T U M

I N S T I T U T F Ü R I N F O R M A T I K

Architecture: Methodology of Decomposition

Maria Spichkova

TUM-I1018
Oktober 10

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-10-I1018-100/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

This paper presents a methodology of formal specification decomposition. We show which development steps are necessary on this phases and how the system architecture can be decomposed schematically.

Contents

1 Introduction

This paper[1] presents a part of specification and verification process developed within the Verisoft-XT project [14]. The purpose of this project is to integrate verification techniques in real industrial development processes – from specification and analysis of requirements to a verified implementation.

The main focus here is on embedded systems. Embedded systems is not only the most important field for current computer-based applications, but is also one of the most challenging fields of software engineering: such a system must meet real-time requirements, is safety critical and distributed over multiple processors.

The complexity of such systems increases from day to day. Therefore, building correct software becomes more and more complicated. Moreover, the developing of an appropriate and manageable software architecture becomes also more and more important. In this paper we discuss the methodology and earlier phases of the process applied to build verified application software.

The starting point of this approach is a requirement specification developed according to the ideas presented in [5]. This kind of specification is semiformal one an cab be inexact as well as contain some underspecifications or, ever worse, contradictions. On base of these semiformal requirements the corresponding message sequence charts (MSCs) can specified – this representation deals for the visualization of the semiformal specification to get ore readability and to find out more inconsistencies or uderspecified parts already on the semiformal level.

After that we can translate the semiformal specification to a formal ones in Focus [2] that is a framework for formal specifications and development of interactive systems: we split the semiformal requirements into two main parts – the general ones, which correspond to the general system requirements (black box view), and more concrete ones, which correspond to the system architecture (glass box view). If some missing requirements are found, they need to be added; if some inconsistencies are found, they must be corrected corrected. Of course, on the specification phase not all of the underspecifications and inconsistencies can be easily found (this is a task for the verification phase), but a number of them can be resolved even before the formal verification.

In this paper we focus on the formal specification phase: on the developing of a logical system architecture and on the corresponding system decomposition. There is a large number of approaches in this area (see, e.g., [6, 8, 16, 4]). The main difference and the main contribution of our decomposition methodology is that it was developed for such a system architecture, where we already know (or, more precisely, have already specified them in a formal way) systems or components properties and need to decompose this whole properties collection to a number of subcomponents to get readable and manageable specifications. Thus, the presented methodology allows us to decompose component architecture decomposition exactly on this point where we see that the component

[1] This work was fully funded by the German Federal Ministry of Education, Science, Research and Technology (BMBF) in the framework of the Verisoft XT project [14]. The responsibility for this article lies with the author.

specification becomes too large and too complex. In many cases the real complexity of a component (and, consequently, of its formal specification) is realized only during the specification process, when we comes from semiformal (or, even harder, from informal) general description to a formal one – only by collecting and combining all the component properties together for the first time we also get the feeling of the component complexity for the first time. Moreover, during this step a number of component properties can added – in most cases some refinement is necessary.

In addition, our methodology helps to perform the next modeling step – translation to the case tool representation and deployment.

The main ideas, presented in the paper, are language independent, but for the better readability and for better understanding of this ideas we shoe them ob the base of formal specifications presented in the Focus specification framework [2].

We can also see this methodology as an extension of the approach "Focus on Isabelle" [10] – it is integrated into a seamless development process, which covers both specification and verification, starts from informal specification and finishes by the corresponding verified C code (see Verisoft-XT project, [13]). Given a system, represented in Focus, one can verify its properties by translating the specification to a Higher-Order Logic and subsequently using the theorem prover Isabelle/HOL or the point of disagreement can be found. For a detailed description of Isabelle/HOL see [7] and [15].

The translation can be done according to the approach "Focus on Isabelle" [10]. Moreover, using this approach one can validate the refinement relation between two given systems, as well as make automatic correctness proofs of syntactic interfaces for specified system components. Having a Focus specification, we can schematically translate it to a specification in Hight-Order Logic and verify properties of the specified system.

As the next step we can schematically translate the Focus specification to an AutoFOCUS model [9, 3, 1]. AutoFOCUS 3 is a tool for modeling and analyzing the structure and behavior of distributed, reactive, and timed computer-based systems. Having an AutoFOCUS 3 model we can simulate it, prove its properties using model checking and also using its translation to Isabelle/HOL, as well as we gan generate C code from it.

We present here only this part of the case study [11] that is needed to present the advantages of the formal decomposition methodology. Please note that this part of the case study is presented in anonymized form and has in the most parts only a weak correlation to the case study presented in [11].

The general architecture of the specified system does not be described in this paper – we focus only on the main system logic (anonymized and changed vs. the original case study) to show how the decomposition methodology works and how it can help us to find out underspecifications and inconsistencies in formal specifications.

Outline:
The next section gives a short introduction to FOCUS: main concepts and specification kinds, and Section 3 presents the main theoretical part of the paper – system architecture decomposition methodology.

After that the an anonymized part of the case study from the Verisoft-XT project [13] is discussed:

- the used data types, constants, auxiliary functions and predicates are described in Sections 4 and 5;
- the system components as well as their decomposition according the presented methodology are discussed in Section 6.
- the system requirements are specified in Section 7 to complete the FOCUS specification phase for the main system logic component.

Section 8 summarizes the paper.

2 FOCUS: Main Aspects

A distributed system in FOCUS is represented by its *components*[2]. Components that are connected by communication lines called *channels*, can interact or work independently of each other.

The channels in FOCUS are *asynchronous communication links* without delays. They are directed, reliable, and order preserving. Via these channels components exchange information in terms of *messages* of specified types. The formal meaning of a FOCUS specification is a relation between the *communication histories* for the external input and output channels.

The specifications can be structured into a number of formulas each characterizing a different kind of property, the most prominent classes of them are *safety* and *liveness properties*. FOCUS supports a variety of *specification styles* which describe system components by logical formulas or by diagrams and tables representing logical formulas.

Specification of a real-time system in the untimed frame may be in some cases shorter or more elegant from mathematical point of view, but case studies have shown, that to understand such specifications and to argue about their properties is in many cases much more difficult in comparison to the corresponding specifications in the timed frame that use causality property explicitly. Moreover, abstraction from timing aspects can easily lead to specification mistakes because of difficulties of correct abstraction.

Thus, we restrict in the methodology "FOCUS on Isabelle" [10] the whole FOCUS specification domain for representation embedded real-time systems to only timed and time-synchronous systems. This not only simplifies the translation into Isabelle/HOL, but also allows us to concentrate on the timing properties

[2] A component in FOCUS means a "logical component" and not a physical one.

to have not only more clear and readable specifications, but also simpler proofs about them.

Considering causality (weak or strong) it is simpler and also more readable to argue not about single messages in a timed stream, but about a sequence of messages that are present in this stream at some time interval. This sequence can be in general empty, contain a single message or a number of messages. In the case of time-synchronous stream this sequence must always contain exactly one message.

For easier argumentation about the behavior of a component at some time interval we introduced a special kind of Focus tables and state transition diagrams, which help us to specify a component in the time interval based way (see [10]). This approach to represent a timed component will be used also for the presented case study.

As mentioned in the methodology "Focus on Isabelle", the concrete meaning of a time interval is not defined in the Focus specification, but it must be specified additionally as a remark to the specification. This interpretation flexibility allows to specify systems also for the case where the "time intervals" does not have the same (constant) duration and are understood as a formal technique for a causality representation.

2.1 Concept of Streams

The central concept in Focus are *streams*, that represent communication histories of *directed channels*. Streams in Focus are functions mapping the indexes in their domains to their messages. For any set of messages M, M^ω denotes the set of all streams, M^∞ and M^* denote the sets of all infinite and all finite streams respectively. $M^{\underline{\omega}}$ denotes the set of all timed streams, $M^{\underline{\infty}}$ and $M^{\underline{*}}$ denote the sets of all infinite and all finite timed streams respectively.

A *timed stream* is represented by a sequence of messages and *time ticks*, the messages are also listed in their order of transmission. The ticks model a discrete notion of time.

The timed domain is the most important one for representation of distributed systems with real-time requirements. Specifications of embedded systems must be *timed*, because by representing a real-time system as an untimed specification a number of properties of the system are loosed (e.g. the causality property) that are not only very important for the system, but also help us to make proofs easier. Another ways of streams formalizations as well as the related work for the approach "Focus on Isabelle" are discussed in [10].

To simplify the specification of the real-time systems we use an additional Focus operator $\mathsf{ti}(s, n)$ that yields the list of messages that are in the timed stream s between the ticks $n-1$ and n (at the nth time unit).

