M12,P20
S265	({M7,P20},P16(1),∅,{M7,⊥})	M12,P21
S266	({M7,P21},P17(1),∅,{M7,P21})	M10,M12,P22
S267	({M6,TopL},P14(2),∅,{⊥,⊥})	M7,P20
S268	({M6,P20},P15(2),∅,{⊥,⊥})	M7,P21
S269	({M6,P21},P16(1),∅,{⊥,P21})	M7,P22
S270	({M6,P22},P17(1),∅,{⊥,P21})	M7,P17(2)
S271	({M5,P20},P14(2),∅,{⊥,⊥})	M6,P21
S272	({M5,P21},P15(2),∅,{⊥,P21})	M6,P22
S273	({M5,P22},P16(1),∅,{⊥,P21})	M6,P16(2)
S274	({M5},∅,∅,{⊥,P21})	M6
S275	({M4,P21},P14(2),∅,{⊥,P21})	M13,P22
S276	({M13,P21},P15(2),∅,{⊥,⊥})	M14,P22
S277	({M4,P22},P15(2),∅,{⊥,P21})	M13,P152
S278	({0,P21},P16(1),∅,{⊥,⊥})	P22
S279	({M13,P22},P16(1),∅,{⊥,⊥})	M14,P16(2)
S280	({0,P22},P17(1),∅,{⊥,⊥})	P17(2)
S281	({M3,P22},P14(2),∅,{⊥,P21})	M4,P142

Figure 45: Part VI of control states for the Inres test case example

State	Value	enabled events
S282	({0,P22},P14(2),∅,{⊥,P21})	P142
S283	({0,P8},∅,∅,{⊥,⊥})	P9
S284	({0,P14(2)},P14(2),∅,{M7,⊥})	TopL
S285	({0,TopL},P15(2),∅,{M7,⊥})	P20
S286	({0,P20},P16(1),∅,{M7,⊥})	P21
S287	({0,P21},P17(1),∅,{M7,P21})	P22
S288	({M7,P9},∅,∅,{⊥,⊥})	M12,P10,P13(2)
S289	({M7,TopL},P14(2),∅,{M7,⊥})	M12,P20
S290	({M7,P20},P15(2),∅,{M7,⊥})	M12,P21
S291	({M7,P21},P16(1),∅,{M7,P21})	M10,M12,P22
S292	({M10,P21},P17(1),∅,{M7,⊥})	M11,P22
S293	({M7,P22},P17(1),∅,{M7,P21})	M10,M12,P17(2)
S294	({M6,P20},P14(2),∅,{⊥,⊥})	M7,P21
S295	({M6,P21},P15(2),∅,{⊥,P21})	M7,P22
S296	({M6,P22},P16(1),∅,{⊥,P21})	M7,P16(2)
S297	({M6},∅,∅,{⊥,P21})	M7
S298	({M5,P21},P14(2),∅,{⊥,P21})	M6,P22
S299	({M5,P22},P15(2),∅,{⊥,P21})	M6,P152
S300	({M13,P21},P14(2),∅,{⊥,⊥})	M14,P22
S301	({M4,P22},P14(2),∅,{⊥,P21})	M13,P142
S302	({0,P21},P15(2),∅,{⊥,⊥})	P22
S303	({M13,P22},P15(2),∅,{⊥,⊥})	M14,P152
S304	({0,P22},P16(1),∅,{⊥,⊥})	P16(2)
S305	({0,P9},∅,∅,{⊥,⊥})	P10,P13(2)
S306	({0,TopL},P14(2),∅,{M7,⊥})	P20
S307	({0,P20},P15(2),∅,{M7,⊥})	P21
S308	({0,P21},P16(1),∅,{M7,P21})	P22
S309	({0,P22},P17(1),∅,{M7,P21})	P17(2)
S310	({M7,P10},∅,∅,{⊥,⊥})	M12,P11
S311	({M7,P13(2)},P13(2),∅,{⊥,⊥})	M12,TopL
S312	({M7,P20},P14(2),∅,{M7,⊥})	M12,P21
S313	({M7,P21},P15(2),∅,{M7,P21})	M10,M12,P22
S314	({M10,P21},P16(1),∅,{M7,⊥})	M11,P22
S315	({M7,P22},P16(1),∅,{M7,P21})	M10,M12,P16(2)
S316	({0,P21},P17(1),∅,{M7,⊥})	P22
S317	({M10,P22},P17(1),∅,{M7,⊥})	M11,P17(2)
S318	({M7},∅,∅,{M7,P21})	M10,M12
S319	({M6,P21},P14(2),∅,{⊥,P21})	M7,P22
S320	({M6,P22},P15(2),∅,{⊥,P21})	M7,P152
S321	({M5,P22},P14(2),∅,{⊥,P21})	M6,P142
S322	({0,P21},P14(2),∅,{⊥,⊥})	P22
S323	({M13,P22},P14(2),∅,{⊥,⊥})	M14,P142
S324	({0,P22},P15(2),∅,{⊥,⊥})	P152
S325	({0,P10},∅,∅,{⊥,⊥})	P11
S326	({0,P13(2)},P13(2),∅,{⊥,⊥})	TopL
S327	({0,P20},P14(2),∅,{M7,⊥})	P21
S328	({0,P21},P15(2),∅,{M7,P21})	P22

Figure 46: Part VII of control states for the Inres test case example

State	Value	enabled events
S329	({0,P22},P16(1),∅,{M7,P21})	P16(2)
S330	(∅,∅,∅,{M7,P21})	
S331	({M7,P11},∅,∅,{⊥,P11})	M8,M12,P12
S332	({M7,TopL},P13(2),∅,{⊥,⊥})	M12,P20
S333	({M7,P21},P14(2),∅,{M7,P21})	M10,M12,P22
S334	({M10,P21},P15(2),∅,{M7,⊥})	M11,P22
S335	({M7,P22},P15(2),∅,{M7,P21})	M10,M12,P152
S336	({0,P21},P16(1),∅,{M7,⊥})	P22
S337	({M10,P22},P16(1),∅,{M7,⊥})	M11,P16(2)
S338	({0,P22},P17(1),∅,{M7,⊥})	P17(2)
S339	({M10},∅,∅,{M7,⊥})	M11
S340	({M6,P22},P14(2),∅,{⊥,P21})	M7,P142
S341	({0,P22},P14(2),∅,{⊥,⊥})	P142
S342	({0,P11},∅,∅,{⊥,P11})	P12
S343	({0,TopL},P13(2),∅,{⊥,⊥})	P20
S344	({0,P21},P14(2),∅,{M7,P21})	P22
S345	({0,P22},P15(2),∅,{M7,P21})	P152
S346	({M8,P11},∅,∅,{⊥,⊥})	M9,P12
S347	({M7},∅,∅,{⊥,P11})	M8,M12
S348	({M7,P20},P13(2),∅,{⊥,⊥})	M12,P21
S349	({M10,P21},P14(2),∅,{M7,⊥})	M11,P22
S350	({M7,P22},P14(2),∅,{M7,P21})	M10,M12,P142
S351	({0,P21},P15(2),∅,{M7,⊥})	P22
S352	({M10,P22},P15(2),∅,{M7,⊥})	M11,P152
S353	({0,P22},P16(1),∅,{M7,⊥})	P16(2)
S354	(∅,∅,∅,{M7,⊥})	
S355	(∅,∅,∅,{⊥,P11})	
S356	({0,P20},P13(2),∅,{⊥,⊥})	P21
S357	({0,P22},P14(2),∅,{M7,P21})	P142
S358	({0,P11},∅,∅,{⊥,⊥})	P12
S359	({M8},∅,∅,{⊥,⊥})	M9
S360	({M7,P21},P13(2),∅,{⊥,P21})	M10,M12,P22
S361	({0,P21},P14(2),∅,{M7,⊥})	P22
S362	({M10,P22},P14(2),∅,{M7,⊥})	M11,P142
S363	({0,P22},P15(2),∅,{M7,⊥})	P152
S364	({0,P21},P13(2),∅,{⊥,P21})	P22
S365	({M10,P21},P13(2),∅,{⊥,⊥})	M11,P22
S366	({M7,P22},P13(2),∅,{⊥,P21})	M10,M12,P132
S367	({0,P22},P14(2),∅,{M7,⊥})	P142
S368	({0,P22},P13(2),∅,{⊥,P21})	P132
S369	({0,P21},P13(2),∅,{⊥,⊥})	P22
S370	({M10,P22},P13(2),∅,{⊥,⊥})	M11,P132
S371	({M7},∅,∅,{⊥,P21})	M10,M12
S372	({0,P22},P13(2),∅,{⊥,⊥})	P132
S373	({M10},∅,∅,{⊥,⊥})	M11

Figure 47: Part VIII of control states for the Inres test case example

$S0 \xrightarrow{M1}_c S1$	$S1 \xrightarrow{M2}_c S2$	$S1 \xrightarrow{PTC}_c S3$	$S2 \xrightarrow{M3}_c S4$	$S2 \xrightarrow{M15}_c S5$
$S2 \xrightarrow{PTC}_c S6$	$S3 \xrightarrow{M2}_c S6$	$S3 \xrightarrow{P1}_c S7$	$S3 \xrightarrow{P19(2)}_c S8$	$S4 \xrightarrow{M4}_c S9$
$S4 \xrightarrow{PTC}_c S10$	$S5 \xrightarrow{PTC}_c S11$	$S6 \xrightarrow{M3}_c S10$	$S6 \xrightarrow{M15}_c S11$	$S6 \xrightarrow{P1}_c S12$
$S6 \xrightarrow{P19(2)}_c S13$	$S7 \xrightarrow{M2}_c S12$	$S7 \xrightarrow{P2}_c S14$	$S8 \xrightarrow{M2}_c S13$	$S8 \xrightarrow{TopL}_c S15$
$S9 \xrightarrow{PTC}_c S16$	$S10 \xrightarrow{M4}_c S16$	$S10 \xrightarrow{P1}_c S17$	$S10 \xrightarrow{P19(2)}_c S18$	$S11 \xrightarrow{P1}_c S19$
$S11 \xrightarrow{P19(2)}_c S20$	$S12 \xrightarrow{M3}_c S17$	$S12 \xrightarrow{M15}_c S19$	$S12 \xrightarrow{P2}_c S21$	$S13 \xrightarrow{M3}_c S18$
$S13 \xrightarrow{M15}_c S20$	$S13 \xrightarrow{TopL}_c S22$	$S14 \xrightarrow{M2}_c S21$	$S14 \xrightarrow{P3}_c S23$	$S14 \xrightarrow{P18(2)}_c S24$
$S15 \xrightarrow{M2}_c S22$	$S15 \xrightarrow{P20}_c S25$	$S16 \xrightarrow{P1}_c S26$	$S16 \xrightarrow{P19(2)}_c S27$	$S17 \xrightarrow{M4}_c S26$
$S17 \xrightarrow{P2}_c S28$	$S18 \xrightarrow{M4}_c S27$	$S18 \xrightarrow{TopL}_c S29$	$S19 \xrightarrow{P2}_c S30$	$S20 \xrightarrow{TopL}_c S31$
$S21 \xrightarrow{M3}_c S28$	$S21 \xrightarrow{M15}_c S30$	$S21 \xrightarrow{P3}_c S32$	$S21 \xrightarrow{P18(2)}_c S33$	$S22 \xrightarrow{M3}_c S29$
$S22 \xrightarrow{M15}_c S31$	$S22 \xrightarrow{P20}_c S34$	$S23 \xrightarrow{M2}_c S32$	$S23 \xrightarrow{P4}_c S35$	$S24 \xrightarrow{M2}_c S33$
$S24 \xrightarrow{TopL}_c S36$	$S25 \xrightarrow{M2}_c S34$	$S25 \xrightarrow{P21}_c S37$	$S26 \xrightarrow{P2}_c S38$	$S27 \xrightarrow{TopL}_c S39$
$S28 \xrightarrow{M4}_c S38$	$S28 \xrightarrow{P3}_c S40$	$S28 \xrightarrow{P18(2)}_c S41$	$S29 \xrightarrow{M4}_c S39$	$S29 \xrightarrow{P20}_c S42$
$S30 \xrightarrow{P3}_c S43$	$S30 \xrightarrow{P18(2)}_c S44$	$S31 \xrightarrow{P20}_c S45$	$S32 \xrightarrow{M3}_c S40$	$S32 \xrightarrow{M15}_c S43$
$S32 \xrightarrow{P4}_c S46$	$S33 \xrightarrow{M3}_c S41$	$S33 \xrightarrow{M15}_c S44$	$S33 \xrightarrow{TopL}_c S47$	$S34 \xrightarrow{M3}_c S42$
$S34 \xrightarrow{M15}_c S45$	$S34 \xrightarrow{P21}_c S48$	$S35 \xrightarrow{M2}_c S46$	$S35 \xrightarrow{P5}_c S49$	$S35 \xrightarrow{P17(2)}_c S50$
$S36 \xrightarrow{M2}_c S47$	$S36 \xrightarrow{P20}_c S51$	$S37 \xrightarrow{M2}_c S48$	$S37 \xrightarrow{P22}_c S52$	$S38 \xrightarrow{P3}_c S53$
$S38 \xrightarrow{P18(2)}_c S54$	$S39 \xrightarrow{P20}_c S55$	$S40 \xrightarrow{M4}_c S53$	$S40 \xrightarrow{P4}_c S56$	$S41 \xrightarrow{M4}_c S54$
$S41 \xrightarrow{TopL}_c S57$	$S42 \xrightarrow{M4}_c S55$	$S42 \xrightarrow{P21}_c S58$	$S43 \xrightarrow{P4}_c S59$	$S44 \xrightarrow{TopL}_c S60$
$S45 \xrightarrow{P21}_c S61$	$S46 \xrightarrow{M3}_c S56$	$S46 \xrightarrow{M15}_c S59$	$S46 \xrightarrow{P5}_c S62$	$S46 \xrightarrow{P17(2)}_c S63$
$S47 \xrightarrow{M3}_c S57$	$S47 \xrightarrow{M15}_c S60$	$S47 \xrightarrow{P20}_c S64$	$S48 \xrightarrow{M3}_c S58$	$S48 \xrightarrow{M15}_c S61$
$S48 \xrightarrow{P22}_c S65$	$S49 \xrightarrow{M2}_c S62$	$S49 \xrightarrow{P6}_c S66$	$S49 \xrightarrow{P16(2)}_c S67$	$S50 \xrightarrow{M2}_c S63$
$S50 \xrightarrow{TopL}_c S68$	$S51 \xrightarrow{M2}_c S64$	$S51 \xrightarrow{P21}_c S69$	$S52 \xrightarrow{M2}_c S65$	$S52 \xrightarrow{P192}_c S70$
$S53 \xrightarrow{P4}_c S71$	$S54 \xrightarrow{TopL}_c S72$	$S55 \xrightarrow{P21}_c S73$	$S56 \xrightarrow{M4}_c S71$	$S56 \xrightarrow{P5}_c S74$
$S56 \xrightarrow{P17(2)}_c S75$	$S57 \xrightarrow{M4}_c S72$	$S57 \xrightarrow{P20}_c S76$	$S58 \xrightarrow{M4}_c S73$	$S58 \xrightarrow{P22}_c S77$
$S59 \xrightarrow{P5}_c S78$	$S59 \xrightarrow{P17(2)}_c S79$	$S60 \xrightarrow{P20}_c S80$	$S61 \xrightarrow{P22}_c S81$	$S62 \xrightarrow{M3}_c S74$
$S62 \xrightarrow{M15}_c S78$	$S62 \xrightarrow{P6}_c S82$	$S62 \xrightarrow{P16(2)}_c S83$	$S63 \xrightarrow{M3}_c S75$	$S63 \xrightarrow{M15}_c S79$
$S63 \xrightarrow{TopL}_c S84$	$S64 \xrightarrow{M3}_c S76$	$S64 \xrightarrow{M15}_c S80$	$S64 \xrightarrow{P21}_c S85$	$S65 \xrightarrow{M3}_c S77$
$S65 \xrightarrow{M15}_c S81$	$S65 \xrightarrow{P192}_c S86$	$S66 \xrightarrow{M2}_c S82$	$S66 \xrightarrow{P7}_c S87$	$S66 \xrightarrow{P15(2)}_c S88$
$S67 \xrightarrow{M2}_c S83$	$S67 \xrightarrow{TopL}_c S89$	$S68 \xrightarrow{M2}_c S84$	$S68 \xrightarrow{P20}_c S90$	$S69 \xrightarrow{M2}_c S85$
$S69 \xrightarrow{P22}_c S91$	$S70 \xrightarrow{M2}_c S86$	$S71 \xrightarrow{M5}_c S92$	$S71 \xrightarrow{P5}_c S93$	$S71 \xrightarrow{P17(2)}_c S94$
$S72 \xrightarrow{P20}_c S95$	$S73 \xrightarrow{M13}_c S96$	$S73 \xrightarrow{P22}_c S97$	$S74 \xrightarrow{M4}_c S93$	$S74 \xrightarrow{P6}_c S98$
$S74 \xrightarrow{P16(2)}_c S99$	$S75 \xrightarrow{M4}_c S94$	$S75 \xrightarrow{TopL}_c S100$	$S76 \xrightarrow{M4}_c S95$	$S76 \xrightarrow{P21}_c S101$
$S77 \xrightarrow{M4}_c S97$	$S77 \xrightarrow{P192}_c S102$	$S78 \xrightarrow{P6}_c S103$	$S78 \xrightarrow{P16(2)}_c S104$	$S79 \xrightarrow{TopL}_c S105$
$S80 \xrightarrow{P21}_c S106$	$S81 \xrightarrow{P192}_c S107$	$S82 \xrightarrow{M3}_c S98$	$S82 \xrightarrow{M15}_c S103$	$S82 \xrightarrow{P7}_c S108$
$S82 \xrightarrow{P15(2)}_c S109$	$S83 \xrightarrow{M3}_c S99$	$S83 \xrightarrow{M15}_c S104$	$S83 \xrightarrow{TopL}_c S110$	$S84 \xrightarrow{M3}_c S100$
$S84 \xrightarrow{M15}_c S105$	$S84 \xrightarrow{P20}_c S111$	$S85 \xrightarrow{M3}_c S101$	$S85 \xrightarrow{M15}_c S106$	$S85 \xrightarrow{P22}_c S112$