The predicate ts holds for a timed stream s, iff s is time-synchronous in the sense that exactly one message is transmitted in each time interval.

The Focus operator $\mathsf{msg}_n(s)$, which holds for a timed stream s, if this stream contains at every time unit at most n messages.

2.2 Specifications

FOCUS specifications can be *elementary* or *composite*. Any elementary FOCUS specification has the following syntax:

Name (Parameter_Declarations) Frame_Labels

in *Input_Declarations*

out *Output_Declarations*

Body

where

- *Name* is the name of the specification;
- *Frame_Labels* lists a number of frame labels, e.g. *untimed*, *timed* or *time-synchronous*, that correspond to the stream types in the specification (see Sect. 2.1);
- *Parameter_Declarations* lists a number of parameters (optional);
- *Input_Declarations* and *Output_Declarations* list the declarations of input and output channels respectively;
- *Body* characterizes the relation between the input and output streams, and can be a number of formulas, or a table, or diagram or a combination of them.

For any elementary timed parameterized specification S we define its semantics, written $[\![S]\!]$, to be the formula:

$$i_S \in I_S^{\underline{\infty}} \;\wedge\; p_S \in P_S \;\wedge\; o_S \in O_S^{\underline{\infty}} \;\wedge\; B_S \tag{1}$$

where i_S and o_S denote lists of input and output channel identifiers, I_S and O_S denote their corresponding types, p_S denotes the list of parameters and P_S denotes their types, B_S is a formula in predicate logic that describes the body of the specification S.

Focus operators used in the paper:
An empty stream is represented in FOCUS by $\langle\rangle$.
$\langle x\rangle$ denotes the one element stream consisting of the element x.
$\#s$ denotes the length of the stream s.
ith time interval of the stream s is represented by $\mathsf{ti}(s, i)$.
$\mathsf{msg}_n(s)$ denotes a stream s that can have at most n messages at each time interval.
s^i_{ft} denotes the first element of the ith time interval of the stream s (partial function).

See [2] and [10] for more background on FOCUS and its extensions.

3 Architecture: Methodology of Decomposition

Let assume a formal (FOCUS) specification of some component, which covers a large number of its properties, s.t. most of which have strong correlation, and let this component describes among others the system states and transitions between them, s.t. the resulting representation must correspond to a state transition diagram.

If we specify this component as a single, non-composite, specification we get a set of formulas that is not really understandable. Trying to built a state transition diagram for the whole component, we will get a large automat with spaghetti-transitions between them – this representation will be useless and not manageable. Moreover, the later representation in some case tool, e.g. AutoFocus, will be not fit the model checker restrictions. Therefore, we have a challenge to decompose it in a number of subcomponents to get some (more) readable specification.

A simple, intuitive, way to decompose a component is not suitable here. In this case we need to have some rules to decompose the component according to the kinds of its logical properties.

We start the decomposition to observe the properties that correspond to the different kinds of automats: Mealy and Moore.

3.1 Mealy vs. Moore?

By definition, any state machine can be either a Mealy automat, where the output depends both on the current input and state, or a Moore automat, where the output depends only from the current state.

Generally, having a specification represented by a number of formulas, we can divide these formulas into two parts: formulas, which correspond to the definition of a Mealy automat, and formulas, which correspond to the definition of the Moore automat. Thus, having a component *CComp* describing large state machine, we can decompose it into two components as follows:

- component C, describing reactions on the component (system) inputs and describing all the state transitions – corresponds to a Mealy automat,
- component *CInf*, describing outputs, which depend only on the system state – corresponds to a Moore automat.

This decomposition also belong to the parameters of the component *CComp*.

Please note, that the sets of output streams of C and *CInf* must be disjoint, and their union without information about system state results the set of output streams of the component *CComp*. Under *information about system state* we understand here the extra output stream *stateInf* that must be added to the component C to send the current state value to the component *CInf*.

In the notation from [2]:

$$o_C \cap o_{CInf} = \emptyset$$
$$o_{CComp} = (o_C \setminus i_{CInf}) \cup o_{CInf}$$

In some cases we have to split formulas to separate the description of reactions on the system inputs as well as all the state transitions from the descriptions of messages about the system.

For example, if we add to the specification *CInf* some formula like

$$stateInf_{\mathsf{ft}}^{t} = SomeState \;\rightarrow\; \mathsf{ti}(x, t+1) = SomeValueOfTimeInterval$$

where x is an output stream of the component *CComp*, then we must move all other definitions of the stream x to this specification also, and simplify in the specification C all the formulas of kind

$$\begin{array}{l} Some\_Term\_1 \;\rightarrow \\ CState' = SomeState \;\wedge\; \mathsf{ti}(x, t+1) = SomeValueOfTimeInterval \\ \wedge\; Some\_Term\_2 \end{array}$$

to the formulas of kind

$$Some\_Term\_1 \;\rightarrow\; CState' = SomeState \;\wedge\; Some\_Term\_2$$

3.2 Local Variables

In this section we discuss the decomposition schema we proposed to use for all local variables $x_1 : M_1, \dots, x_n : M_n$ that are moved via decomposition from a component C to some extra component *CLoc*.

This schema describes not only the way to write the specification *CLoc*, but also the changes we need to do in the specification C. After applying this schema we get two specifications, C' and *CLoc*, which composition results the specification C:

$$C = C' \otimes CLoc$$

1. The set of input channels of the component *CLoc* is a subset of the corresponding set of the component C. In the notation from [2]:

   $$i_{CLoc} \subseteq i_C$$

2. Add all the assumptions about the input streams according to the specification C.

3. The set of output channels of the component *CLoc* corresponds to the local variables to move and have the same data type as these variables. Let call these channels $m_1 : M_1, \dots, m_n : M_n$, s.t. the channel m_i corresponds to the variable x_i.

   In some cases we can use for these streams the same names as for the variables.

4. Move all corresponding formulas from the specification C to the specification *CLoc*.

5. Add to the component C the channels $m_1 : M_1, \dots, m_n : M_n$ as extra input channels. Add to the component C assumptions that these streams are time-synchronous.

6. Delete from the interface of C all the input streams that are used only in the formulas moved to the specification *CLoc*. Delete all the assumptions about these streams.

7. Define in *CLoc* for all $x \in \{x_1, \dots, x_n\}$ the initial value of the corresponding stream m as follows according to the initial value of the local variable:

   $$x = \mathit{SomeValue}_1$$

   will be translated to the formula

   $$\mathsf{ti}(m, 0) = \langle \mathit{SomeValue}_1 \rangle$$

8. Replace all the entrances of the local variables x from $x_1 : M_1, \dots, x_n : M_n$ at the current time interval t (denoted by x) by the values of tth time interval of the corresponding stream m^t_{ft}.

9. Replace all the entrances of the local variable x at the time interval $t+1$ or $t+n$ (denoted by x' and $x^{(n)}$ respectively) by $\mathsf{ti}(m, t+1)$ and $\mathsf{ti}(m, t+n)$ respectively.

   Convert the related part of formula to the time interval syntax, e.g.

   $$x = \mathit{SomeValue}_2$$

   must be converted to

   $$\mathsf{ti}(m, t) = \langle \mathit{SomeValue}_2 \rangle$$

   Another solution for this point is to replace all the entrances of the local variable x at the time interval $t+1$ or $t+n$ by m^{t+1}_{ft} and m^{t+1}_{ft} respectively. These kind of specification will also define the whole corresponding time interval: the stream m that corresponds to the local variable x is by definition a time-synchronous one, thus, it has exactly one message at each time unit, therefore, we can define this message as the first (and the only one) message of the time interval.

10. Delete from the specification C definitions of the initial values of the local variables $x_1 : M_1, \dots, x_n : M_n$.

11. Add to the specification *CLoc* all the needed parameters of the component C. Remove from the component C the parameters that are not in use any more.

Please note, that the component *CLoc* is strong causal, where the component C' preserves the causality property of the component C.

3.3 Outputs That Depends from Inputs

In this section we discuss the decomposition schema we proposed to use for all output streams $o_1 : M_1, \dots, o_n : M_n$ and corresponding formulas describing them (depending only on the component state, local variables and some inputs) that are moved via decomposition from a component C' to some extra component $COut$.

We suggest to extract specification parts according this schema must after the moving of local variables via decomposition.