Figure 48: Part I of control state transitions for the Inres test case example

$S86 \xrightarrow{M3}_c S102$	$S86 \xrightarrow{M15}_c S107$	$S87 \xrightarrow{M2}_c S108$	$S87 \xrightarrow{P14(2)}_c S113$	$S88 \xrightarrow{M2}_c S109$
$S88 \xrightarrow{TopL}_c S114$	$S89 \xrightarrow{M2}_c S110$	$S89 \xrightarrow{P20}_c S115$	$S90 \xrightarrow{M2}_c S111$	$S90 \xrightarrow{P21}_c S116$
$S91 \xrightarrow{M2}_c S112$	$S91 \xrightarrow{P182}_c S70$	$S92 \xrightarrow{M6}_c S117$	$S92 \xrightarrow{P5}_c S118$	$S92 \xrightarrow{P17(2)}_c S119$
$S93 \xrightarrow{M5}_c S118$	$S93 \xrightarrow{P6}_c S120$	$S93 \xrightarrow{P16(2)}_c S121$	$S94 \xrightarrow{M5}_c S119$	$S94 \xrightarrow{TopL}_c S122$
$S95 \xrightarrow{P21}_c S123$	$S96 \xrightarrow{M14}_c S124$	$S96 \xrightarrow{P22}_c S125$	$S97 \xrightarrow{M13}_c S125$	$S97 \xrightarrow{P192}_c S126$
$S98 \xrightarrow{M4}_c S120$	$S98 \xrightarrow{P7}_c S127$	$S98 \xrightarrow{P15(2)}_c S128$	$S99 \xrightarrow{M4}_c S121$	$S99 \xrightarrow{TopL}_c S129$
$S100 \xrightarrow{M4}_c S122$	$S100 \xrightarrow{P20}_c S130$	$S101 \xrightarrow{M4}_c S123$	$S101 \xrightarrow{P22}_c S131$	$S102 \xrightarrow{M4}_c S126$
$S103 \xrightarrow{P7}_c S132$	$S103 \xrightarrow{P15(2)}_c S133$	$S104 \xrightarrow{TopL}_c S134$	$S105 \xrightarrow{P20}_c S135$	$S106 \xrightarrow{P22}_c S136$
$S108 \xrightarrow{M3}_c S127$	$S108 \xrightarrow{M15}_c S132$	$S108 \xrightarrow{P14(2)}_c S137$	$S109 \xrightarrow{M3}_c S128$	$S109 \xrightarrow{M15}_c S133$
$S109 \xrightarrow{TopL}_c S138$	$S110 \xrightarrow{M3}_c S129$	$S110 \xrightarrow{M15}_c S134$	$S110 \xrightarrow{P20}_c S139$	$S111 \xrightarrow{M3}_c S130$
$S111 \xrightarrow{M15}_c S135$	$S111 \xrightarrow{P21}_c S140$	$S112 \xrightarrow{M3}_c S131$	$S112 \xrightarrow{M15}_c S136$	$S112 \xrightarrow{P182}_c S86$
$S113 \xrightarrow{M2}_c S137$	$S113 \xrightarrow{TopL}_c S141$	$S114 \xrightarrow{M2}_c S138$	$S114 \xrightarrow{P20}_c S142$	$S115 \xrightarrow{M2}_c S139$
$S115 \xrightarrow{P21}_c S143$	$S116 \xrightarrow{M2}_c S140$	$S116 \xrightarrow{P22}_c S144$	$S117 \xrightarrow{M7}_c S145$	$S117 \xrightarrow{P5}_c S146$
$S117 \xrightarrow{P17(2)}_c S147$	$S118 \xrightarrow{M6}_c S146$	$S118 \xrightarrow{P6}_c S148$	$S118 \xrightarrow{P16(2)}_c S149$	$S119 \xrightarrow{M6}_c S147$
$S119 \xrightarrow{TopL}_c S150$	$S120 \xrightarrow{M5}_c S148$	$S120 \xrightarrow{P7}_c S151$	$S120 \xrightarrow{P15(2)}_c S152$	$S121 \xrightarrow{M5}_c S149$
$S121 \xrightarrow{TopL}_c S153$	$S122 \xrightarrow{M5}_c S150$	$S122 \xrightarrow{P20}_c S154$	$S123 \xrightarrow{M13}_c S155$	$S123 \xrightarrow{P22}_c S156$
$S124 \xrightarrow{P22}_c S157$	$S125 \xrightarrow{M14}_c S157$	$S125 \xrightarrow{P192}_c S158$	$S126 \xrightarrow{M13}_c S158$	$S127 \xrightarrow{M4}_c S151$
$S127 \xrightarrow{P14(2)}_c S159$	$S128 \xrightarrow{M4}_c S152$	$S128 \xrightarrow{TopL}_c S160$	$S129 \xrightarrow{M4}_c S153$	$S129 \xrightarrow{P20}_c S161$
$S130 \xrightarrow{M4}_c S154$	$S130 \xrightarrow{P21}_c S162$	$S131 \xrightarrow{M4}_c S156$	$S131 \xrightarrow{P182}_c S102$	$S132 \xrightarrow{P14(2)}_c S163$
$S133 \xrightarrow{TopL}_c S164$	$S134 \xrightarrow{P20}_c S165$	$S135 \xrightarrow{P21}_c S166$	$S136 \xrightarrow{P182}_c S107$	$S137 \xrightarrow{M3}_c S159$
$S137 \xrightarrow{M15}_c S163$	$S137 \xrightarrow{TopL}_c S167$	$S138 \xrightarrow{M3}_c S160$	$S138 \xrightarrow{M15}_c S164$	$S138 \xrightarrow{P20}_c S168$
$S139 \xrightarrow{M3}_c S161$	$S139 \xrightarrow{M15}_c S165$	$S139 \xrightarrow{P21}_c S169$	$S140 \xrightarrow{M3}_c S162$	$S140 \xrightarrow{M15}_c S166$
$S140 \xrightarrow{P22}_c S170$	$S141 \xrightarrow{M2}_c S167$	$S141 \xrightarrow{P20}_c S171$	$S142 \xrightarrow{M2}_c S168$	$S142 \xrightarrow{P21}_c S172$
$S143 \xrightarrow{M2}_c S169$	$S143 \xrightarrow{P22}_c S173$	$S144 \xrightarrow{M2}_c S170$	$S144 \xrightarrow{P172}_c S70$	$S145 \xrightarrow{M12}_c S174$
$S145 \xrightarrow{P5}_c S175$	$S145 \xrightarrow{P17(2)}_c S176$	$S146 \xrightarrow{M7}_c S175$	$S146 \xrightarrow{P6}_c S177$	$S146 \xrightarrow{P16(2)}_c S178$
$S147 \xrightarrow{M7}_c S176$	$S147 \xrightarrow{TopL}_c S179$	$S148 \xrightarrow{M6}_c S177$	$S148 \xrightarrow{P7}_c S180$	$S148 \xrightarrow{P15(2)}_c S181$
$S149 \xrightarrow{M6}_c S178$	$S149 \xrightarrow{TopL}_c S182$	$S150 \xrightarrow{M6}_c S179$	$S150 \xrightarrow{P20}_c S183$	$S151 \xrightarrow{M5}_c S180$
$S151 \xrightarrow{P14(2)}_c S184$	$S152 \xrightarrow{M5}_c S181$	$S152 \xrightarrow{TopL}_c S185$	$S153 \xrightarrow{M5}_c S182$	$S153 \xrightarrow{P20}_c S186$
$S154 \xrightarrow{M5}_c S183$	$S154 \xrightarrow{P21}_c S187$	$S155 \xrightarrow{M14}_c S188$	$S155 \xrightarrow{P22}_c S189$	$S156 \xrightarrow{M13}_c S189$
$S156 \xrightarrow{P182}_c S126$	$S157 \xrightarrow{P192}_c S190$	$S158 \xrightarrow{M14}_c S190$	$S159 \xrightarrow{M4}_c S184$	$S159 \xrightarrow{TopL}_c S191$
$S160 \xrightarrow{M4}_c S185$	$S160 \xrightarrow{P20}_c S192$	$S161 \xrightarrow{M4}_c S186$	$S161 \xrightarrow{P21}_c S193$	$S162 \xrightarrow{M4}_c S187$
$S162 \xrightarrow{P22}_c S194$	$S163 \xrightarrow{TopL}_c S195$	$S164 \xrightarrow{P20}_c S196$	$S165 \xrightarrow{P21}_c S197$	$S166 \xrightarrow{P22}_c S198$
$S167 \xrightarrow{M3}_c S191$	$S167 \xrightarrow{M15}_c S195$	$S167 \xrightarrow{P20}_c S199$	$S168 \xrightarrow{M3}_c S192$	$S168 \xrightarrow{M15}_c S196$
$S168 \xrightarrow{P21}_c S200$	$S169 \xrightarrow{M3}_c S193$	$S169 \xrightarrow{M15}_c S197$	$S169 \xrightarrow{P22}_c S201$	$S170 \xrightarrow{M3}_c S194$
$S170 \xrightarrow{M15}_c S198$	$S170 \xrightarrow{P172}_c S86$	$S171 \xrightarrow{M2}_c S199$	$S171 \xrightarrow{P21}_c S202$	$S172 \xrightarrow{M2}_c S200$
$S172 \xrightarrow{P22}_c S203$	$S173 \xrightarrow{M2}_c S201$	$S173 \xrightarrow{P162}_c S70$	$S174 \xrightarrow{P5}_c S204$	$S174 \xrightarrow{P17(2)}_c S205$
$S175 \xrightarrow{M12}_c S204$	$S175 \xrightarrow{P6}_c S206$	$S175 \xrightarrow{P16(2)}_c S207$	$S176 \xrightarrow{M12}_c S205$	$S176 \xrightarrow{TopL}_c S208$

Figure 49: Part II of control state transitions for the Inres test case example

$S177 \xrightarrow{M7}_c S206$	$S177 \xrightarrow{P7}_c S209$	$S177 \xrightarrow{P15(2)}_c S210$	$S178 \xrightarrow{M7}_c S207$	$S178 \xrightarrow{TopL}_c S211$
$S179 \xrightarrow{M7}_c S208$	$S179 \xrightarrow{P20}_c S212$	$S180 \xrightarrow{M6}_c S209$	$S180 \xrightarrow{P14(2)}_c S213$	$S181 \xrightarrow{M6}_c S210$
$S181 \xrightarrow{TopL}_c S214$	$S182 \xrightarrow{M6}_c S211$	$S182 \xrightarrow{P20}_c S215$	$S183 \xrightarrow{M6}_c S212$	$S183 \xrightarrow{P21}_c S216$
$S184 \xrightarrow{M5}_c S213$	$S184 \xrightarrow{TopL}_c S217$	$S185 \xrightarrow{M5}_c S214$	$S185 \xrightarrow{P20}_c S218$	$S186 \xrightarrow{M5}_c S215$
$S186 \xrightarrow{P21}_c S219$	$S187 \xrightarrow{M13}_c S220$	$S187 \xrightarrow{P22}_c S221$	$S188 \xrightarrow{P22}_c S222$	$S189 \xrightarrow{M14}_c S222$
$S189 \xrightarrow{P182}_c S158$	$S191 \xrightarrow{M4}_c S217$	$S191 \xrightarrow{P20}_c S223$	$S192 \xrightarrow{M4}_c S218$	$S192 \xrightarrow{P21}_c S224$
$S193 \xrightarrow{M4}_c S219$	$S193 \xrightarrow{P22}_c S225$	$S194 \xrightarrow{M4}_c S221$	$S194 \xrightarrow{P172}_c S102$	$S195 \xrightarrow{P20}_c S226$
$S196 \xrightarrow{P21}_c S227$	$S197 \xrightarrow{P22}_c S228$	$S198 \xrightarrow{P172}_c S107$	$S199 \xrightarrow{M3}_c S223$	$S199 \xrightarrow{M15}_c S226$
$S199 \xrightarrow{P21}_c S229$	$S200 \xrightarrow{M3}_c S224$	$S200 \xrightarrow{M15}_c S227$	$S200 \xrightarrow{P22}_c S230$	$S201 \xrightarrow{M3}_c S225$
$S201 \xrightarrow{M15}_c S228$	$S201 \xrightarrow{P162}_c S86$	$S202 \xrightarrow{M2}_c S229$	$S202 \xrightarrow{P22}_c S231$	$S203 \xrightarrow{M2}_c S230$
$S203 \xrightarrow{P152}_c S70$	$S204 \xrightarrow{P6}_c S232$	$S204 \xrightarrow{P16(2)}_c S233$	$S205 \xrightarrow{TopL}_c S234$	$S206 \xrightarrow{M12}_c S232$
$S206 \xrightarrow{P7}_c S235$	$S206 \xrightarrow{P15(2)}_c S236$	$S207 \xrightarrow{M12}_c S233$	$S207 \xrightarrow{TopL}_c S237$	$S208 \xrightarrow{M12}_c S234$
$S208 \xrightarrow{P20}_c S238$	$S209 \xrightarrow{M7}_c S235$	$S209 \xrightarrow{P14(2)}_c S239$	$S210 \xrightarrow{M7}_c S236$	$S210 \xrightarrow{TopL}_c S240$
$S211 \xrightarrow{M7}_c S237$	$S211 \xrightarrow{P20}_c S241$	$S212 \xrightarrow{M7}_c S238$	$S212 \xrightarrow{P21}_c S242$	$S213 \xrightarrow{M6}_c S239$
$S213 \xrightarrow{TopL}_c S243$	$S214 \xrightarrow{M6}_c S240$	$S214 \xrightarrow{P20}_c S244$	$S215 \xrightarrow{M6}_c S241$	$S215 \xrightarrow{P21}_c S245$
$S216 \xrightarrow{M6}_c S242$	$S216 \xrightarrow{P22}_c S246$	$S217 \xrightarrow{M5}_c S243$	$S217 \xrightarrow{P20}_c S247$	$S218 \xrightarrow{M5}_c S244$
$S218 \xrightarrow{P21}_c S248$	$S219 \xrightarrow{M13}_c S249$	$S219 \xrightarrow{P22}_c S250$	$S220 \xrightarrow{M14}_c S251$	$S220 \xrightarrow{P22}_c S252$
$S221 \xrightarrow{M13}_c S252$	$S221 \xrightarrow{P172}_c S126$	$S222 \xrightarrow{P182}_c S190$	$S223 \xrightarrow{M4}_c S247$	$S223 \xrightarrow{P21}_c S253$
$S224 \xrightarrow{M4}_c S248$	$S224 \xrightarrow{P22}_c S254$	$S225 \xrightarrow{M4}_c S250$	$S225 \xrightarrow{P162}_c S102$	$S226 \xrightarrow{P21}_c S255$
$S227 \xrightarrow{P22}_c S256$	$S228 \xrightarrow{P162}_c S107$	$S229 \xrightarrow{M3}_c S253$	$S229 \xrightarrow{M15}_c S255$	$S229 \xrightarrow{P22}_c S257$
$S230 \xrightarrow{M3}_c S254$	$S230 \xrightarrow{M15}_c S256$	$S230 \xrightarrow{P152}_c S86$	$S231 \xrightarrow{M2}_c S257$	$S231 \xrightarrow{P142}_c S70$
$S232 \xrightarrow{P7}_c S258$	$S232 \xrightarrow{P15(2)}_c S259$	$S233 \xrightarrow{TopL}_c S260$	$S234 \xrightarrow{P20}_c S261$	$S235 \xrightarrow{M12}_c S258$
$S235 \xrightarrow{P8}_c S262$	$S235 \xrightarrow{P14(2)}_c S263$	$S236 \xrightarrow{M12}_c S259$	$S236 \xrightarrow{TopL}_c S264$	$S237 \xrightarrow{M12}_c S260$
$S237 \xrightarrow{P20}_c S265$	$S238 \xrightarrow{M12}_c S261$	$S238 \xrightarrow{P21}_c S266$	$S239 \xrightarrow{M7}_c S263$	$S239 \xrightarrow{TopL}_c S267$
$S240 \xrightarrow{M7}_c S264$	$S240 \xrightarrow{P20}_c S268$	$S241 \xrightarrow{M7}_c S265$	$S241 \xrightarrow{P21}_c S269$	$S242 \xrightarrow{M7}_c S266$
$S242 \xrightarrow{P22}_c S270$	$S243 \xrightarrow{M6}_c S267$	$S243 \xrightarrow{P20}_c S271$	$S244 \xrightarrow{M6}_c S268$	$S244 \xrightarrow{P21}_c S272$
$S245 \xrightarrow{M6}_c S269$	$S245 \xrightarrow{P22}_c S273$	$S246 \xrightarrow{M6}_c S270$	$S246 \xrightarrow{P172}_c S274$	$S247 \xrightarrow{M5}_c S271$
$S247 \xrightarrow{P21}_c S275$	$S248 \xrightarrow{M13}_c S276$	$S248 \xrightarrow{P22}_c S277$	$S249 \xrightarrow{M14}_c S278$	$S249 \xrightarrow{P22}_c S279$
$S250 \xrightarrow{M13}_c S279$	$S250 \xrightarrow{P162}_c S126$	$S251 \xrightarrow{P22}_c S280$	$S252 \xrightarrow{M14}_c S280$	$S252 \xrightarrow{P172}_c S158$
$S253 \xrightarrow{M4}_c S275$	$S253 \xrightarrow{P22}_c S281$	$S254 \xrightarrow{M4}_c S277$	$S254 \xrightarrow{P152}_c S102$	$S255 \xrightarrow{P22}_c S282$
$S256 \xrightarrow{P152}_c S107$	$S257 \xrightarrow{M3}_c S281$	$S257 \xrightarrow{M15}_c S282$	$S257 \xrightarrow{P142}_c S86$	$S258 \xrightarrow{P8}_c S283$
$S258 \xrightarrow{P14(2)}_c S284$	$S259 \xrightarrow{TopL}_c S285$	$S260 \xrightarrow{P20}_c S286$	$S261 \xrightarrow{P21}_c S287$	$S262 \xrightarrow{M12}_c S283$
$S262 \xrightarrow{P9}_c S288$	$S263 \xrightarrow{M12}_c S284$	$S263 \xrightarrow{TopL}_c S289$	$S264 \xrightarrow{M12}_c S285$	$S264 \xrightarrow{P20}_c S290$
$S265 \xrightarrow{M12}_c S286$	$S265 \xrightarrow{P21}_c S291$	$S266 \xrightarrow{M10}_c S292$	$S266 \xrightarrow{M12}_c S287$	$S266 \xrightarrow{P22}_c S293$
$S267 \xrightarrow{M7}_c S289$	$S267 \xrightarrow{P20}_c S294$	$S268 \xrightarrow{M7}_c S290$	$S268 \xrightarrow{P21}_c S295$	$S269 \xrightarrow{M7}_c S291$
$S269 \xrightarrow{P22}_c S296$	$S270 \xrightarrow{M7}_c S293$	$S270 \xrightarrow{P172}_c S297$	$S271 \xrightarrow{M6}_c S294$	$S271 \xrightarrow{P21}_c S298$
$S272 \xrightarrow{M6}_c S295$	$S272 \xrightarrow{P22}_c S299$	$S273 \xrightarrow{M6}_c S296$	$S273 \xrightarrow{P162}_c S274$	$S274 \xrightarrow{M6}_c S297$
$S275 \xrightarrow{M13}_c S300$	$S275 \xrightarrow{P22}_c S301$	$S276 \xrightarrow{M14}_c S302$	$S276 \xrightarrow{P22}_c S303$	$S277 \xrightarrow{M13}_c S303$

Figure 50: Part III of control state transitions for the Inres test case example

$S277 \xrightarrow{P152}_c S126$	$S278 \xrightarrow{P22}_c S304$	$S279 \xrightarrow{M14}_c S304$	$S279 \xrightarrow{P162}_c S158$	$S280 \xrightarrow{P172}_c S190$
$S281 \xrightarrow{M4}_c S301$	$S281 \xrightarrow{P142}_c S102$	$S282 \xrightarrow{P142}_c S107$	$S283 \xrightarrow{P9}_c S305$	$S284 \xrightarrow{TopL}_c S306$
$S285 \xrightarrow{P20}_c S307$	$S286 \xrightarrow{P21}_c S308$	$S287 \xrightarrow{P22}_c S309$	$S288 \xrightarrow{M12}_c S305$	$S288 \xrightarrow{P10}_c S310$
$S288 \xrightarrow{P13(2)}_c S311$	$S289 \xrightarrow{M12}_c S306$	$S289 \xrightarrow{P20}_c S312$	$S290 \xrightarrow{M12}_c S307$	$S290 \xrightarrow{P21}_c S313$
$S291 \xrightarrow{M10}_c S314$	$S291 \xrightarrow{M12}_c S308$	$S291 \xrightarrow{P22}_c S315$	$S292 \xrightarrow{M11}_c S316$	$S292 \xrightarrow{P22}_c S317$
$S293 \xrightarrow{M10}_c S317$	$S293 \xrightarrow{M12}_c S309$	$S293 \xrightarrow{P172}_c S318$	$S294 \xrightarrow{M7}_c S312$	$S294 \xrightarrow{P21}_c S319$
$S295 \xrightarrow{M7}_c S313$	$S295 \xrightarrow{P22}_c S320$	$S296 \xrightarrow{M7}_c S315$	$S296 \xrightarrow{P162}_c S297$	$S297 \xrightarrow{M7}_c S318$
$S298 \xrightarrow{M6}_c S319$	$S298 \xrightarrow{P22}_c S321$	$S299 \xrightarrow{M6}_c S320$	$S299 \xrightarrow{P152}_c S274$	$S300 \xrightarrow{M14}_c S322$
$S300 \xrightarrow{P22}_c S323$	$S301 \xrightarrow{M13}_c S323$	$S301 \xrightarrow{P142}_c S126$	$S302 \xrightarrow{P22}_c S324$	$S303 \xrightarrow{M14}_c S324$
$S303 \xrightarrow{P152}_c S158$	$S304 \xrightarrow{P162}_c S190$	$S305 \xrightarrow{P10}_c S325$	$S305 \xrightarrow{P13(2)}_c S326$	$S306 \xrightarrow{P20}_c S327$
$S307 \xrightarrow{P21}_c S328$	$S308 \xrightarrow{P22}_c S329$	$S309 \xrightarrow{P172}_c S330$	$S310 \xrightarrow{M12}_c S325$	$S310 \xrightarrow{P11}_c S331$
$S311 \xrightarrow{M12}_c S326$	$S311 \xrightarrow{TopL}_c S332$	$S312 \xrightarrow{M12}_c S327$	$S312 \xrightarrow{P21}_c S333$	$S313 \xrightarrow{M10}_c S334$
$S313 \xrightarrow{M12}_c S328$	$S313 \xrightarrow{P22}_c S335$	$S314 \xrightarrow{M11}_c S336$	$S314 \xrightarrow{P22}_c S337$	$S315 \xrightarrow{M10}_c S337$
$S315 \xrightarrow{M12}_c S329$	$S315 \xrightarrow{P162}_c S318$	$S316 \xrightarrow{P22}_c S338$	$S317 \xrightarrow{M11}_c S338$	$S317 \xrightarrow{P172}_c S339$
$S318 \xrightarrow{M10}_c S339$	$S318 \xrightarrow{M12}_c S330$	$S319 \xrightarrow{M7}_c S333$	$S319 \xrightarrow{P22}_c S340$	$S320 \xrightarrow{M7}_c S335$
$S320 \xrightarrow{P152}_c S297$	$S321 \xrightarrow{M6}_c S340$	$S321 \xrightarrow{P142}_c S274$	$S322 \xrightarrow{P22}_c S341$	$S323 \xrightarrow{M14}_c S341$
$S323 \xrightarrow{P142}_c S158$	$S324 \xrightarrow{P152}_c S190$	$S325 \xrightarrow{P11}_c S342$	$S326 \xrightarrow{TopL}_c S343$	$S327 \xrightarrow{P21}_c S344$
$S328 \xrightarrow{P22}_c S345$	$S329 \xrightarrow{P162}_c S330$	$S331 \xrightarrow{M8}_c S346$	$S331 \xrightarrow{M12}_c S342$	$S331 \xrightarrow{P12}_c S347$
$S332 \xrightarrow{M12}_c S343$	$S332 \xrightarrow{P20}_c S348$	$S333 \xrightarrow{M10}_c S349$	$S333 \xrightarrow{M12}_c S344$	$S333 \xrightarrow{P22}_c S350$
$S334 \xrightarrow{M11}_c S351$	$S334 \xrightarrow{P22}_c S352$	$S335 \xrightarrow{M10}_c S352$	$S335 \xrightarrow{M12}_c S345$	$S335 \xrightarrow{P152}_c S318$
$S336 \xrightarrow{P22}_c S353$	$S337 \xrightarrow{M11}_c S353$	$S337 \xrightarrow{P162}_c S339$	$S338 \xrightarrow{P172}_c S354$	$S339 \xrightarrow{M11}_c S354$
$S340 \xrightarrow{M7}_c S350$	$S340 \xrightarrow{P142}_c S297$	$S341 \xrightarrow{P142}_c S190$	$S342 \xrightarrow{P12}_c S355$	$S343 \xrightarrow{P20}_c S356$
$S344 \xrightarrow{P22}_c S357$	$S345 \xrightarrow{P152}_c S330$	$S346 \xrightarrow{M9}_c S358$	$S346 \xrightarrow{P12}_c S359$	$S347 \xrightarrow{M8}_c S359$
$S347 \xrightarrow{M12}_c S355$	$S348 \xrightarrow{M12}_c S356$	$S348 \xrightarrow{P21}_c S360$	$S349 \xrightarrow{M11}_c S361$	$S349 \xrightarrow{P22}_c S362$
$S350 \xrightarrow{M10}_c S362$	$S350 \xrightarrow{M12}_c S357$	$S350 \xrightarrow{P142}_c S318$	$S351 \xrightarrow{P22}_c S363$	$S352 \xrightarrow{M11}_c S363$
$S352 \xrightarrow{P152}_c S339$	$S353 \xrightarrow{P162}_c S354$	$S356 \xrightarrow{P21}_c S364$	$S357 \xrightarrow{P142}_c S330$	$S358 \xrightarrow{P12}_c S190$
$S359 \xrightarrow{M9}_c S190$	$S360 \xrightarrow{M10}_c S365$	$S360 \xrightarrow{M12}_c S364$	$S360 \xrightarrow{P22}_c S366$	$S361 \xrightarrow{P22}_c S367$
$S362 \xrightarrow{M11}_c S367$	$S362 \xrightarrow{P142}_c S339$	$S363 \xrightarrow{P152}_c S354$	$S364 \xrightarrow{P22}_c S368$	$S365 \xrightarrow{M11}_c S369$
$S365 \xrightarrow{P22}_c S370$	$S366 \xrightarrow{M10}_c S370$	$S366 \xrightarrow{M12}_c S368$	$S366 \xrightarrow{P132}_c S371$	$S367 \xrightarrow{P142}_c S354$
$S368 \xrightarrow{P132}_c S107$	$S369 \xrightarrow{P22}_c S372$	$S370 \xrightarrow{M11}_c S372$	$S370 \xrightarrow{P132}_c S373$	$S371 \xrightarrow{M10}_c S373$
$S371 \xrightarrow{M12}_c S107$	$S372 \xrightarrow{P132}_c S190$	$S373 \xrightarrow{M11}_c S190$		