1. If the formulas to extract to the component $COut$ contain also some local variables $l_1 : M_1, \dots, l_k : M_k$ of the component C', then the corresponding output channels $ml_1 : M_1, \dots, ml_k : M_k$ must be added to the component C', s.t. the channel ml_i corresponds to the variable l_i. As result we get a component C'', s.t.

   $$i_{C''} = i_{C'} \cup \{ml_1, \dots, ml_k\}.$$

   The semantics of the specification C' is extended to the semantics of C'' by the formulas describing the streams $ml_1, \dots, ml_k$:

   $$\forall\, t \in \mathbb{N} : \ \mathsf{ti}(ml_1, t) = \langle l_1 \rangle$$

   ...

   $$\forall\, t \in \mathbb{N} : \ \mathsf{ti}(ml_k, t) = \langle l_k \rangle$$

2. The set of input channels of the component $COut$ is a subset of union of the input and output channels sets of the component C''. In the notation from [2]:

   $$i_{COut} \subseteq (i_{C''} \cup o_{C''})$$

   where the set of output channels of the component $COut$ is only the set of output channels $o_1 : M_1, \dots, o_n : M_n$ moved from C' to$COut$. We remove these outputs from the definition of C''.

3. Add to the specification $COut$ all the assumptions about its input streams according to the specification C''.

4. If values of some input streams are used only in the formulas to extract to the component $COut$, then we can remove these inputs from interface of the component C''. As result we get a component C'''.

5. Delete from the specification C''' all the assumptions about the input streams that are removed according the previous step.

6. Add the assumption about all the extra channels channels $ml_1 : M_1, \dots, ml_k : M_k$: the corresponding streams must be time-synchronous.

7. Move all corresponding formulas from the specification C''' to the specification $COut$.

8. Replace all the entrances of all local variables l from $l_1 : M_1, \dots, l_k : M_k$ at the current time interval t (denoted by l) by ml^t_{ft}.

9. Replace all the entrances of the local variable l at the time interval $t+1$ or $t+n$ (denoted by l' and $l^{(n)}$ respectively) by $\mathsf{ti}(ml, t+1)$ and $\mathsf{ti}(ml, t+n)$ respectively.

   Convert the related part of formula to the time interval syntax, e.g.

   $l = \mathit{SomeValue}_2$

   must be converted to $\mathsf{ti}(l, t) = \langle \mathit{SomeValue}_2 \rangle$.

10. Add to the specification *Cout* all the needed parameters of the component C'. Remove from the component C' the parameters that are not in use any more.

Please note, that the component *COut* is strong causal only if the component C was strong causal.

4 Case Study: Data Types and Constants

In this section we define data types, which are needed to specify the case study system and its components. The main part of these data types must be inhered from the semiformal specification, but some of them can represent refined versions of the data types from the semiformal specification. Please note that we present here only these data types and constants, which are used in the specification of the main system logic component, and omit all other data type and constant definitions.[3]

We deal here with a system that has 8 logical states of type *StateType* and 8 main control signals of type *SignalAType* as well as with 3 control signals the type *SValueType*. The *Event* type represents signals showing that some event was happen.

$$\mathsf{type} \quad \mathit{StateType} = \{S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$$

$$\mathsf{type} \quad \mathit{SignalAType} = \{\mathit{SignalA}_1, \mathit{SignalA}_2, \mathit{SignalA}_3, \mathit{SignalA}_4, \mathit{SignalA}_5, \mathit{SignalA}_6, \mathit{SignalA}_7, \mathit{SignalA}_8\}$$

$$\mathsf{type} \quad \mathit{SValueType} = \{V_1, V_2, V_3\}$$

$$\mathsf{type} \quad \mathit{Event} = \{\mathit{event}\}$$

[3] Please also note that we present here an anonymized version of the case study specification [11]. In the full case study – among other differences – some other, more meaningful, names for function, constants, data types etc. are used, but here the concrete meaning of the names is unimportant to understand the methodology.

The system has the following parameters:

- X_Appl, $Xcounter$, $CValue$: $\mathbb{N}$ – correction values;
- $MinCurrentValue$, $MaxCurrentValue$: $\mathbb{N}$ – bounds for the current value of the main sensor;
- $MinTargetValue$, $MaxTargetValue$: $\mathbb{N}$ – bounds for the target value of the main sensor;
- $sSignal1\_target1, sSignal1\_target2 \in \mathbb{N}$ – abstraction of the target values that are set by system transition from the state S_3 or from the state S_4 to the states S_5 and S_6 respectively.
- $tcontrol \in \mathbb{N}$ - target control value.

5 Case Study: Auxiliary Functions and Predicates

In this section we present the Focus specifications of all auxiliary functions and predicates used to represent the Case Study System in a formal way.

5.1 Function ModSubtraction

The function *ModSubtraction* returns the difference between two natural numbers as a positive natural number.

ModSubtraction

$\mathbb{N} \times \mathbb{N} \to \mathbb{N}$

$ModSubtraction(x, y) =$
  if $x < y$
  then $y - x$
  else $x - y$
  fi

5.2 Predicate Signal1Precondition

The predicate *Signal1Precondition* analyses a given time interval of the infinite timed stream of type *SignalType* and returns true only if this time interval is not empty and contains exactly one of the following messages: $SignalA_2$, $SignalA_3$, $SignalA_4$, $SignalA_5$, $SignalA_6$, $SignalA_7$, $SignalA_8$.

Signal1Precondition

$$SignalType^* \to \mathbb{B}\mathrm{ool}$$

$$Signal1Precondition(\langle\rangle) = \mathsf{false}$$
$$Signal1Precondition(\langle x \rangle \frown s) =$$
$$s = \langle\rangle$$
$$\wedge$$
$$((x = SignalA_2) \vee (x = SignalA_3) \vee (x = SignalA_4) \vee$$
$$(x = SignalA_5) \vee (x = SignalA_6) \vee$$
$$(x = SignalA_7) \vee (x = SignalA_8))$$

5.3 Function SignalAccepted

The following function describes relations between a *current_value*, a *targetValue*, applicable values x and *xcounter*, as well as current numbers of counters, $counter_1$ and $counter_2$.
Please note that *MaxTargetValue*, *MinTargetValue*, *Xcounter* and *CValue* are global system parameters, thus, there is no need to include them as parameters of the function *SignalAccepted*.
The first parameter of the function has a boolean type and is used to choose a mode in which the function is applied.

SignalAccepted

$$current\_value, targetValue, counter_1, counter_2 \in Nat$$

$$SignalAccepted(\mathsf{true}, current\_value, targetValue, counter_1, counter_2) =$$
$$counter_1 < Xcounter \wedge counter_2 = 0 \wedge$$
$$ModSubtraction(current\_value, targetValue) \leq x \wedge$$
$$targetValue + CValue \leq MaxTargetValue$$

$$SignalAccepted(\mathsf{false}, current\_value, targetValue, counter_1, counter_2) =$$
$$counter_2 < Xcounter \wedge counter_1 = 0 \wedge$$
$$ModSubtraction(current\_value, targetValue) \leq Xcounter \wedge$$
$$(MinTargetValue + CValue \geq targetValue)$$

5.4 Predicates SystemStateSubset and CrCtStateActive

The predicate *SystemStateSubset* holds if its input value is one of the following state values: S_3, S_4, S_5, or S_6, where the predicate *SystemStateSubset2* holds if its input value is S_4, S_5, or S_6.

SystemStateSubset

$st \in StateType$

$$SystemStateSubset(st) = (st = S_3) \lor (st = S_4) \lor (st = S_5) \lor (st = S_6)$$

SystemStateSubset2

$st \in StateType$

$$SystemStateSubset(st) = (st = S_4) \lor (st = S_5) \lor (st = S_6)$$

It is easy to see the following implication relations between these predicates:

$$SystemStateSubset2(st) \rightarrow SystemStateSubset(st)$$
$$SystemStateSubset(st) \nrightarrow SystemStateSubset2(st)$$

5.5 Function LimitedValue

The function *LimitedValue* calculates the limitation of the target speed value according to the current value v and to the applicable parameters $minS$, $maxS$, $minT$, and $maxT$ representing the minimal and the maximal current speed as well as the minimal and the maximal target speed respectively.

LimitedValue

$\mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

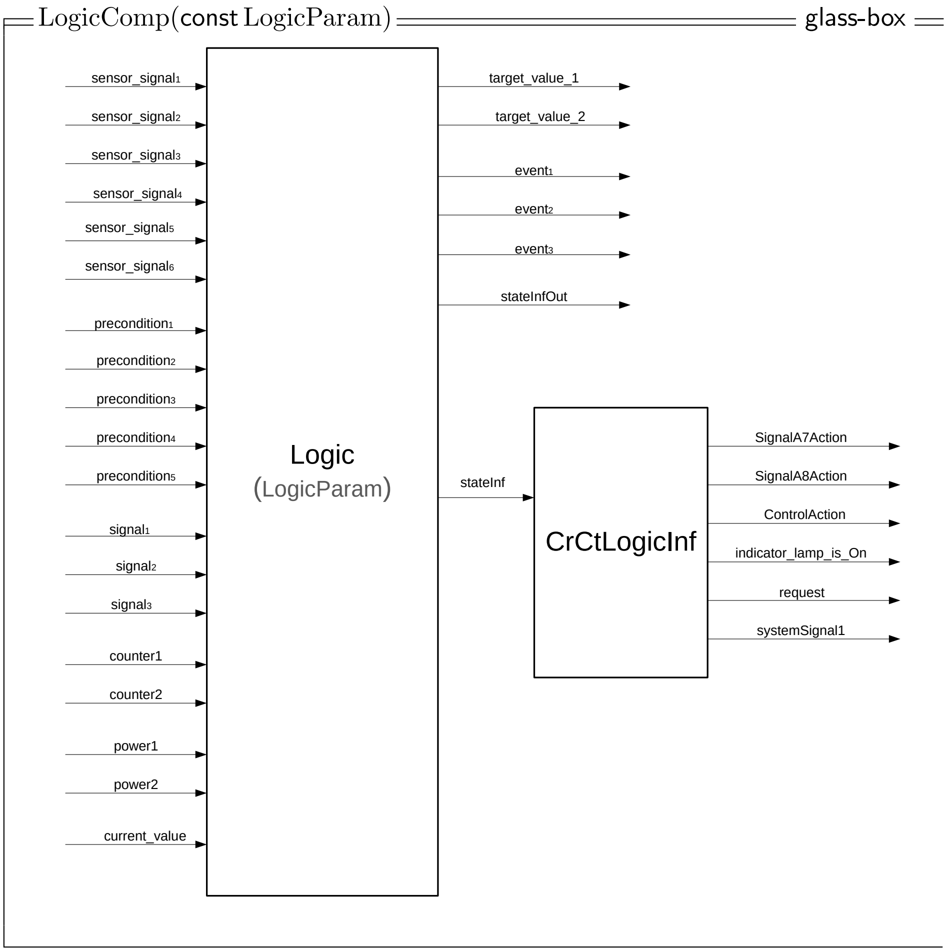
$LimitedValue(v, minS, minT, maxS, maxT) =$
if $v < minTV$
then $minTV$
else (if $v > maxTV$
then $maxTV$
else v
fi)

where $minTV, maxTV$ so that
$minTV = max(minS, minT)$
$maxTV = min(maxS, maxT)$

6 Case Study: Specifications of the Subcomponents

6.1 Logic

The specification of the component *LogicComp* that describes an imaginer[4] logic component, developed by authors of the paper to show the main ideas of the decomposition methodology. The component *LogicComp* fulfills all the properties described in Section 3 – it covers a large number of components properties most of which have a strong correlation, moreover, this component describes the system states and transitions between them. Thus, we need to decompose *LogicComp* in a number of subcomponents to get more readable specification, and we do it according the methodology presented in Section 3.



We do not present here the component *LogicComp* as a collection of Focus formulas, because this kind of specification is very unreadable and do not fit on a page also with the tiny letter size.

[4]Some influence to the development of this component was given by the The Cruise Control case study [12, 11].

6.1.1 Decomposition: Mealy vs. Moore

Decomposing the component *LogicComp* by the rules described in 3.1 we get the following two components, which describe the main logic of the system:

- The component *Logic* describes reactions on the system inputs as well as all the state transitions.
- The component *LogicInf* is used to represent only these requirements, which describe some messages about the system.

Their sets of output streams are disjoint – we had separate the description of reactions on the system inputs as well as all the state transitions from the descriptions of messages about the system.

For example, we add to the specification *LogicInf* the following formula

$$stateInf^{t}_{\mathsf{ft}} = S_5 \;\rightarrow\; \mathsf{ti}(systemSignal1, t) = \langle sSignal1\_target1 \rangle$$

and this implies that we must move all other definitions of the stream $systemSignal1$ to this specification also, and simplify in the specification *Logic* all the formulas of kind

$$\begin{aligned} & Some\_Term\_1 \;\rightarrow \\ & SystemState' = S_5 \;\wedge\; \mathsf{ti}(systemSignal1, t+1) = \langle sSignal1\_target1 \rangle \\ & \wedge\; Some\_Term\_2 \end{aligned}$$

to the formulas of kind

$$Some\_Term\_1 \;\rightarrow\; SystemState' = S_5 \;\wedge\; Some\_Term\_2$$

Please note the time stamp convention within these two specifications: the component *Logic* is strong causal, where the component *LogicInf* is weak causal and works only with one input stream — the *stateInf* output stream of *Logic* (this stream is a local one for the composition of these two components). Thus, the outputs of both components have the same delay wrt. to the inputs of the component *Logic*.

Let us proceed with the specification details of the components *LogicInf* and *Logic*.

6.1.2 LogicInf Component

The Focus specification of the component *LogicInf* is presented below. This component describes messages to signal about system state or actions according to the system state.

LogicInf (const tcontrol, sSignal1_target1, sSignal1_target2) — timed

in $stateInf : StateType$

out $SignalA7Action, SignalA8Action, S_4Action : Event$;
$indicator\_lamp\_is\_On : \mathbb{B}ool$; $request, systemSignal1 : \mathbb{N}$;

asm $\mathsf{ts}(stateInf)$

gar
$\forall\, t \in \mathbb{N}$:

$$SystemStateSubset2(stateInf_{\mathsf{ft}}^{t}) \;\rightarrow\; \mathsf{ti}(indicator\_lamp\_is\_On, t) = \langle \mathsf{true} \rangle$$

$$\neg SystemStateSubset2(stateInf_{\mathsf{ft}}^{t}) \;\rightarrow\; \mathsf{ti}(indicator\_lamp\_is\_On, t) = \langle \mathsf{false} \rangle$$

$$\mathsf{ti}(stateInf, t) = \langle S_5 \rangle \;\rightarrow\; \mathsf{ti}(SignalA7Action, t) = \langle event \rangle$$
$$\mathsf{ti}(stateInf, t) \neq \langle S_5 \rangle \;\rightarrow\; \mathsf{ti}(SignalA7Action, t) = \langle\rangle$$

$$\mathsf{ti}(stateInf, t) = \langle S_6 \rangle \;\rightarrow\; \mathsf{ti}(SignalA8Action, t) = \langle event \rangle$$
$$\mathsf{ti}(stateInf, t) \neq \langle S_6 \rangle \rightarrow\; \mathsf{ti}(SignalA8Action, t) = \langle\rangle$$

$$\mathsf{ti}(stateInf, t) = \langle S_4 \rangle \;\rightarrow\; \mathsf{ti}(S_4Action, t) = \langle event \rangle \;\wedge\; \mathsf{ti}(control\_request, t) = \langle tcontrol \rangle$$

$$\mathsf{ti}(stateInf, t) \neq \langle S_4 \rangle \;\rightarrow\; \mathsf{ti}(S_4Action, t) = \langle\rangle \;\wedge\; \mathsf{ti}(control\_request, t) = \langle\rangle$$

$$stateInf_{\mathsf{ft}}^{t} = S_5 \;\rightarrow \mathsf{ti}(systemSignal1, t) = \langle sSignal1\_target1 \rangle$$

$$stateInf_{\mathsf{ft}}^{t} = S_6 \;\rightarrow \mathsf{ti}(systemSignal1, t) = \langle sSignal1\_target2 \rangle$$

Specifying the system in a formal way one can found some missing assumptions or conclusions not only by verification but also from the specification itself. In our case we add here to the the specification *LogicInf* the following new (wrt. semiformal specification) subformula

$$stateInf_{\mathsf{ft}}^{t} \neq S_5 \;\wedge\; stateInf_{\mathsf{ft}}^{t} \neq S_6 \;\rightarrow\; \mathsf{ti}(systemSignal1, t) = \langle\rangle$$

otherwise the values of the stream $systemSignal1$ will be defined only for two system states. We can also join the 9th formula with the 3rd one, and the 10th with the 5th one. The resulting specification is presented below:

LogicInf (const tcontrol, sSignal1_target1, sSignal1_target2) timed

in $stateInf : StateType$

out $SignalA7Action, SignalA8Action, S_4Action : Event;$
$indicator\_lamp\_is\_On : \mathbb{B}ool;\ control\_request, systemSignal1 : \mathbb{N};$

asm $\mathsf{ts}(stateInf)$

gar
$\forall\, t \in \mathbb{N}:$

$SystemStateSubset2(stateInf^{t}_{\mathsf{ft}}) \;\rightarrow\; \mathsf{ti}(indicator\_lamp\_is\_On, t) = \langle \mathsf{true} \rangle$