Figure 51: Part IV of control state transitions for the Inres test case example

State	Value		enabled events
	control state	data state	
$ST0$	$((\{MTC\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M1$
$ST1$	$((\{M1\},\emptyset,M1,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, PTC$
$ST2$	$((\{M2\},\emptyset,M1,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, PTC$
$ST3$	$((\{M1,PTC\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, P1, P19(2)$
$ST4$	$((\{M3\},\emptyset,M1,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, PTC$
$ST5$	$((\emptyset,\emptyset,M1,\{\perp,\perp\}),$	$(\propto,\propto,F))$	
$ST6$	$((\{M2,PTC\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, P1, P19(2)$
$ST7$	$((\{M1,P1\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, P2$
$ST8$	$((\{M1,P19(2)\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, TopL$
$ST9$	$((\{M4\},\emptyset,M1,\{\perp,\perp\}),$	$(\propto,\propto,none))$	PTC
$ST10$	$((\{M3,PTC\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, P1, P19(2)$
$ST11$	$((\emptyset,\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,F))$	
$ST12$	$((\{M2,P1\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, P2$
$ST13$	$((\{M2,P19(2)\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, TopL$
$ST14$	$((\{M1,P2\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, P3, P18(2)$
$ST15$	$((\{M1,TopL\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, P20$
$ST16$	$((\{M4,PTC\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$P1, P19(2)$
$ST17$	$((\{M3,P1\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, P2$
$ST18$	$((\{M3,P19(2)\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, TopL$
$ST19$	$((\{M2,P2\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, P3, P18(2)$
$ST20$	$((\emptyset,P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,F))$	
$ST21$	$((\{M2,TopL\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, P20$
$ST22$	$((\{M1,P3\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, P4$
$ST23$	$((\{M1,P18(2)\},P18(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, TopL$
$ST24$	$((\{M1,P20\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, P21$
$ST25$	$((\{M4,P1\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$P2$
$ST26$	$((\{M4,P19(2)\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$TopL$
$ST27$	$((\{M3,P2\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, P3, P18(2)$
$ST28$	$((\{M3,TopL\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, P20$
$ST29$	$((\{M2,P3\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, P4$
$ST30$	$((\{M2,P18(2)\},P18(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, TopL$
$ST31$	$((\{M2,P20\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, P21$
$ST32$	$((\{M1,P4\},\emptyset,\emptyset,\{\perp,P4\}),$	$(\propto,\propto,none))$	$M2, P5, P17(2)$
$ST33$	$((\{M1,TopL\},P18(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M2, P20$
$ST34$	$((\{M1,P21\},P19(2),\emptyset,\{\perp,P21\}),$	$(\propto,\propto,none))$	$M2, P22$
$ST35$	$((\{M4,P2\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$P3, P18(2)$
$ST36$	$((\{M4,TopL\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$P20$
$ST37$	$((\{M3,P3\},\emptyset,\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, P4$
$ST38$	$((\{M3,P18(2)\},P18(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, TopL$
$ST39$	$((\{M3,P20\},P19(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M4, P21$
$ST40$	$((\{M2,P4\},\emptyset,\emptyset,\{\perp,P4\}),$	$(\propto,\propto,none))$	$M3, M15, P5, P17(2)$
$ST41$	$((\emptyset,P18(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,F))$	
$ST42$	$((\{M2,TopL\},P18(2),\emptyset,\{\perp,\perp\}),$	$(\propto,\propto,none))$	$M3, M15, P20$
$ST43$	$((\{M2,P21\},P19(2),\emptyset,\{\perp,P21\}),$	$(\propto,\propto,none))$	$M3, M15, P22$
$ST44$	$((\{M1,P5\},\emptyset,\emptyset,\{\perp,P4\}),$	$(\propto,\propto,none))$	$M2, P6, P16(2)$
$ST45$	$((\{M1,P17(2)\},P17(2),\emptyset,\{\perp,P4\}),$	$(\propto,\propto,none))$	$M2, TopL$

Figure 52: Part I of states for the Inres test case example

State	Value		enabled events
	control state	data state	
$ST46$	$((\{M1, P20\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M2, P21$
$ST47$	$((\{M1, P22\}, P19(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P192$
$ST48$	$((\{M4, P3\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$P4$
$ST49$	$((\{M4, P18(2)\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$TopL$
$ST50$	$((\{M4, P20\}, P19(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$P21$
$ST51$	$((\{M3, P4\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P5, P17(2)$
$ST52$	$((\{M3, TopL\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M4, P20$
$ST53$	$((\{M3, P21\}, P19(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P22$
$ST54$	$((\emptyset, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, F))$	
$ST55$	$((\{M2, P5\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P6, P16(2)$
$ST56$	$((\{M2, P17(2)\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, TopL$
$ST57$	$((\{M2, P20\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M3, M15, P21$
$ST58$	$((\emptyset, P19(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, F))$	
$ST59$	$((\{M2, P22\}, P19(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P192$
$ST60$	$((\{M1, P6\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P7, P15(2)$
$ST61$	$((\{M1, P16(2)\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, TopL$
$ST62$	$((\{M1, TopL\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P20$
$ST63$	$((\{M1, P21\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P22$
$ST64$	$((\{M1\}, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2$
$ST65$	$((\{M4, P4\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P5, P17(2)$
$ST66$	$((\{M4, TopL\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$P20$
$ST67$	$((\{M4, P21\}, P19(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P22$
$ST68$	$((\{M3, P5\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P6, P16(2)$
$ST69$	$((\{M3, P17(2)\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, TopL$
$ST70$	$((\{M3, P20\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M4, P21$
$ST71$	$((\{M3, P22\}, P19(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P192$
$ST72$	$((\{M2, P6\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P7, P15(2)$
$ST73$	$((\{M2, P16(2)\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, TopL$
$ST74$	$((\emptyset, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, F))$	
$ST75$	$((\{M2, TopL\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P20$
$ST76$	$((\{M2, P21\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P22$
$ST77$	$((\{M2\}, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15$
$ST78$	$((\{M1, P7\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P14(2)$
$ST79$	$((\{M1, P15(2)\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, TopL$
$ST80$	$((\{M1, TopL\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P20$
$ST81$	$((\{M1, P20\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P21$
$ST82$	$((\{M1, P22\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P182$
$ST83$	$((\{M5, P4\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P5, P17(2)$
$ST84$	$((\{M4, P5\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P6, P16(2)$
$ST85$	$((\{M4, P17(2)\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, TopL$
$ST86$	$((\{M4, P20\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$P21$
$ST87$	$((\{M13, P21\}, P19(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P22$
$ST88$	$((\{M4, P22\}, P19(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P192$
$ST89$	$((\{M3, P6\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P7, P15(2)$
$ST90$	$((\{M3, P16(2)\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, TopL$
$ST91$	$((\{M3, TopL\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P20$

Figure 53: Part II of states for the Inres test case example

State	Value		enabled events
	control state	data state	
$ST92$	$((\{M3, P21\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P22$
$ST93$	$((\{M3\}, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4$
$ST94$	$((\{M2, P7\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P14(2)$
$ST95$	$((\{M2, P15(2)\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, TopL$
$ST96$	$((\emptyset, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, F))$	
$ST97$	$((\{M2, TopL\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P20$
$ST98$	$((\{M2, P20\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P21$
$ST99$	$((\emptyset, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, F))$	
$ST100$	$((\{M2, P22\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P182$
$ST101$	$((\emptyset, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, \propto, F))$	
$ST102$	$((\{M1, P14(2)\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, TopL$
$ST103$	$((\{M1, TopL\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P20$
$ST104$	$((\{M1, P20\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P21$
$ST105$	$((\{M1, P21\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P22$
$ST106$	$((\{M6, P4\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P5, P17(2)$
$ST107$	$((\{M5, P5\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P6, P16(2)$
$ST108$	$((\{M5, P17(2)\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, TopL$
$ST109$	$((\{M4, P6\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P7, P15(2)$
$ST110$	$((\{M4, P16(2)\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, TopL$
$ST111$	$((\{M4, TopL\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P20$
$ST112$	$((\{M4, P21\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P22$
$ST113$	$((\{M13, P22\}, P19(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P192$
$ST114$	$((\{M4\}, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13$
$ST115$	$((\{M3, P7\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P14(2)$
$ST116$	$((\{M3, P15(2)\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, TopL$
$ST117$	$((\{M3, TopL\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P20$
$ST118$	$((\{M3, P20\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P21$
$ST119$	$((\{M3, P22\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P182$
$ST120$	$((\{M2, P14(2)\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, TopL$
$ST121$	$((\emptyset, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, F))$	
$ST122$	$((\{M2, TopL\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P20$
$ST123$	$((\{M2, P20\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P21$
$ST124$	$((\{M2, P21\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P22$
$ST125$	$((\{M1, TopL\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P20$
$ST126$	$((\{M1, P20\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P21$
$ST127$	$((\{M1, P21\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P22$
$ST128$	$((\{M1, P22\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P172$
$ST129$	$((\{M7, P4\}, \emptyset, \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P5, P17(2)$
$ST130$	$((\{M6, P5\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P6, P16(2)$
$ST131$	$((\{M6, P17(2)\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, TopL$
$ST132$	$((\{M5, P6\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P7, P15(2)$
$ST133$	$((\{M5, P16(2)\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, TopL$
$ST134$	$((\{M5, TopL\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P20$
$ST135$	$((\{M4, P7\}, \emptyset, \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P14(2)$
$ST136$	$((\{M4, P15(2)\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, TopL$
$ST137$	$((\{M4, TopL\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P20$