$\neg SystemStateSubset2(stateInf^{t}_{\mathsf{ft}}) \;\rightarrow\; \mathsf{ti}(indicator\_lamp\_is\_On, t) = \langle \mathsf{false} \rangle$

$\mathsf{ti}(stateInf, t) = \langle S_5 \rangle$
$\rightarrow\; \mathsf{ti}(SignalA7Action, t) = \langle event \rangle \;\wedge\; \mathsf{ti}(systemSignal1, t) = \langle sSignal1\_target1 \rangle$

$\mathsf{ti}(stateInf, t) \neq \langle S_5 \rangle \;\rightarrow\; \mathsf{ti}(SignalA7Action, t) = \langle\rangle$

$\mathsf{ti}(stateInf, t) = \langle S_6 \rangle$
$\rightarrow\; \mathsf{ti}(SignalA8Action, t) = \langle event \rangle \;\wedge\; \mathsf{ti}(systemSignal1, t) = \langle sSignal1\_target2 \rangle$

$\mathsf{ti}(stateInf, t) \neq \langle S_6 \rangle \;\rightarrow\; \mathsf{ti}(SignalA8Action, t) = \langle\rangle$

$\mathsf{ti}(stateInf, t) = \langle S_4 \rangle$
$\rightarrow\; \mathsf{ti}(S_4Action, t) = \langle event \rangle \;\wedge\; \mathsf{ti}(control\_request, t) = \langle tcontrol \rangle$

$\mathsf{ti}(stateInf, t) \neq \langle S_4 \rangle$
$\rightarrow\; \mathsf{ti}(S_4Action, t) = \langle\rangle \;\wedge\; \mathsf{ti}(control\_request, t) = \langle\rangle$

$stateInf^{t}_{\mathsf{ft}} \neq S_5 \;\wedge\; stateInf^{t}_{\mathsf{ft}} \neq S_6 \;\rightarrow\; \mathsf{ti}(systemSignal1, t) = \langle\rangle$

This component is weak-causal and does not use any local variables. Therefore, we can represent it in AutoFocus as a functional specification.

6.1.3 Logic Component

The Focus specification of the component *Logic* is presented below.

Please note that we extend here the original syntax of the Focus specification language to have within component specifications bodies with a large number of describing formulas an enumeration of these formulas.

To make our specification more readable we can decompose also the component *Logic*. Beholding the specification *Logic* we can divide its formulas into three parts, applying the ideas of formal decomposition – firstly from Section 3.2 and secondly from Section 3.3:

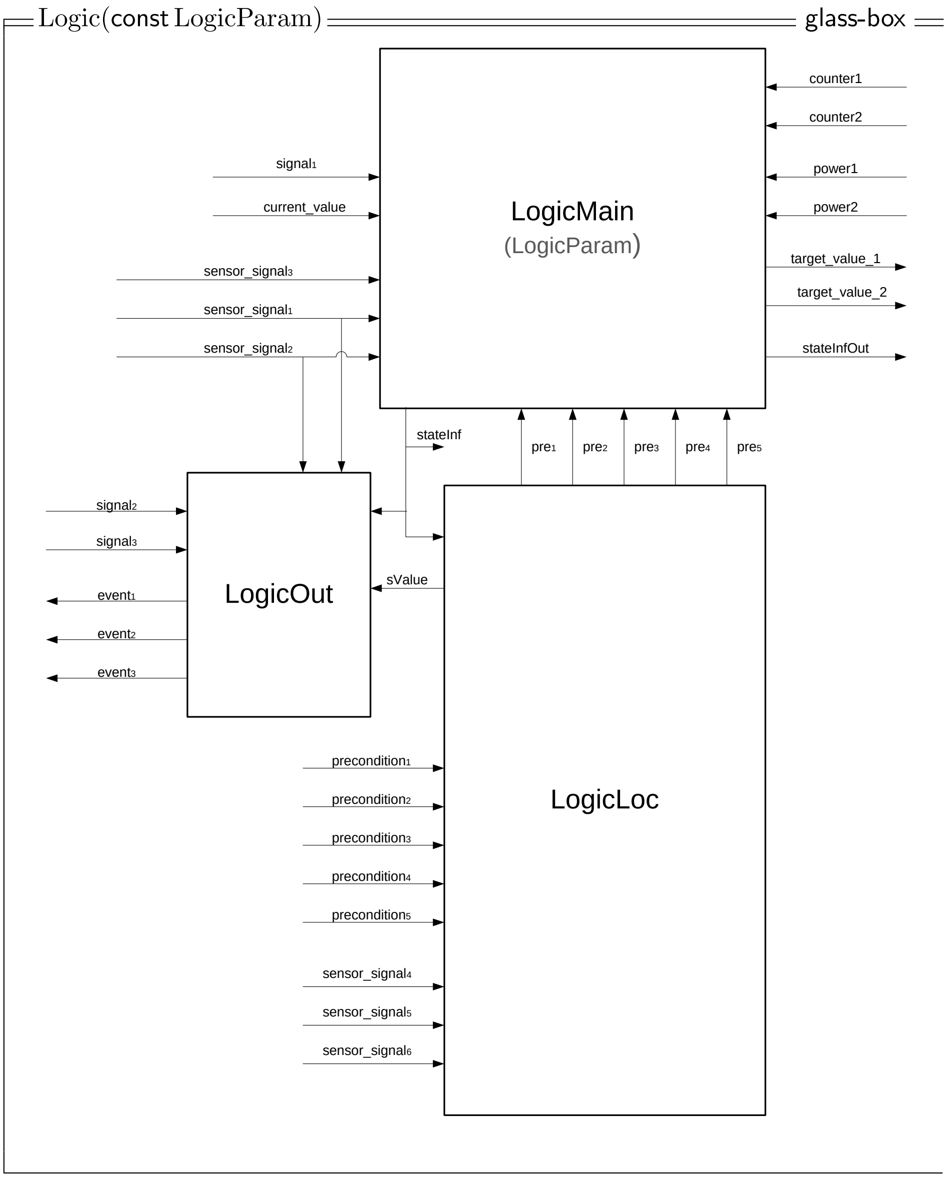
(1) describing inter alia state transitions;

(2) describing no state transitions, but only component outputs – depending on the component state, local variables and some inputs;

(3) describing no state transitions, but only some manipulations on the local variables – depending on the component state and some inputs.

This decomposition means, that the main component *LogicMain* has less local variables than the component *Logic*.

The optimized architecture of the component *Logic* as well as the decomposition steps are presented below.



Logic(const LogicParam) — timed

in
$sensor\_signal1, sensor\_signal2, sensor\_signal3, sensor\_signal4, sensor\_signal5, sensor\_signal6 : \mathbb{B}ool$
$signal_1 : SignalType; \quad current\_value, counter1, counter2 : \mathbb{N};$
$precondition_1, precondition_2, precondition_3, precondition_4, precondition_5, power1, power2, signal_2, signal_3 : Event$

out
$event_1, event_2, event_3 : Event; \; target\_value\_1, target\_value\_2 : \mathbb{N}; \quad stateInf, stateInfOut : StateType$

---

local $SystemState : StateType; \; pre_1, pre_2, pre_3, pre_4, pre_5 : \mathbb{B}ool; \; targetValue : \mathbb{N}; \; sValue : SValueType;$

---

init $SystemState = S_0; \; pre_1 = \mathsf{false}; \; pre_2 = \mathsf{false}; \; pre_3 = \mathsf{false}; \; pre_4 = \mathsf{false}; \; pre_5 = \mathsf{false}; \; targetValue = 0; \; sValue = V_1;$

---

asm
$\mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2) \wedge \mathsf{ts}(sensor\_signal3) \wedge \mathsf{ts}(sensor\_signal4) \wedge \mathsf{ts}(sensor\_signal5) \wedge \mathsf{ts}(sensor\_signal6)$
$\mathsf{msg}_1(signal_1) \wedge \mathsf{ts}(current\_value) \wedge \mathsf{ts}(counter1) \wedge \mathsf{ts}(counter2) \wedge \mathsf{msg}_1(power1) \wedge \mathsf{msg}_1(power2)$
$\mathsf{msg}_1(precondition_1) \wedge \mathsf{msg}_1(precondition_2) \wedge \mathsf{msg}_1(precondition_3) \wedge \mathsf{msg}_1(precondition_4) \wedge \mathsf{msg}_1(precondition_5)$

---

gar

1 $stateInfOut = stateInf \wedge target\_value\_2 = target\_value\_1$

$\forall\, t \in \mathbb{N}$:

2 $\mathsf{ti}(stateInf, t) = \langle SystemState \rangle \wedge \mathsf{ti}(target\_value\_1, t) = \langle targetValue \rangle$

3 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge \neg\, Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_3$
4 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_2$

5 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true} \rangle \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle \rightarrow SystemState' = S_2$

6 $SystemState = S_4 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge \neg sensor\_signal_{12} \wedge SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_5)$

7 $SystemState = S_4 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge \neg sensor\_signal_{12} \wedge SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_6)$

8 $SystemState \neq S_2 \wedge SystemState' = S_2 \wedge sValue = V_1 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_3, t+1) = \langle event \rangle$
9 $SystemState \neq S_2 \wedge SystemState' = S_2 \wedge sValue = V_2 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_1, t+1) = \langle event \rangle$
10 $SystemState \neq S_2 \wedge SystemState' = S_2 \wedge sValue = V_3 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_2, t+1) = \langle event \rangle$
11 $SystemState = S_2 \wedge \mathsf{ti}(signal_2, t) \neq \langle\rangle \wedge \mathsf{ti}(signal_3, t) \neq \langle\rangle \rightarrow \mathsf{ti}(event_3, t+1) = \langle event \rangle$

12 $SystemState = S_0 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \rightarrow targetValue' = 0 \wedge CrCtSate' = S_1$

13 $\mathsf{ti}(power1, t) = \langle\rangle \rightarrow CrCtSate' = S_0$
14 $\mathsf{ti}(precondition_1, t) \neq \langle\rangle \rightarrow pre_1' = \mathsf{true}$
15 $\mathsf{ti}(precondition_2, t) \neq \langle\rangle \rightarrow pre_2' = \mathsf{true}$
16 $\mathsf{ti}(precondition_3, t) \neq \langle\rangle \rightarrow pre_3' = \mathsf{true}$
17 $\mathsf{ti}(precondition_4, t) \neq \langle\rangle \rightarrow pre_4' = \mathsf{true}$
18 $\mathsf{ti}(precondition_5, t) \neq \langle\rangle \rightarrow pre_5' = \mathsf{true}$

19 $SystemState = S_1 \wedge pre_1' \wedge pre_2' \wedge pre_3' \wedge pre_4' \wedge pre_5' \rightarrow SystemState' = S_2$
20 $SystemState = S_1 \wedge (\neg\, pre_1' \vee \neg\, pre_2' \vee \neg\, pre_3' \vee \neg\, pre_4' \vee \neg\, pre_5') \rightarrow SystemState' = S_1$

21 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

22 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

23 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge \mathsf{ti}(counter2, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

24 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge \mathsf{ti}(counter1, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

25 $(SystemStateSubset(SystemState) \vee SystemState = S_2) \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{true} \rangle \rightarrow SystemState' = S_7$

26 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$
27 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue > 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4 \rangle \rightarrow SystemState' = S_4 \wedge targetValue' = targetValue$
28 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue = 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4 \rangle \rightarrow SystemState' = S_3$
29 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \rightarrow SystemState' = S_5 \wedge targetValue' = limTargetValue$
30 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \rightarrow SystemState' = S_6 \wedge targetValue' = limTargetValue$

31 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \rightarrow SystemState' = S_5$
32 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \rightarrow SystemState' = S_6$
33 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

34 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} > targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle \wedge \neg\, sensor\_signal_{12}$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

35 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} \leq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle \wedge \neg\, sensor\_signal_{12}$
$\rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

36 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue) \wedge \neg\, sensor\_signal_{12}$
$\rightarrow targetValue' = min(MaxCurrentValue, MaxTargetValue) \wedge SystemState' = S_4$

37 $SystemState = S_5 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}}$
$\rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

38 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} < targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \wedge \neg\, sensor\_signal_{12} \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

39 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} \geq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \wedge \neg\, sensor\_signal_{12} \rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

40 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue) \wedge \neg\, sensor\_signal_{12} \rightarrow targetValue' = max(MinCurrentValue, MinTargetValue) \wedge SystemState' = S_4$

41 $SystemState = S_6 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}} \rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

42 $SystemState = S_7 \wedge \mathsf{ti}(power1, t) = \langle\rangle \rightarrow SystemState' = S_7$

43 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal3, t) = \langle \mathsf{true} \rangle \rightarrow targetValue' = 0$
44 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{true} \rangle \rightarrow sValue' = V_1$
45 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{true} \rangle \rightarrow sValue' = V_2$
46 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal6, t) = \langle \mathsf{true} \rangle \rightarrow sValue' = V_3$

47 $SystemState \neq S_2 \wedge \neg \mathsf{ti}(power1, t) = \langle\rangle \wedge \mathsf{ti}(power2, t) = \langle\rangle \rightarrow SystemState' = S_1$

48 $(SystemState = S_4 \vee SystemState = S_5) \wedge \mathsf{ti}(signal_1, t) = \langle\rangle \rightarrow SystemState' = S_6$

where $sensor\_signal_{12}, limTargetValue$ so that
$sensor\_signal_{12} = sensor\_signal2^t_{\mathsf{ft}} \vee sensor\_signal1^t_{\mathsf{ft}}$
$limTargetValue = LimitedValue(current\_value^t_{\mathsf{ft}}, MinCurrentValue, MinTargetValue, MaxCurrentValue, MaxTargetValue)$

6.1.4 Decomposition: Local Variables

Now we apply the schema from Section 3.2 to get the component *LogicLoc_1* (we add _1 to its name, because we assume some underspecification here that must be clarified to get the component *LogicLoc*). The component *LogicLoc* will be discussed in the next section.

Here we present the decomposition process as well as the results of changes in *Logic* according to the schema: we call this specification *LogicNew* – this specification is just an intermediate result and will be used to extract from it the specifications *LogicMain* and *LogicOut* (see Sections 6.1.9 and 6.1.11).

First of all we determine the set of local variables that we want to move to the extra component as well as in which formulas of the specification *Logic* they appear, which dependencies they have with the input channels of this component and whether they influence on some other local variables or values on the output channels:

- The local variables pre_1, pre_2, pre_3, pre_4 and pre_5:
  - ◇ The values of these local variables at each time unit were defined by the formulas 14, ..., 18, using only the values of the input channels $precondition_1$, $precondition_2$, $precondition_3$, $precondition_4$ and $precondition_5$. Thus, we need to have these channels as input channels of the component *LogicLoc_1*.
  - ◇ The values of these local variables influence on the values of the local variable *SystemState* (formulas 19 and 20), which will be not removed from the main component.

- The local variable *sValue*:
  - ◇ The values of this local variable at each time unit is defined by the formulas 44, 45, and 46, using only the values of the input channels *sensor_signal*4, *sensor_signal*5, and *sensor_signal*6 as well as using the value of the local variable *SystemState*. Thus, we need to have the channels *sensor_signal*4, *sensor_signal*5, and *sensor_signal*6 as input channels of the component *LogicLoc_1*. The value of the local variable *SystemState* is equal (according to the formula 1 of the specification *Logic*) to the value of the output channel *stateInf*. We need to have this channel as input channels of the component *LogicLoc_1* also.
  - ◇ The value of this local variable influences on the values of the output channels variable $event_1$, $event_2$ and $event_3$ (formulas 8, 9, and 10).

Now we apply the decomposition scheme:

1. The set of input channels of the component *LogicLoc_1* is a subset of the

corresponding set of the component *Logic*.

$$stateInf : StateType;$$
$$sensor\_signal_4, sensor\_signal_5, sensor\_signal_6 : \mathbb{B}\text{ool}$$
$$precondition_1, precondition_2, precondition_3 : Event$$
$$precondition_4, precondition_5 : Event$$

2. We add all the assumptions about the input streams according to the specification *LogicLoc*:

$$\mathsf{ts}(sensor\_signal_4),\ \mathsf{ts}(sensor\_signal_5),\ \mathsf{ts}(sensor\_signal_6),$$
$$\mathsf{msg}_1(precondition_1),\ \mathsf{msg}_1(precondition_2),\ \mathsf{msg}_1(precondition_3),$$
$$\mathsf{msg}_1(precondition_4),\ \mathsf{msg}_1(precondition_5),$$
$$\mathsf{ts}(stateInf)$$

3. The output streams of *LogicLoc* correspond to the three local variables and have the same data as these variables. Let us call this streams by the variable names: $sValue$, pre_1, pre_2, pre_3, pre_4, and pre_5.

4. Move from the specification *Logic* to the specification *LogicLoc_1* all the formulas, in which the values of these local variables are defined: the formulas 14, ..., 18, 44, 45, and 46.

5. Add to the component *Logic* the following channels:
   - $sValue : SValueType$,
   - $pre_1, pre_2, pre_3, pre_4, pre_5 : \mathbb{B}\text{ool}$

   Add to the component *Logic* corresponding assumptions about these streams:

$$\mathsf{ts}(sValue)$$
$$\mathsf{ts}(pre_1),\ \mathsf{ts}(pre_2),\ \mathsf{ts}(pre_3),\ \mathsf{ts}(pre_4),\ \mathsf{ts}(pre_5)$$

6. Delete from the interface of *Logic* all the input streams that are used only in the formulas moved to the specification *LogicLoc*:

$$sensor\_signal_4, sensor\_signal_5, sensor\_signal_6 : \mathbb{B}\text{ool}$$
$$precondition_1, precondition_2, precondition_3 : Event$$
$$precondition_4, precondition_5 : Event$$

   Delete all the assumptions about these streams:

$$\mathsf{ts}(sensor\_signal_4),\ \mathsf{ts}(sensor\_signal_5),\ \mathsf{ts}(sensor\_signal_6),$$
$$\mathsf{msg}_1(precondition_1),\ \mathsf{msg}_1(precondition_2),\ \mathsf{msg}_1(precondition_3),$$
$$\mathsf{msg}_1(precondition_4),\ \mathsf{msg}_1(precondition_5)$$

7. Define in *LogicLoc_1* the initial value of the output streams according to the initial value of the local variables:

$$\mathsf{ti}(sValue, 0) = \langle V_1 \rangle$$
$$\mathsf{ti}(pre_1, 0) = \langle \mathsf{false} \rangle$$
$$\mathsf{ti}(pre_2, 0) = \langle \mathsf{false} \rangle$$
$$\mathsf{ti}(pre_3, 0) = \langle \mathsf{false} \rangle$$
$$\mathsf{ti}(pre_4, 0) = \langle \mathsf{false} \rangle$$
$$\mathsf{ti}(pre_5, 0) = \langle \mathsf{false} \rangle$$

8. Replace all the entrances of the local variables at the current time interval t by the values of the corresponding streams at this time interval:
   - $sValue$ by $\mathsf{ft.ti}(sValue, t)$: the formulas 8, 9, and 10.
   - The local variables pre_1, pre_2, pre_3, pre_4, and pre_5 are specified and used only for time interval $t+1$.

9. Replace
   - $sValue'$ by $\mathsf{ft.ti}(sValue, t+1)$: the moved formulas 44, 45, and 46.
   - pre'_1 by $\mathsf{ft.ti}(pre_1, t+1)$: the formulas 19 and 20 as well as the moved formula 14.
   - pre'_2 by $\mathsf{ft.ti}(pre_2, t+1)$: the formulas 19 and 20 as well as the moved formula 15.
   - pre'_3 by $\mathsf{ft.ti}(pre_3, t+1)$: the formulas 19 and 20 as well as the moved formula 16.
   - pre'_4 by $\mathsf{ft.ti}(pre_4, t+1)$: the formulas 19 and 20 as well as the moved formula 17.
   - pre'_5 by $\mathsf{ft.ti}(pre_5, t+1)$: the formulas 19 and 20 as well as the moved formula 18.

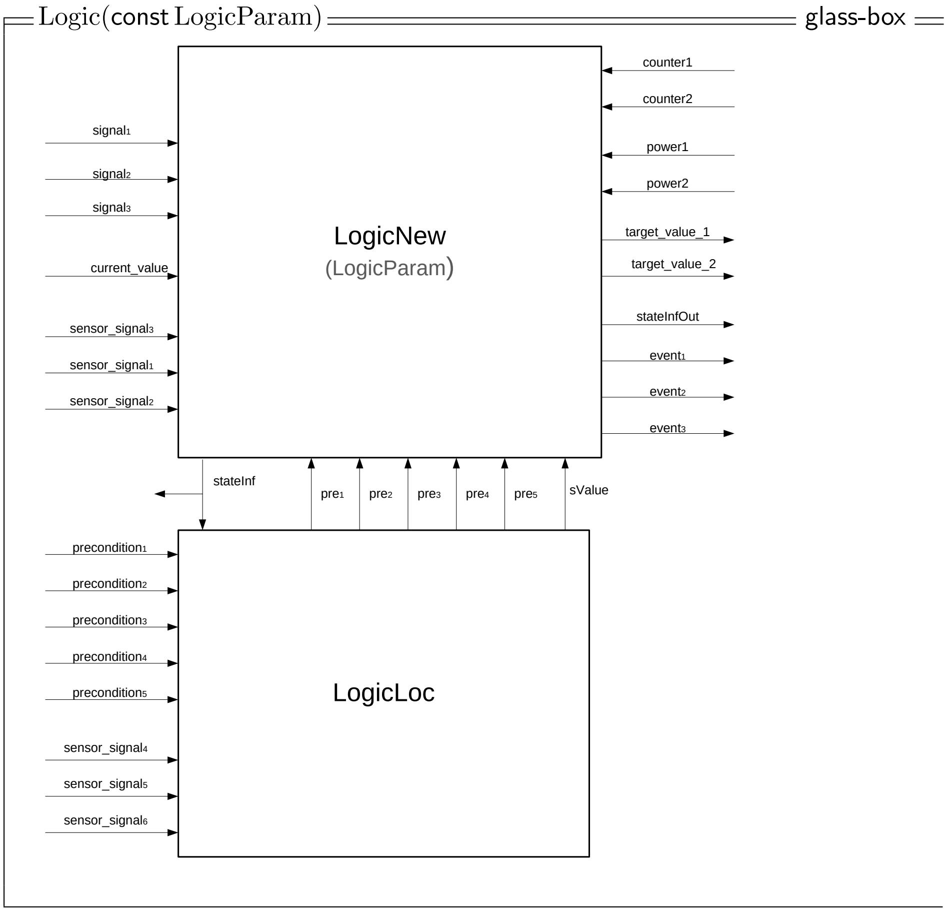
   respectively.

10. Delete from the specification *Logic* declarations of the local variables $sValue : SValueType$, $pre_1, pre_2, pre_3, pre_4, pre_5 : \mathbb{B}\text{ool}$, as well as the given definitions of their initial values:

$$\begin{array}{ll}
\mathsf{local} & SystemState : StateType; \\
 & \underline{pre_1, pre_2, pre_3, pre_4, pre_5 : \mathbb{B}\text{ool};} \\
 & tspeed : \mathbb{N};\ \underline{sValue : SValueType;} \\
\mathsf{init} & SystemState = S_0; \\
 & \underline{pre_1 = \mathsf{false};\ pre_2 = \mathsf{false};\ pre_3 = \mathsf{false};\ pre_4 = \mathsf{false};\ pre_5 = \mathsf{false};} \\
 & tspeed = 0; \\
 & \underline{sValue = V_1;}
\end{array}$$

11. We do not use in the specification *LogicLoc* any parameter of the component *Logic* – we do not need to (re)move any parameter.

The intermediate result of the decomposition is presented below by the new graphical specification of the component *Logic*:



LogicLoc_1 — timed

in

$stateInf : StateType;$
$sensor\_signal4, sensor\_signal5, sensor\_signal6 : \mathbb{B}\mathrm{ool}$
$precondition_1, precondition_2, precondition_3 : Event$
$precondition_4, precondition_5 : Event$

out

$sValue : SValueType;$
$pre_1, pre_2, pre_3, pre_4, pre_5 : \mathbb{B}\mathrm{ool}$

asm

$\mathsf{ts}(sensor\_signal4)$
$\mathsf{ts}(sensor\_signal5)$
$\mathsf{ts}(sensor\_signal6)$
$\mathsf{msg}_1(precondition_1) \ \wedge\ \mathsf{msg}_1(precondition_2) \ \wedge\ \mathsf{msg}_1(precondition_3)$
$\mathsf{msg}_1(precondition_4) \ \wedge\ \mathsf{msg}_1(precondition_5)$
$\mathsf{ts}(stateInf)$

gar

$\mathsf{ti}(sValue, 0) = \langle V_1 \rangle$
$\mathsf{ti}(pre_1, 0) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(pre_2, 0) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(pre_3, 0) = \langle \mathsf{false} \rangle$
$\mathsf{ti}(pre_4, 0) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(pre_5, 0) = \langle \mathsf{false} \rangle$