Figure 54: Part III of states for the Inres test case example

State	Value		enabled events
	control state	data state	
$ST138$	$((\{M4, P20\}, P17(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P21$
$ST139$	$((\{M13, P21\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P22$
$ST140$	$((\{M4, P22\}, P18(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P182$
$ST141$	$((\{M13\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14$
$ST142$	$((\{M3, P14(2)\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, TopL$
$ST143$	$((\{M3, TopL\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P20$
$ST144$	$((\{M3, P20\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P21$
$ST145$	$((\{M3, P21\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P22$
$ST146$	$((\emptyset, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, F))$	
$ST147$	$((\{M2, TopL\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P20$
$ST148$	$((\{M2, P20\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P21$
$ST149$	$((\{M2, P21\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P22$
$ST150$	$((\emptyset, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, F))$	
$ST151$	$((\{M2, P22\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P172$
$ST152$	$((\{M1, P20\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M2, P21$
$ST153$	$((\{M1, P21\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P22$
$ST154$	$((\{M1, P22\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P162$
$ST155$	$((\emptyset, \emptyset, \emptyset, \{M7, \perp\}),$	$(\propto, \propto, F))$	
$ST156$	$((\{M7, P5\}, \emptyset, \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P6, P16(2)$
$ST157$	$((\{M7, P17(2)\}, P17(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, TopL$
$ST158$	$((\{M6, P6\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P7, P15(2)$
$ST159$	$((\{M6, P16(2)\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, TopL$
$ST160$	$((\{M6, TopL\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P20$
$ST161$	$((\{M5, P7\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P14(2)$
$ST162$	$((\{M5, P15(2)\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, TopL$
$ST163$	$((\{M5, TopL\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P20$
$ST164$	$((\{M5, P20\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P21$
$ST165$	$((\{M4, P14(2)\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, TopL$
$ST166$	$((\{M4, TopL\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P20$
$ST167$	$((\{M4, P20\}, P16(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P21$
$ST168$	$((\{M4, P21\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P22$
$ST169$	$((\{M13, P22\}, P18(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P182$
$ST170$	$((\{M3, TopL\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P20$
$ST171$	$((\{M3, P20\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P21$
$ST172$	$((\{M3, P21\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P22$
$ST173$	$((\{M3, P22\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P172$
$ST174$	$((\{M2, P20\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M3, M15, P21$
$ST175$	$((\{M2, P21\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P22$
$ST176$	$((\emptyset, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, F))$	
$ST177$	$((\{M2, P22\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P162$
$ST178$	$((\{M1, P21\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P22$
$ST179$	$((\{M1, P22\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P152$
$ST180$	$((\{M7, P6\}, \emptyset, \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P7, P15(2)$
$ST181$	$((\{M7, P16(2)\}, P16(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, TopL$
$ST182$	$((\emptyset, P17(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, F))$	
$ST183$	$((\{M7, TopL\}, P17(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P20$

Figure 55: Part IV of states for the Inres test case example

State	Value		enabled events
	control state	data state	
$ST184$	$((\{M6, P7\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P14(2)$
$ST185$	$((\{M6, P15(2)\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, TopL$
$ST186$	$((\{M6, TopL\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P20$
$ST187$	$((\{M6, P20\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P21$
$ST188$	$((\{M5, P14(2)\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, TopL$
$ST189$	$((\{M5, TopL\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P20$
$ST190$	$((\{M5, P20\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P21$
$ST191$	$((\{M5, P21\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P22$
$ST192$	$((\{M4, TopL\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P20$
$ST193$	$((\{M4, P20\}, P15(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P21$
$ST194$	$((\{M4, P21\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P22$
$ST195$	$((\{M13, P21\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P22$
$ST196$	$((\{M4, P22\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P172$
$ST197$	$((\{M3, P20\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M4, P21$
$ST198$	$((\{M3, P21\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P22$
$ST199$	$((\{M3, P22\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P162$
$ST200$	$((\{M2, P21\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P22$
$ST201$	$((\emptyset, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, F))$	
$ST202$	$((\{M2, P22\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P152$
$ST203$	$((\{M1, P22\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M2, P142$
$ST204$	$((\{M7, P7\}, \emptyset, \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P8, P14(2)$
$ST205$	$((\{M7, P15(2)\}, P15(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, TopL$
$ST206$	$((\emptyset, P16(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, F))$	
$ST207$	$((\{M7, TopL\}, P16(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P20$
$ST208$	$((\{M7, P20\}, P17(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P21$
$ST209$	$((\{M6, P14(2)\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, TopL$
$ST210$	$((\{M6, TopL\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P20$
$ST211$	$((\{M6, P20\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P21$
$ST212$	$((\{M6, P21\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P22$
$ST213$	$((\{M5, TopL\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P20$
$ST214$	$((\{M5, P20\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P21$
$ST215$	$((\{M5, P21\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P22$
$ST216$	$((\{M5, P22\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P172$
$ST217$	$((\{M4, P20\}, P14(2), \emptyset, \{\perp, P4\}),$	$(\propto, \propto, none))$	$M5, P21$
$ST218$	$((\{M4, P21\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P22$
$ST219$	$((\{M13, P21\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P22$
$ST220$	$((\{M4, P22\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P162$
$ST221$	$((\emptyset, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, F))$	
$ST222$	$((\{M13, P22\}, P17(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P172$
$ST223$	$((\{M3, P21\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P22$
$ST224$	$((\{M3, P22\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P152$
$ST225$	$((\emptyset, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, F))$	
$ST226$	$((\{M2, P22\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M3, M15, P142$
$ST227$	$((\{M7, P8\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, none))$	$M12, P9$
$ST228$	$((\{M7, P14(2)\}, P14(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, TopL$
$ST229$	$((\emptyset, P15(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, F))$	

Figure 56: Part V of states for the Inres test case example

State	Value		enabled events
	control state	data state	
$ST230$	$((\{M7, TopL\}, P15(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P20$
$ST231$	$((\{M7, P20\}, P16(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P21$
$ST232$	$((\{M7, P21\}, P17(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P22$
$ST233$	$((\{M6, TopL\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P20$
$ST234$	$((\{M6, P20\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P21$
$ST235$	$((\{M6, P21\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P22$
$ST236$	$((\{M6, P22\}, P17(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P172$
$ST237$	$((\{M5, P20\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M6, P21$
$ST238$	$((\{M5, P21\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P22$
$ST239$	$((\{M5, P22\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P162$
$ST240$	$((\{M5\}, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6$
$ST241$	$((\{M4, P21\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P22$
$ST242$	$((\{M13, P21\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P22$
$ST243$	$((\{M4, P22\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P152$
$ST244$	$((\emptyset, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, F))$	
$ST245$	$((\{M13, P22\}, P16(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P162$
$ST246$	$((\{M3, P22\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M4, P142$
$ST247$	$((\emptyset, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, F))$	
$ST248$	$((\{M7, P9\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, none))$	$M12, P10, P13(2)$
$ST249$	$((\emptyset, P14(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, F))$	
$ST250$	$((\{M7, TopL\}, P14(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P20$
$ST251$	$((\{M7, P20\}, P15(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P21$
$ST252$	$((\{M7, P21\}, P16(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P22$
$ST253$	$((\{M10, P21\}, P17(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P22$
$ST254$	$((\emptyset, P17(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, F))$	
$ST255$	$((\{M7, P22\}, P17(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P172$
$ST256$	$((\{M6, P20\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, none))$	$M7, P21$
$ST257$	$((\{M6, P21\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P22$
$ST258$	$((\{M6, P22\}, P16(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P162$
$ST259$	$((\{M6\}, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7$
$ST260$	$((\{M5, P21\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P22$
$ST261$	$((\{M5, P22\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P152$
$ST262$	$((\{M13, P21\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P22$
$ST263$	$((\{M4, P22\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M13, P142$
$ST264$	$((\emptyset, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, F))$	
$ST265$	$((\{M13, P22\}, P15(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P152$
$ST266$	$((\{M7, P10\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, none))$	$M12, P11$
$ST267$	$((\{M7, P13(2)\}, P13(2), \emptyset, \{\perp, \perp\}),$	$(\propto, 2, none))$	$M12, TopL$
$ST268$	$((\{M7, P20\}, P14(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, none))$	$M12, P21$
$ST269$	$((\{M7, P21\}, P15(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P22$
$ST270$	$((\{M10, P21\}, P16(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P22$
$ST271$	$((\emptyset, P16(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, F))$	
$ST272$	$((\{M7, P22\}, P16(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P162$
$ST273$	$((\{M10, P22\}, P17(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P172$
$ST274$	$((\{M7\}, \emptyset, \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12$
$ST275$	$((\{M6, P21\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P22$

Figure 57: Part VI of states for the Inres test case example

State	Value		enabled events
	control state	data state	
$ST276$	$((\{M6, P22\}, P15(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P152$
$ST277$	$((\{M5, P22\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M6, P142$
$ST278$	$((\emptyset, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, F))$	
$ST279$	$((\{M13, P22\}, P14(2), \emptyset, \{\perp, \perp\}),$	$(\propto, \propto, (F)))$	$M14, P142$
$ST280$	$((\{M7, P11\}, \emptyset, \emptyset, \{\perp, P11\}),$	$(\propto, 2, none))$	$M8, M12, P12$
$ST281$	$((\emptyset, P13(2), \emptyset, \{\perp, \perp\}),$	$(\propto, 2, F))$	
$ST282$	$((\{M7, TopL\}, P13(2), \emptyset, \{\perp, \perp\}),$	$(\propto, 2, none))$	$M12, P20$
$ST283$	$((\{M7, P21\}, P14(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P22$
$ST284$	$((\{M10, P21\}, P15(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P22$
$ST285$	$((\emptyset, P15(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, F))$	
$ST286$	$((\{M7, P22\}, P15(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P152$
$ST287$	$((\{M10, P22\}, P16(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P162$
$ST288$	$((\{M10\}, \emptyset, \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11$
$ST289$	$((\emptyset, \emptyset, \emptyset, \{M7, P21\}),$	$(\propto, \propto, F))$	
$ST290$	$((\{M6, P22\}, P14(2), \emptyset, \{\perp, P21\}),$	$(\propto, \propto, none))$	$M7, P142$
$ST291$	$((\{M8, P11\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, (P)))$	$M9, P12$
$ST292$	$((\emptyset, \emptyset, \emptyset, \{\perp, P11\}),$	$(\propto, 2, F))$	
$ST293$	$((\{M7\}, \emptyset, \emptyset, \{\perp, P11\}),$	$(\propto, 2, none))$	$M8, M12$
$ST294$	$((\{M7, P20\}, P13(2), \emptyset, \{\perp, \perp\}),$	$(\propto, 2, none))$	$M12, P21$
$ST295$	$((\{M10, P21\}, P14(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P22$
$ST296$	$((\emptyset, P14(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, F))$	
$ST297$	$((\{M7, P22\}, P14(2), \emptyset, \{M7, P21\}),$	$(\propto, \propto, none))$	$M10, M12, P142$
$ST298$	$((\{M10, P22\}, P15(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P152$
$ST299$	$((\emptyset, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, P))$	
$ST300$	$((\{M8\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, (P)))$	$M9$
$ST301$	$((\{M7, P21\}, P13(2), \emptyset, \{\perp, P21\}),$	$(\propto, 2, none))$	$M10, M12, P22$
$ST302$	$((\{M10, P22\}, P14(2), \emptyset, \{M7, \perp\}),$	$(\propto, \propto, (F)))$	$M11, P142$
$ST303$	$((\{M10, P21\}, P13(2), \emptyset, \{\perp, \perp\}),$	$(\propto, 2, (F)))$	$M11, P22$
$ST304$	$((\emptyset, P13(2), \emptyset, \{\perp, P21\}),$	$(\propto, 2, F))$	
$ST305$	$((\{M7, P22\}, P13(2), \emptyset, \{\perp, P21\}),$	$(\propto, 2, none))$	$M10, M12, P132$
$ST306$	$((\{M10, P22\}, P13(2), \emptyset, \{\perp, \perp\}),$	$(\propto, 2, (F)))$	$M11, P132$
$ST307$	$((\{M7\}, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, 2, none))$	$M10, M12$
$ST308$	$((\{M10\}, \emptyset, \emptyset, \{\perp, \perp\}),$	$(\propto, 2, (F)))$	$M11$
$ST309$	$((\emptyset, \emptyset, \emptyset, \{\perp, P21\}),$	$(\propto, 2, F))$	

Figure 58: Part VII of states for the Inres test case example

$ST0 \xrightarrow{M1}_{c,d} ST1$	$ST1 \xrightarrow{M2}_{c,d} ST2$	$ST1 \xrightarrow{PTC}_{c,d} ST3$	$ST2 \xrightarrow{M3}_{c,d} ST4$
$ST2 \xrightarrow{M15}_{c,d} ST5$	$ST2 \xrightarrow{PTC}_{c,d} ST6$	$ST3 \xrightarrow{M2}_{c,d} ST6$	$ST3 \xrightarrow{P1}_{c,d} ST7$
$ST3 \xrightarrow{P19(2)}_{c,d} ST8$	$ST4 \xrightarrow{M4}_{c,d} ST9$	$ST4 \xrightarrow{PTC}_{c,d} ST10$	$ST6 \xrightarrow{M3}_{c,d} ST10$
$ST6 \xrightarrow{M15}_{c,d} ST11$	$ST6 \xrightarrow{P1}_{c,d} ST12$	$ST6 \xrightarrow{P19(2)}_{c,d} ST13$	$ST7 \xrightarrow{M2}_{c,d} ST12$
$ST7 \xrightarrow{P2}_{c,d} ST14$	$ST8 \xrightarrow{M2}_{c,d} ST13$	$ST8 \xrightarrow{TopL}_{c,d} ST15$	$ST9 \xrightarrow{PTC}_{c,d} ST16$
$ST10 \xrightarrow{M4}_{c,d} ST16$	$ST10 \xrightarrow{P1}_{c,d} ST17$	$ST10 \xrightarrow{P19(2)}_{c,d} ST18$	$ST12 \xrightarrow{M3}_{c,d} ST17$
$ST12 \xrightarrow{M15}_{c,d} ST11$	$ST12 \xrightarrow{P2}_{c,d} ST19$	$ST13 \xrightarrow{M3}_{c,d} ST18$	$ST13 \xrightarrow{M15}_{c,d} ST20$
$ST13 \xrightarrow{TopL}_{c,d} ST21$	$ST14 \xrightarrow{M2}_{c,d} ST19$	$ST14 \xrightarrow{P3}_{c,d} ST22$	$ST14 \xrightarrow{P18(2)}_{c,d} ST23$
$ST15 \xrightarrow{M2}_{c,d} ST21$	$ST15 \xrightarrow{P20}_{c,d} ST24$	$ST16 \xrightarrow{P1}_{c,d} ST25$	$ST16 \xrightarrow{P19(2)}_{c,d} ST26$
$ST17 \xrightarrow{M4}_{c,d} ST25$	$ST17 \xrightarrow{P2}_{c,d} ST27$	$ST18 \xrightarrow{M4}_{c,d} ST26$	$ST18 \xrightarrow{TopL}_{c,d} ST28$
$ST19 \xrightarrow{M3}_{c,d} ST27$	$ST19 \xrightarrow{M15}_{c,d} ST11$	$ST19 \xrightarrow{P3}_{c,d} ST29$	$ST19 \xrightarrow{P18(2)}_{c,d} ST30$
$ST21 \xrightarrow{M3}_{c,d} ST28$	$ST21 \xrightarrow{M15}_{c,d} ST20$	$ST21 \xrightarrow{P20}_{c,d} ST31$	$ST22 \xrightarrow{M2}_{c,d} ST29$
$ST22 \xrightarrow{P4}_{c,d} ST32$	$ST23 \xrightarrow{M2}_{c,d} ST30$	$ST23 \xrightarrow{TopL}_{c,d} ST33$	$ST24 \xrightarrow{M2}_{c,d} ST31$
$ST24 \xrightarrow{P21}_{c,d} ST34$	$ST25 \xrightarrow{P2}_{c,d} ST35$	$ST26 \xrightarrow{TopL}_{c,d} ST36$	$ST27 \xrightarrow{M4}_{c,d} ST35$
$ST27 \xrightarrow{P3}_{c,d} ST37$	$ST27 \xrightarrow{P18(2)}_{c,d} ST38$	$ST28 \xrightarrow{M4}_{c,d} ST36$	$ST28 \xrightarrow{P20}_{c,d} ST39$
$ST29 \xrightarrow{M3}_{c,d} ST37$	$ST29 \xrightarrow{M15}_{c,d} ST11$	$ST29 \xrightarrow{P4}_{c,d} ST40$	$ST30 \xrightarrow{M3}_{c,d} ST38$
$ST30 \xrightarrow{M15}_{c,d} ST41$	$ST30 \xrightarrow{TopL}_{c,d} ST42$	$ST31 \xrightarrow{M3}_{c,d} ST39$	$ST31 \xrightarrow{M15}_{c,d} ST20$
$ST31 \xrightarrow{P21}_{c,d} ST43$	$ST32 \xrightarrow{M2}_{c,d} ST40$	$ST32 \xrightarrow{P5}_{c,d} ST44$	$ST32 \xrightarrow{P17(2)}_{c,d} ST45$
$ST33 \xrightarrow{M2}_{c,d} ST42$	$ST33 \xrightarrow{P20}_{c,d} ST46$	$ST34 \xrightarrow{M2}_{c,d} ST43$	$ST34 \xrightarrow{P22}_{c,d} ST47$
$ST35 \xrightarrow{P3}_{c,d} ST48$	$ST35 \xrightarrow{P18(2)}_{c,d} ST49$	$ST36 \xrightarrow{P20}_{c,d} ST50$	$ST37 \xrightarrow{M4}_{c,d} ST48$
$ST37 \xrightarrow{P4}_{c,d} ST51$	$ST38 \xrightarrow{M4}_{c,d} ST49$	$ST38 \xrightarrow{TopL}_{c,d} ST52$	$ST39 \xrightarrow{M4}_{c,d} ST50$
$ST39 \xrightarrow{P21}_{c,d} ST53$	$ST40 \xrightarrow{M3}_{c,d} ST51$	$ST40 \xrightarrow{M15}_{c,d} ST54$	$ST40 \xrightarrow{P5}_{c,d} ST55$
$ST40 \xrightarrow{P17(2)}_{c,d} ST56$	$ST42 \xrightarrow{M3}_{c,d} ST52$	$ST42 \xrightarrow{M15}_{c,d} ST41$	$ST42 \xrightarrow{P20}_{c,d} ST57$
$ST43 \xrightarrow{M3}_{c,d} ST53$	$ST43 \xrightarrow{M15}_{c,d} ST58$	$ST43 \xrightarrow{P22}_{c,d} ST59$	$ST44 \xrightarrow{M2}_{c,d} ST55$
$ST44 \xrightarrow{P6}_{c,d} ST60$	$ST44 \xrightarrow{P16(2)}_{c,d} ST61$	$ST45 \xrightarrow{M2}_{c,d} ST56$	$ST45 \xrightarrow{TopL}_{c,d} ST62$
$ST46 \xrightarrow{M2}_{c,d} ST57$	$ST46 \xrightarrow{P21}_{c,d} ST63$	$ST47 \xrightarrow{M2}_{c,d} ST59$	$ST47 \xrightarrow{P192}_{c,d} ST64$
$ST48 \xrightarrow{P4}_{c,d} ST65$	$ST49 \xrightarrow{TopL}_{c,d} ST66$	$ST50 \xrightarrow{P21}_{c,d} ST67$	$ST51 \xrightarrow{M4}_{c,d} ST65$
$ST51 \xrightarrow{P5}_{c,d} ST68$	$ST51 \xrightarrow{P17(2)}_{c,d} ST69$	$ST52 \xrightarrow{M4}_{c,d} ST66$	$ST52 \xrightarrow{P20}_{c,d} ST70$
$ST53 \xrightarrow{M4}_{c,d} ST67$	$ST53 \xrightarrow{P22}_{c,d} ST71$	$ST55 \xrightarrow{M3}_{c,d} ST68$	$ST55 \xrightarrow{M15}_{c,d} ST54$
$ST55 \xrightarrow{P6}_{c,d} ST72$	$ST55 \xrightarrow{P16(2)}_{c,d} ST73$	$ST56 \xrightarrow{M3}_{c,d} ST69$	$ST56 \xrightarrow{M15}_{c,d} ST74$
$ST56 \xrightarrow{TopL}_{c,d} ST75$	$ST57 \xrightarrow{M3}_{c,d} ST70$	$ST57 \xrightarrow{M15}_{c,d} ST41$	$ST57 \xrightarrow{P21}_{c,d} ST76$
$ST59 \xrightarrow{M3}_{c,d} ST71$	$ST59 \xrightarrow{M15}_{c,d} ST58$	$ST59 \xrightarrow{P192}_{c,d} ST77$	$ST60 \xrightarrow{M2}_{c,d} ST72$
$ST60 \xrightarrow{P7}_{c,d} ST78$	$ST60 \xrightarrow{P15(2)}_{c,d} ST79$	$ST61 \xrightarrow{M2}_{c,d} ST73$	$ST61 \xrightarrow{TopL}_{c,d} ST80$
$ST62 \xrightarrow{M2}_{c,d} ST75$	$ST62 \xrightarrow{P20}_{c,d} ST81$	$ST63 \xrightarrow{M2}_{c,d} ST76$	$ST63 \xrightarrow{P22}_{c,d} ST82$
$ST64 \xrightarrow{M2}_{c,d} ST77$	$ST65 \xrightarrow{M5}_{c,d} ST83$	$ST65 \xrightarrow{P5}_{c,d} ST84$	$ST65 \xrightarrow{P17(2)}_{c,d} ST85$
$ST66 \xrightarrow{P20}_{c,d} ST86$	$ST67 \xrightarrow{M13}_{c,d} ST87$	$ST67 \xrightarrow{P22}_{c,d} ST88$	$ST68 \xrightarrow{M4}_{c,d} ST84$
$ST68 \xrightarrow{P6}_{c,d} ST89$	$ST68 \xrightarrow{P16(2)}_{c,d} ST90$	$ST69 \xrightarrow{M4}_{c,d} ST85$	$ST69 \xrightarrow{TopL}_{c,d} ST91$

Figure 59: Part I of state transitions for the Inres test case example

$ST70 \xrightarrow{M4}_{c,d} ST86$	$ST70 \xrightarrow{P21}_{c,d} ST92$	$ST71 \xrightarrow{M4}_{c,d} ST88$	$ST71 \xrightarrow{P192}_{c,d} ST93$
$ST72 \xrightarrow{M3}_{c,d} ST89$	$ST72 \xrightarrow{M15}_{c,d} ST54$	$ST72 \xrightarrow{P7}_{c,d} ST94$	$ST72 \xrightarrow{P15(2)}_{c,d} ST95$
$ST73 \xrightarrow{M3}_{c,d} ST90$	$ST73 \xrightarrow{M15}_{c,d} ST96$	$ST73 \xrightarrow{TopL}_{c,d} ST97$	$ST75 \xrightarrow{M3}_{c,d} ST91$
$ST75 \xrightarrow{M15}_{c,d} ST74$	$ST75 \xrightarrow{P20}_{c,d} ST98$	$ST76 \xrightarrow{M3}_{c,d} ST92$	$ST76 \xrightarrow{M15}_{c,d} ST99$
$ST76 \xrightarrow{P22}_{c,d} ST100$	$ST77 \xrightarrow{M3}_{c,d} ST93$	$ST77 \xrightarrow{M15}_{c,d} ST101$	$ST78 \xrightarrow{M2}_{c,d} ST94$
$ST78 \xrightarrow{P14(2)}_{c,d} ST102$	$ST79 \xrightarrow{M2}_{c,d} ST95$	$ST79 \xrightarrow{TopL}_{c,d} ST103$	$ST80 \xrightarrow{M2}_{c,d} ST97$
$ST80 \xrightarrow{P20}_{c,d} ST104$	$ST81 \xrightarrow{M2}_{c,d} ST98$	$ST81 \xrightarrow{P21}_{c,d} ST105$	$ST82 \xrightarrow{M2}_{c,d} ST100$
$ST82 \xrightarrow{P182}_{c,d} ST64$	$ST83 \xrightarrow{M6}_{c,d} ST106$	$ST83 \xrightarrow{P5}_{c,d} ST107$	$ST83 \xrightarrow{P17(2)}_{c,d} ST108$
$ST84 \xrightarrow{M5}_{c,d} ST107$	$ST84 \xrightarrow{P6}_{c,d} ST109$	$ST84 \xrightarrow{P16(2)}_{c,d} ST110$	$ST85 \xrightarrow{M5}_{c,d} ST108$
$ST85 \xrightarrow{TopL}_{c,d} ST111$	$ST86 \xrightarrow{P21}_{c,d} ST112$	$ST87 \xrightarrow{M14}_{c,d} ST20$	$ST87 \xrightarrow{P22}_{c,d} ST113$
$ST88 \xrightarrow{M13}_{c,d} ST113$	$ST88 \xrightarrow{P192}_{c,d} ST114$	$ST89 \xrightarrow{M4}_{c,d} ST109$	$ST89 \xrightarrow{P7}_{c,d} ST115$
$ST89 \xrightarrow{P15(2)}_{c,d} ST116$	$ST90 \xrightarrow{M4}_{c,d} ST110$	$ST90 \xrightarrow{TopL}_{c,d} ST117$	$ST91 \xrightarrow{M4}_{c,d} ST111$
$ST91 \xrightarrow{P20}_{c,d} ST118$	$ST92 \xrightarrow{M4}_{c,d} ST112$	$ST92 \xrightarrow{P22}_{c,d} ST119$	$ST93 \xrightarrow{M4}_{c,d} ST114$
$ST94 \xrightarrow{M3}_{c,d} ST115$	$ST94 \xrightarrow{M15}_{c,d} ST54$	$ST94 \xrightarrow{P14(2)}_{c,d} ST120$	$ST95 \xrightarrow{M3}_{c,d} ST116$
$ST95 \xrightarrow{M15}_{c,d} ST121$	$ST95 \xrightarrow{TopL}_{c,d} ST122$	$ST97 \xrightarrow{M3}_{c,d} ST117$	$ST97 \xrightarrow{M15}_{c,d} ST96$
$ST97 \xrightarrow{P20}_{c,d} ST123$	$ST98 \xrightarrow{M3}_{c,d} ST118$	$ST98 \xrightarrow{M15}_{c,d} ST74$	$ST98 \xrightarrow{P21}_{c,d} ST124$
$ST100 \xrightarrow{M3}_{c,d} ST119$	$ST100 \xrightarrow{M15}_{c,d} ST99$	$ST100 \xrightarrow{P182}_{c,d} ST77$	$ST102 \xrightarrow{M2}_{c,d} ST120$
$ST102 \xrightarrow{TopL}_{c,d} ST125$	$ST103 \xrightarrow{M2}_{c,d} ST122$	$ST103 \xrightarrow{P20}_{c,d} ST126$	$ST104 \xrightarrow{M2}_{c,d} ST123$
$ST104 \xrightarrow{P21}_{c,d} ST127$	$ST105 \xrightarrow{M2}_{c,d} ST124$	$ST105 \xrightarrow{P22}_{c,d} ST128$	$ST106 \xrightarrow{M7}_{c,d} ST129$
$ST106 \xrightarrow{P5}_{c,d} ST130$	$ST106 \xrightarrow{P17(2)}_{c,d} ST131$	$ST107 \xrightarrow{M6}_{c,d} ST130$	$ST107 \xrightarrow{P6}_{c,d} ST132$
$ST107 \xrightarrow{P16(2)}_{c,d} ST133$	$ST108 \xrightarrow{M6}_{c,d} ST131$	$ST108 \xrightarrow{TopL}_{c,d} ST134$	$ST109 \xrightarrow{M5}_{c,d} ST132$
$ST109 \xrightarrow{P7}_{c,d} ST135$	$ST109 \xrightarrow{P15(2)}_{c,d} ST136$	$ST110 \xrightarrow{M5}_{c,d} ST133$	$ST110 \xrightarrow{TopL}_{c,d} ST137$
$ST111 \xrightarrow{M5}_{c,d} ST134$	$ST111 \xrightarrow{P20}_{c,d} ST138$	$ST112 \xrightarrow{M13}_{c,d} ST139$	$ST112 \xrightarrow{P22}_{c,d} ST140$
$ST113 \xrightarrow{M14}_{c,d} ST20$	$ST113 \xrightarrow{P192}_{c,d} ST141$	$ST114 \xrightarrow{M13}_{c,d} ST141$	$ST115 \xrightarrow{M4}_{c,d} ST135$
$ST115 \xrightarrow{P14(2)}_{c,d} ST142$	$ST116 \xrightarrow{M4}_{c,d} ST136$	$ST116 \xrightarrow{TopL}_{c,d} ST143$	$ST117 \xrightarrow{M4}_{c,d} ST137$
$ST117 \xrightarrow{P20}_{c,d} ST144$	$ST118 \xrightarrow{M4}_{c,d} ST138$	$ST118 \xrightarrow{P21}_{c,d} ST145$	$ST119 \xrightarrow{M4}_{c,d} ST140$
$ST119 \xrightarrow{P182}_{c,d} ST93$	$ST120 \xrightarrow{M3}_{c,d} ST142$	$ST120 \xrightarrow{M15}_{c,d} ST146$	$ST120 \xrightarrow{TopL}_{c,d} ST147$
$ST122 \xrightarrow{M3}_{c,d} ST143$	$ST122 \xrightarrow{M15}_{c,d} ST121$	$ST122 \xrightarrow{P20}_{c,d} ST148$	$ST123 \xrightarrow{M3}_{c,d} ST144$
$ST123 \xrightarrow{M15}_{c,d} ST96$	$ST123 \xrightarrow{P21}_{c,d} ST149$	$ST124 \xrightarrow{M3}_{c,d} ST145$	$ST124 \xrightarrow{M15}_{c,d} ST150$
$ST124 \xrightarrow{P22}_{c,d} ST151$	$ST125 \xrightarrow{M2}_{c,d} ST147$	$ST125 \xrightarrow{P20}_{c,d} ST152$	$ST126 \xrightarrow{M2}_{c,d} ST148$
$ST126 \xrightarrow{P21}_{c,d} ST153$	$ST127 \xrightarrow{M2}_{c,d} ST149$	$ST127 \xrightarrow{P22}_{c,d} ST154$	$ST128 \xrightarrow{M2}_{c,d} ST151$
$ST128 \xrightarrow{P172}_{c,d} ST64$	$ST129 \xrightarrow{M12}_{c,d} ST155$	$ST129 \xrightarrow{P5}_{c,d} ST156$	$ST129 \xrightarrow{P17(2)}_{c,d} ST157$
$ST130 \xrightarrow{M7}_{c,d} ST156$	$ST130 \xrightarrow{P6}_{c,d} ST158$	$ST130 \xrightarrow{P16(2)}_{c,d} ST159$	$ST131 \xrightarrow{M7}_{c,d} ST157$
$ST131 \xrightarrow{TopL}_{c,d} ST160$	$ST132 \xrightarrow{M6}_{c,d} ST158$	$ST132 \xrightarrow{P7}_{c,d} ST161$	$ST132 \xrightarrow{P15(2)}_{c,d} ST162$
$ST133 \xrightarrow{M6}_{c,d} ST159$	$ST133 \xrightarrow{TopL}_{c,d} ST163$	$ST134 \xrightarrow{M6}_{c,d} ST160$	$ST134 \xrightarrow{P20}_{c,d} ST164$
$ST135 \xrightarrow{M5}_{c,d} ST161$	$ST135 \xrightarrow{P14(2)}_{c,d} ST165$	$ST136 \xrightarrow{M5}_{c,d} ST162$	$ST136 \xrightarrow{TopL}_{c,d} ST166$
$ST137 \xrightarrow{M5}_{c,d} ST163$	$ST137 \xrightarrow{P20}_{c,d} ST167$	$ST138 \xrightarrow{M5}_{c,d} ST164$	$ST138 \xrightarrow{P21}_{c,d} ST168$

Figure 60: Part II of state transitions for the Inres test case example

$ST139 \xrightarrow{M14}_{c,d} ST41$	$ST139 \xrightarrow{P22}_{c,d} ST169$	$ST140 \xrightarrow{M13}_{c,d} ST169$	$ST140 \xrightarrow{P182}_{c,d} ST114$
$ST141 \xrightarrow{M14}_{c,d} ST11$	$ST142 \xrightarrow{M4}_{c,d} ST165$	$ST142 \xrightarrow{TopL}_{c,d} ST170$	$ST143 \xrightarrow{M4}_{c,d} ST166$
$ST143 \xrightarrow{P20}_{c,d} ST171$	$ST144 \xrightarrow{M4}_{c,d} ST167$	$ST144 \xrightarrow{P21}_{c,d} ST172$	$ST145 \xrightarrow{M4}_{c,d} ST168$
$ST145 \xrightarrow{P22}_{c,d} ST173$	$ST147 \xrightarrow{M3}_{c,d} ST170$	$ST147 \xrightarrow{M15}_{c,d} ST146$	$ST147 \xrightarrow{P20}_{c,d} ST174$
$ST148 \xrightarrow{M3}_{c,d} ST171$	$ST148 \xrightarrow{M15}_{c,d} ST121$	$ST148 \xrightarrow{P21}_{c,d} ST175$	$ST149 \xrightarrow{M3}_{c,d} ST172$
$ST149 \xrightarrow{M15}_{c,d} ST176$	$ST149 \xrightarrow{P22}_{c,d} ST177$	$ST151 \xrightarrow{M3}_{c,d} ST173$	$ST151 \xrightarrow{M15}_{c,d} ST150$
$ST151 \xrightarrow{P172}_{c,d} ST77$	$ST152 \xrightarrow{M2}_{c,d} ST174$	$ST152 \xrightarrow{P21}_{c,d} ST178$	$ST153 \xrightarrow{M2}_{c,d} ST175$
$ST153 \xrightarrow{P22}_{c,d} ST179$	$ST154 \xrightarrow{M2}_{c,d} ST177$	$ST154 \xrightarrow{P162}_{c,d} ST64$	$ST156 \xrightarrow{M12}_{c,d} ST155$
$ST156 \xrightarrow{P6}_{c,d} ST180$	$ST156 \xrightarrow{P16(2)}_{c,d} ST181$	$ST157 \xrightarrow{M12}_{c,d} ST182$	$ST157 \xrightarrow{TopL}_{c,d} ST183$
$ST158 \xrightarrow{M7}_{c,d} ST180$	$ST158 \xrightarrow{P7}_{c,d} ST184$	$ST158 \xrightarrow{P15(2)}_{c,d} ST185$	$ST159 \xrightarrow{M7}_{c,d} ST181$
$ST159 \xrightarrow{TopL}_{c,d} ST186$	$ST160 \xrightarrow{M7}_{c,d} ST183$	$ST160 \xrightarrow{P20}_{c,d} ST187$	$ST161 \xrightarrow{M6}_{c,d} ST184$
$ST161 \xrightarrow{P14(2)}_{c,d} ST188$	$ST162 \xrightarrow{M6}_{c,d} ST185$	$ST162 \xrightarrow{TopL}_{c,d} ST189$	$ST163 \xrightarrow{M6}_{c,d} ST186$
$ST163 \xrightarrow{P20}_{c,d} ST190$	$ST164 \xrightarrow{M6}_{c,d} ST187$	$ST164 \xrightarrow{P21}_{c,d} ST191$	$ST165 \xrightarrow{M5}_{c,d} ST188$
$ST165 \xrightarrow{TopL}_{c,d} ST192$	$ST166 \xrightarrow{M5}_{c,d} ST189$	$ST166 \xrightarrow{P20}_{c,d} ST193$	$ST167 \xrightarrow{M5}_{c,d} ST190$
$ST167 \xrightarrow{P21}_{c,d} ST194$	$ST168 \xrightarrow{M13}_{c,d} ST195$	$ST168 \xrightarrow{P22}_{c,d} ST196$	$ST169 \xrightarrow{M14}_{c,d} ST41$
$ST169 \xrightarrow{P182}_{c,d} ST141$	$ST170 \xrightarrow{M4}_{c,d} ST192$	$ST170 \xrightarrow{P20}_{c,d} ST197$	$ST171 \xrightarrow{M4}_{c,d} ST193$
$ST171 \xrightarrow{P21}_{c,d} ST198$	$ST172 \xrightarrow{M4}_{c,d} ST194$	$ST172 \xrightarrow{P22}_{c,d} ST199$	$ST173 \xrightarrow{M4}_{c,d} ST196$
$ST173 \xrightarrow{P172}_{c,d} ST93$	$ST174 \xrightarrow{M3}_{c,d} ST197$	$ST174 \xrightarrow{M15}_{c,d} ST146$	$ST174 \xrightarrow{P21}_{c,d} ST200$
$ST175 \xrightarrow{M3}_{c,d} ST198$	$ST175 \xrightarrow{M15}_{c,d} ST201$	$ST175 \xrightarrow{P22}_{c,d} ST202$	$ST177 \xrightarrow{M3}_{c,d} ST199$
$ST177 \xrightarrow{M15}_{c,d} ST176$	$ST177 \xrightarrow{P162}_{c,d} ST77$	$ST178 \xrightarrow{M2}_{c,d} ST200$	$ST178 \xrightarrow{P22}_{c,d} ST203$
$ST179 \xrightarrow{M2}_{c,d} ST202$	$ST179 \xrightarrow{P152}_{c,d} ST64$	$ST180 \xrightarrow{M12}_{c,d} ST155$	$ST180 \xrightarrow{P7}_{c,d} ST204$
$ST180 \xrightarrow{P15(2)}_{c,d} ST205$	$ST181 \xrightarrow{M12}_{c,d} ST206$	$ST181 \xrightarrow{TopL}_{c,d} ST207$	$ST183 \xrightarrow{M12}_{c,d} ST182$
$ST183 \xrightarrow{P20}_{c,d} ST208$	$ST184 \xrightarrow{M7}_{c,d} ST204$	$ST184 \xrightarrow{P14(2)}_{c,d} ST209$	$ST185 \xrightarrow{M7}_{c,d} ST205$
$ST185 \xrightarrow{TopL}_{c,d} ST210$	$ST186 \xrightarrow{M7}_{c,d} ST207$	$ST186 \xrightarrow{P20}_{c,d} ST211$	$ST187 \xrightarrow{M7}_{c,d} ST208$
$ST187 \xrightarrow{P21}_{c,d} ST212$	$ST188 \xrightarrow{M6}_{c,d} ST209$	$ST188 \xrightarrow{TopL}_{c,d} ST213$	$ST189 \xrightarrow{M6}_{c,d} ST210$
$ST189 \xrightarrow{P20}_{c,d} ST214$	$ST190 \xrightarrow{M6}_{c,d} ST211$	$ST190 \xrightarrow{P21}_{c,d} ST215$	$ST191 \xrightarrow{M6}_{c,d} ST212$
$ST191 \xrightarrow{P22}_{c,d} ST216$	$ST192 \xrightarrow{M5}_{c,d} ST213$	$ST192 \xrightarrow{P20}_{c,d} ST217$	$ST193 \xrightarrow{M5}_{c,d} ST214$
$ST193 \xrightarrow{P21}_{c,d} ST218$	$ST194 \xrightarrow{M13}_{c,d} ST219$	$ST194 \xrightarrow{P22}_{c,d} ST220$	$ST195 \xrightarrow{M14}_{c,d} ST221$
$ST195 \xrightarrow{P22}_{c,d} ST222$	$ST196 \xrightarrow{M13}_{c,d} ST222$	$ST196 \xrightarrow{P172}_{c,d} ST114$	$ST197 \xrightarrow{M4}_{c,d} ST217$
$ST197 \xrightarrow{P21}_{c,d} ST223$	$ST198 \xrightarrow{M4}_{c,d} ST218$	$ST198 \xrightarrow{P22}_{c,d} ST224$	$ST199 \xrightarrow{M4}_{c,d} ST220$
$ST199 \xrightarrow{P162}_{c,d} ST93$	$ST200 \xrightarrow{M3}_{c,d} ST223$	$ST200 \xrightarrow{M15}_{c,d} ST225$	$ST200 \xrightarrow{P22}_{c,d} ST226$
$ST202 \xrightarrow{M3}_{c,d} ST224$	$ST202 \xrightarrow{M15}_{c,d} ST201$	$ST202 \xrightarrow{P152}_{c,d} ST77$	$ST203 \xrightarrow{M2}_{c,d} ST226$
$ST203 \xrightarrow{P142}_{c,d} ST64$	$ST204 \xrightarrow{M12}_{c,d} ST155$	$ST204 \xrightarrow{P8}_{c,d} ST227$	$ST204 \xrightarrow{P14(2)}_{c,d} ST228$
$ST205 \xrightarrow{M12}_{c,d} ST229$	$ST205 \xrightarrow{TopL}_{c,d} ST230$	$ST207 \xrightarrow{M12}_{c,d} ST206$	$ST207 \xrightarrow{P20}_{c,d} ST231$
$ST208 \xrightarrow{M12}_{c,d} ST182$	$ST208 \xrightarrow{P21}_{c,d} ST232$	$ST209 \xrightarrow{M7}_{c,d} ST228$	$ST209 \xrightarrow{TopL}_{c,d} ST233$
$ST210 \xrightarrow{M7}_{c,d} ST230$	$ST210 \xrightarrow{P20}_{c,d} ST234$	$ST211 \xrightarrow{M7}_{c,d} ST231$	$ST211 \xrightarrow{P21}_{c,d} ST235$
$ST212 \xrightarrow{M7}_{c,d} ST232$	$ST212 \xrightarrow{P22}_{c,d} ST236$	$ST213 \xrightarrow{M6}_{c,d} ST233$	$ST213 \xrightarrow{P20}_{c,d} ST237$

Figure 61: Part III of state transitions for the Inres test case example

$ST214 \xrightarrow{M6}_{c,d} ST234$	$ST214 \xrightarrow{P21}_{c,d} ST238$	$ST215 \xrightarrow{M6}_{c,d} ST235$	$ST215 \xrightarrow{P22}_{c,d} ST239$
$ST216 \xrightarrow{M6}_{c,d} ST236$	$ST216 \xrightarrow{P172}_{c,d} ST240$	$ST217 \xrightarrow{M5}_{c,d} ST237$	$ST217 \xrightarrow{P21}_{c,d} ST241$
$ST218 \xrightarrow{M13}_{c,d} ST242$	$ST218 \xrightarrow{P22}_{c,d} ST243$	$ST219 \xrightarrow{M14}_{c,d} ST244$	$ST219 \xrightarrow{P22}_{c,d} ST245$
$ST220 \xrightarrow{M13}_{c,d} ST245$	$ST220 \xrightarrow{P162}_{c,d} ST114$	$ST222 \xrightarrow{M14}_{c,d} ST221$	$ST222 \xrightarrow{P172}_{c,d} ST141$
$ST223 \xrightarrow{M4}_{c,d} ST241$	$ST223 \xrightarrow{P22}_{c,d} ST246$	$ST224 \xrightarrow{M4}_{c,d} ST243$	$ST224 \xrightarrow{P152}_{c,d} ST93$
$ST226 \xrightarrow{M3}_{c,d} ST246$	$ST226 \xrightarrow{M15}_{c,d} ST225$	$ST226 \xrightarrow{P142}_{c,d} ST77$	$ST227 \xrightarrow{M12}_{c,d} ST247$
$ST227 \xrightarrow{P9}_{c,d} ST248$	$ST228 \xrightarrow{M12}_{c,d} ST249$	$ST228 \xrightarrow{TopL}_{c,d} ST250$	$ST230 \xrightarrow{M12}_{c,d} ST229$
$ST230 \xrightarrow{P20}_{c,d} ST251$	$ST231 \xrightarrow{M12}_{c,d} ST206$	$ST231 \xrightarrow{P21}_{c,d} ST252$	$ST232 \xrightarrow{M10}_{c,d} ST253$
$ST232 \xrightarrow{M12}_{c,d} ST254$	$ST232 \xrightarrow{P22}_{c,d} ST255$	$ST233 \xrightarrow{M7}_{c,d} ST250$	$ST233 \xrightarrow{P20}_{c,d} ST256$
$ST234 \xrightarrow{M7}_{c,d} ST251$	$ST234 \xrightarrow{P21}_{c,d} ST257$	$ST235 \xrightarrow{M7}_{c,d} ST252$	$ST235 \xrightarrow{P22}_{c,d} ST258$
$ST236 \xrightarrow{M7}_{c,d} ST255$	$ST236 \xrightarrow{P172}_{c,d} ST259$	$ST237 \xrightarrow{M6}_{c,d} ST256$	$ST237 \xrightarrow{P21}_{c,d} ST260$
$ST238 \xrightarrow{M6}_{c,d} ST257$	$ST238 \xrightarrow{P22}_{c,d} ST261$	$ST239 \xrightarrow{M6}_{c,d} ST258$	$ST239 \xrightarrow{P162}_{c,d} ST240$
$ST240 \xrightarrow{M6}_{c,d} ST259$	$ST241 \xrightarrow{M13}_{c,d} ST262$	$ST241 \xrightarrow{P22}_{c,d} ST263$	$ST242 \xrightarrow{M14}_{c,d} ST264$
$ST242 \xrightarrow{P22}_{c,d} ST265$	$ST243 \xrightarrow{M13}_{c,d} ST265$	$ST243 \xrightarrow{P152}_{c,d} ST114$	$ST245 \xrightarrow{M14}_{c,d} ST244$
$ST245 \xrightarrow{P162}_{c,d} ST141$	$ST246 \xrightarrow{M4}_{c,d} ST263$	$ST246 \xrightarrow{P142}_{c,d} ST93$	$ST248 \xrightarrow{M12}_{c,d} ST247$
$ST248 \xrightarrow{P10}_{c,d} ST266$	$ST248 \xrightarrow{P13(2)}_{c,d} ST267$	$ST250 \xrightarrow{M12}_{c,d} ST249$	$ST250 \xrightarrow{P20}_{c,d} ST268$
$ST251 \xrightarrow{M12}_{c,d} ST229$	$ST251 \xrightarrow{P21}_{c,d} ST269$	$ST252 \xrightarrow{M10}_{c,d} ST270$	$ST252 \xrightarrow{M12}_{c,d} ST271$
$ST252 \xrightarrow{P22}_{c,d} ST272$	$ST253 \xrightarrow{M11}_{c,d} ST182$	$ST253 \xrightarrow{P22}_{c,d} ST273$	$ST255 \xrightarrow{M10}_{c,d} ST273$
$ST255 \xrightarrow{M12}_{c,d} ST254$	$ST255 \xrightarrow{P172}_{c,d} ST274$	$ST256 \xrightarrow{M7}_{c,d} ST268$	$ST256 \xrightarrow{P21}_{c,d} ST275$
$ST257 \xrightarrow{M7}_{c,d} ST269$	$ST257 \xrightarrow{P22}_{c,d} ST276$	$ST258 \xrightarrow{M7}_{c,d} ST272$	$ST258 \xrightarrow{P162}_{c,d} ST259$
$ST259 \xrightarrow{M7}_{c,d} ST274$	$ST260 \xrightarrow{M6}_{c,d} ST275$	$ST260 \xrightarrow{P22}_{c,d} ST277$	$ST261 \xrightarrow{M6}_{c,d} ST276$
$ST261 \xrightarrow{P152}_{c,d} ST240$	$ST262 \xrightarrow{M14}_{c,d} ST278$	$ST262 \xrightarrow{P22}_{c,d} ST279$	$ST263 \xrightarrow{M13}_{c,d} ST279$
$ST263 \xrightarrow{P142}_{c,d} ST114$	$ST265 \xrightarrow{M14}_{c,d} ST264$	$ST265 \xrightarrow{P152}_{c,d} ST141$	$ST266 \xrightarrow{M12}_{c,d} ST247$
$ST266 \xrightarrow{P11}_{c,d} ST280$	$ST267 \xrightarrow{M12}_{c,d} ST281$	$ST267 \xrightarrow{TopL}_{c,d} ST282$	$ST268 \xrightarrow{M12}_{c,d} ST249$
$ST268 \xrightarrow{P21}_{c,d} ST283$	$ST269 \xrightarrow{M10}_{c,d} ST284$	$ST269 \xrightarrow{M12}_{c,d} ST285$	$ST269 \xrightarrow{P22}_{c,d} ST286$
$ST270 \xrightarrow{M11}_{c,d} ST206$	$ST270 \xrightarrow{P22}_{c,d} ST287$	$ST272 \xrightarrow{M10}_{c,d} ST287$	$ST272 \xrightarrow{M12}_{c,d} ST271$
$ST272 \xrightarrow{P162}_{c,d} ST274$	$ST273 \xrightarrow{M11}_{c,d} ST182$	$ST273 \xrightarrow{P172}_{c,d} ST288$	$ST274 \xrightarrow{M10}_{c,d} ST288$
$ST274 \xrightarrow{M12}_{c,d} ST289$	$ST275 \xrightarrow{M7}_{c,d} ST283$	$ST275 \xrightarrow{P22}_{c,d} ST290$	$ST276 \xrightarrow{M7}_{c,d} ST286$
$ST276 \xrightarrow{P152}_{c,d} ST259$	$ST277 \xrightarrow{M6}_{c,d} ST290$	$ST277 \xrightarrow{P142}_{c,d} ST240$	$ST279 \xrightarrow{M14}_{c,d} ST278$
$ST279 \xrightarrow{P142}_{c,d} ST141$	$ST280 \xrightarrow{M8}_{c,d} ST291$	$ST280 \xrightarrow{M12}_{c,d} ST292$	$ST280 \xrightarrow{P12}_{c,d} ST293$
$ST282 \xrightarrow{M12}_{c,d} ST281$	$ST282 \xrightarrow{P20}_{c,d} ST294$	$ST283 \xrightarrow{M10}_{c,d} ST295$	$ST283 \xrightarrow{M12}_{c,d} ST296$
$ST283 \xrightarrow{P22}_{c,d} ST297$	$ST284 \xrightarrow{M11}_{c,d} ST229$	$ST284 \xrightarrow{P22}_{c,d} ST298$	$ST286 \xrightarrow{M10}_{c,d} ST298$
$ST286 \xrightarrow{M12}_{c,d} ST285$	$ST286 \xrightarrow{P152}_{c,d} ST274$	$ST287 \xrightarrow{M11}_{c,d} ST206$	$ST287 \xrightarrow{P162}_{c,d} ST288$
$ST288 \xrightarrow{M11}_{c,d} ST155$	$ST290 \xrightarrow{M7}_{c,d} ST297$	$ST290 \xrightarrow{P142}_{c,d} ST259$	$ST291 \xrightarrow{M9}_{c,d} ST299$
$ST291 \xrightarrow{P12}_{c,d} ST300$	$ST293 \xrightarrow{M8}_{c,d} ST300$	$ST293 \xrightarrow{M12}_{c,d} ST292$	$ST294 \xrightarrow{M12}_{c,d} ST281$
$ST294 \xrightarrow{P21}_{c,d} ST301$	$ST295 \xrightarrow{M11}_{c,d} ST249$	$ST295 \xrightarrow{P22}_{c,d} ST302$	$ST297 \xrightarrow{M10}_{c,d} ST302$
$ST297 \xrightarrow{M12}_{c,d} ST296$	$ST297 \xrightarrow{P142}_{c,d} ST274$	$ST298 \xrightarrow{M11}_{c,d} ST229$	$ST298 \xrightarrow{P152}_{c,d} ST288$
$ST300 \xrightarrow{M9}_{c,d} ST299$	$ST301 \xrightarrow{M10}_{c,d} ST303$	$ST301 \xrightarrow{M12}_{c,d} ST304$	$ST301 \xrightarrow{P22}_{c,d} ST305$
$ST302 \xrightarrow{M11}_{c,d} ST249$	$ST302 \xrightarrow{P142}_{c,d} ST288$	$ST303 \xrightarrow{M11}_{c,d} ST281$	$ST303 \xrightarrow{P22}_{c,d} ST306$

Figure 62: Part IV of state transitions for the Inres test case example

$ST305 \xrightarrow{M10}_{c,d} ST306$	$ST305 \xrightarrow{M12}_{c,d} ST304$	$ST305 \xrightarrow{P132}_{c,d} ST307$	$ST306 \xrightarrow{M11}_{c,d} ST281$
$ST306 \xrightarrow{P132}_{c,d} ST308$	$ST307 \xrightarrow{M10}_{c,d} ST308$	$ST307 \xrightarrow{M12}_{c,d} ST309$	$ST308 \xrightarrow{M11}_{c,d} ST247$

Figure 63: Part V of state transitions for the Inres test case example

References

[1] H. J. Genrich. Predicate/Transition Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and their Properties, Advances in Petri Nets, Part I*, volume 254 of *Lecture Notes in Computer Science*. Springer-Verlag, 1986.

[2] J. Grabowski. *Test Case Generation and Test Case Specification with Message Sequence Charts*. PhD thesis, University of Berne, Institute for Informatics and Applied Mathematics, February 1994.

[3] J. Grabowski, D. Hogrefe, and R. Nahm. Test Case Generation with Test Purpose Specification by MSCs. In O. Faergemand and A. Sarma, editors, *SDL'93 - Using Objects*. North-Holland, October 1993.

[4] J. Grabowski, R. Nahm, A. Spichiger, and D. Hogrefe. Die SAMSTAG Methode und ihre Rolle im Konformitätstesten. *Praxis der Informationsverarbeitung und Kommunikation*, 4/94:214 – 224, December 1994.

[5] J. Grabowski, R. Scheurer, D. Toggweiler, and D. Hogrefe. Dealing with the Complexity of State Space Exploration Algorithms. In *Proceedings of the 6th. GI/ITG technical meeting on 'Formal Description Techniques for Distributed Systems', University of Erlangen*, June 1996.

[6] D. Hogrefe. *Estelle, LOTOS und SDL - Standard Spezifikationssprachen für verteilte Systeme*. Springer Verlag, 1989.

[7] D. Hogrefe. OSI Formal Specification Case Study: The Inres Protocol and Service (revised). Technical Report IAM-91-012, Universität Bern, Institut für Informatik, May 1991, Update May 1992.

[8] J. E. Hopcroft and J. D. Ullmann. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, 1979.

[9] ISO, E. Brinksma (ed.). Information Processing Systems - Open Systems Interconnection - LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observable Behaviour. International Standard 8807, ISO, Geneva, 1988.

[10] ISO/IEC JTC 1/SC 21 N. Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework. International Multipart Standard 9646, ISO/IEC, 1992.

[11] ISO/IEC JTC 1/SC21. Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Proposed Draft Amendment 1 to ISO/IEC 9646 Part 3: TTCN Extensions. Proposed Draft Amendment 1 9646-3, ISO/IEC, May 1991.

[12] ISO/IEC JTC 1/SC21. Amendment to ISO/IEC 9646-3: TTCN - Further Extensions. Working Draft, ISO/IEC, May 1992.

[13] ISO/IEC JTC 1/SC21. Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework - Part 3: The Tree and Tabular Combined Notation. International Standard 9646-3, ISO/IEC, 1992.

[14] ITU Telecommunication Standards Sector SG 10. ITU-T Recommendation Z.100: Specification and Description Language (SDL) (formerly CCITT Recommendation Z.100). ITU, Geneva, June 1992.

[15] K. Jensen. Coloured Petri Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and their Properties, Advances in Petri Nets, Part I*, volume 254 of *Lecture Notes in Computer Science*. Springer-Verlag, 1986.

[16] J. Kroon and A. Wiles. A Tutorial on TTCN. In *Proceedings of the 11th International IFIP WG 6.1 Symposium on Protocol, Specification, Testing and Verification*, 1991.

[17] J. v. Leeuwen, editor. *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*. Elsevier Science Publishers B.V., 1990.

[18] R. Nahm. *Conformance Testing Based on Formal Description Techniques and Message Sequence Charts*. PhD thesis, University of Berne, Institute for Informatics and Applied Mathematics, February 1994.

[19] G. Plotkin. A Structural Approach to Operational Semantics. Technical report, Aarhus University, Computer Science Department, 1981.

[20] Telelogic AB, Box 4128, S-203 12 Malmö, Sweden. *ITEX 3.0 User's Guide*, 1994.

[21] D. Toggweiler, J. Grabowski, and D. Hogrefe. Partial Order Simulation of SDL Specifications. In O. Bræk and A. Sarma, editors, *SDL'95 with MSC in CASE*. North-Holland, September 1995.

[22] M. Törö and K. Tarnay. Principles for Validation of Abstract Test Suites specified in Concurrent TTCN. In *IFIP WG 6.1 Fifteenth International Symposium on 'Protocol Specification, Testing and Verification' (PSTV'95)*, June 1995.

[23] T. Walter, J. Ellsberger, F. Kristoffersen, and P. v. d. Merkhof. A Common Semantics Representation for SDL and TTCN. In *PSTV XII*. North-Holland, 1992.

[24] T. Walter, J. Ellsberger, F. Kristoffersen, and P. v. d. Merkhof. Methods for Testing and Specification (MTS) – Semantical Relationship between SDL and TTCN – A Common Semantics Representation. Technical Report ETR 071, European Telecommunications Standards Institute, 1993.

[25] T. Walter and B. Plattner. An Operational Semantics for Concurrent TTCN. In *PTS V*. North-Holland, 1992.