$\forall\, t \in \mathbb{N} :$
$\mathsf{ti}(precondition_1, t) \neq \langle\rangle \ \to\ \mathsf{ti}(pre_1, t+1) = \langle \mathsf{true} \rangle$
$\mathsf{ti}(precondition_2, t) \neq \langle\rangle \ \to\ \mathsf{ti}(pre_2, t+1) = \langle \mathsf{true} \rangle$
$\mathsf{ti}(precondition_3, t) \neq \langle\rangle \ \to\ \mathsf{ti}(pre_3, t+1) = \langle \mathsf{true} \rangle$
$\mathsf{ti}(precondition_4, t) \neq \langle\rangle \ \to\ \mathsf{ti}(pre_4, t+1) = \langle \mathsf{true} \rangle$
$\mathsf{ti}(precondition_5, t) \neq \langle\rangle \ \to\ \mathsf{ti}(pre_5, t+1) = \langle \mathsf{true} \rangle$

$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{true} \rangle$
$\quad \to\ \mathsf{ti}(sValue, t+1) = \langle V_1 \rangle$

$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle$
$\wedge\ \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{true} \rangle$
$\quad \to\ \mathsf{ti}(sValue, t+1) = \langle V_2 \rangle$

$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle$
$\wedge\ \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{false} \rangle$
$\wedge\ \mathsf{ti}(sensor\_signal6, t) = \langle \mathsf{true} \rangle$
$\quad \to\ \mathsf{ti}(sValue, t+1) = \langle V_3 \rangle$

Please note, that we do not change the enumeration of formulas in the specification *LogicNew*, thus, this specification has formulas with the following numbers: 1 – 13, 19 – 43, 47, 48.

LogicNew(const LogicParam) — timed

in
$sensor\_signal1, sensor\_signal2, sensor\_signal3 : \mathbb{B}ool$
$signal_1 : SignalType;\ \ current\_value, counter1, counter2 : \mathbb{N};\ power1, power2, signal_2, signal_3 : Event$
$sValue : SValueType;\ pre_1, pre_2, pre_3, pre_4, pre_5 : Event$

out
$event_1, event_2, event_3 : Event;\ target\_value\_1, target\_value\_2 : \mathbb{N};\ \ stateInf, stateInfOut : StateType$

local $SystemState : StateType;\ targetValue : \mathbb{N}$

init $SystemState = S_0;\ targetValue = 0;$

asm
$\mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2) \wedge \mathsf{ts}(sensor\_signal3)$
$\mathsf{msg}_1(signal_1) \wedge \mathsf{ts}(current\_value) \wedge \mathsf{ts}(counter1) \wedge \mathsf{ts}(counter2) \wedge \mathsf{msg}_1(power1) \wedge \mathsf{msg}_1(power2)$
$\mathsf{ts}(sValue) \wedge \mathsf{ts}(pre_1) \wedge \mathsf{ts}(pre_2) \wedge \mathsf{ts}(pre_3) \wedge \mathsf{ts}(pre_4) \wedge \mathsf{ts}(pre_5)$

gar

1 $stateInfOut = stateInf \wedge target\_value\_2 = target\_value\_1$

$\forall\, t \in \mathbb{N}:$

2 $\mathsf{ti}(stateInf, t) = \langle SystemState\rangle \wedge \mathsf{ti}(target\_value\_1, t) = \langle targetValue\rangle$

3 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge \neg\ Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_3$
4 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_2$

5 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true}\rangle \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false}\rangle \rightarrow SystemState' = S_2$

6 $SystemState = S_4 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle \wedge \neg sensor\_signal_{12} \wedge SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_5)$

7 $SystemState = S_4 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle \wedge \neg sensor\_signal_{12} \wedge SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_6)$

8 $SystemState \neq S_2 \wedge SystemState' = S_2 \wedge \mathsf{ft.ti}(sValue, t) = V_1 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_3, t+1) = \langle event\rangle$
9 $SystemState \neq S_2 \wedge SystemState' = S_2 \wedge \mathsf{ft.ti}(sValue, t) = V_2 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_1, t+1) = \langle event\rangle$
10 $SystemState \neq S_2 \wedge SystemState' = S_2 \wedge \mathsf{ft.ti}(sValue, t) = V_3 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_2, t+1) = \langle event\rangle$
11 $SystemState = S_2 \wedge \mathsf{ti}(signal_2, t) \neq \langle\rangle \wedge \mathsf{ti}(signal_3, t) \neq \langle\rangle \rightarrow \mathsf{ti}(event_3, t+1) = \langle event\rangle$

12 $SystemState = S_0 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \rightarrow targetValue' = 0 \wedge CrCtSate' = S_1$

13 $\mathsf{ti}(power1, t) = \langle\rangle \rightarrow CrCtSate' = S_0$

19 $SystemState = S_1 \wedge \mathsf{ft.ti}(pre_1, t+1) \wedge \mathsf{ft.ti}(pre_2, t+1) \wedge \mathsf{ft.ti}(pre_3, t+1) \wedge \mathsf{ft.ti}(pre_4, t+1) \wedge \mathsf{ft.ti}(pre_5, t+1) \rightarrow SystemState' = S_2$
20 $SystemState = S_1 \wedge (\neg\ \mathsf{ft.ti}(pre_1, t+1) \vee \neg\ \mathsf{ft.ti}(pre_2, t+1) \vee \neg\ \mathsf{ft.ti}(pre_3, t+1) \vee \neg\ \mathsf{ft.ti}(pre_4, t+1) \vee \neg\ \mathsf{ft.ti}(pre_5, t+1)) \rightarrow SystemState' = S_1$

21 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

22 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

23 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle \wedge \mathsf{ti}(counter2, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

24 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle \wedge \mathsf{ti}(counter1, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

25 $(SystemStateSubset(SystemState) \vee SystemState = S_2) \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{true}\rangle \rightarrow SystemState' = S_7$

26 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$
27 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue > 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle \rightarrow SystemState' = S_4 \wedge targetValue' = targetValue$
28 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue = 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle \rightarrow SystemState' = S_3$
29 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle \rightarrow SystemState' = S_5 \wedge targetValue' = limTargetValue$
30 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle \rightarrow SystemState' = S_6 \wedge targetValue' = limTargetValue$

31 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle \rightarrow SystemState' = S_5$
32 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle \rightarrow SystemState' = S_6$
33 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

34 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} > targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle \wedge \neg\ sensor\_signal_{12}$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

35 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} \leq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle \wedge \neg\ sensor\_signal_{12}$
$\rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

36 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue) \wedge \neg\ sensor\_signal_{12}$
$\rightarrow targetValue' = min(MaxCurrentValue, MaxTargetValue) \wedge SystemState' = S_4$

37 $SystemState = S_5 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}}$
$\rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

38 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} < targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle \wedge \neg\ sensor\_signal_{12} \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

39 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} \geq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle \wedge \neg\ sensor\_signal_{12} \rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

40 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue) \wedge \neg\ sensor\_signal_{12} \rightarrow targetValue' = max(MinCurrentValue, MinTargetValue) \wedge SystemState' = S_4$

41 $SystemState = S_6 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}} \rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

42 $SystemState = S_7 \wedge \mathsf{ti}(power1, t) = \langle\rangle \rightarrow SystemState' = S_7$

43 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal3, t) = \langle \mathsf{true}\rangle \rightarrow targetValue' = 0$

47 $SystemState \neq S_2 \wedge \neg\mathsf{ti}(power1, t) = \langle\rangle \wedge \mathsf{ti}(power2, t) = \langle\rangle \rightarrow SystemState' = S_1$

48 $(SystemState = S_4 \vee SystemState = S_5) \wedge \mathsf{ti}(signal_1, t) = \langle\rangle \rightarrow SystemState' = S_6$

where $sensor\_signal_{12}, limTargetValue$ so that
$sensor\_signal_{12} = sensor\_signal2^t_{\mathsf{ft}} \vee sensor\_signal1^t_{\mathsf{ft}}$
$limTargetValue = LimitedValue(current\_value^t_{\mathsf{ft}}, MinCurrentValue, MinTargetValue, MaxCurrentValue, MaxTargetValue)$

6.1.5 LogicLoc Component

Now, focusing on the small part of the specification *Logic*, it is easier to see in the specification *LogicLoc_1* that the specification contains no formula describing the case that the shutdown value remains unchanged if the predicate *SystemStateSubset* does not hold for the current state or if all the signals $sensor\_signal4$, $sensor\_signal5$, $sensor\_signal6$ are false. We can refine to the specification *LogicLoc* by the following formula

$$
\begin{aligned}
&\neg SystemStateSubset(stateInf_{\mathsf{ft}}^{t}) \lor \\
&(\mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle \\
&\ \ \land\ \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{false} \rangle \\
&\ \ \land\ \mathsf{ti}(sensor\_signal6, t) = \langle \mathsf{false} \rangle) \\
&\qquad \to\ \mathsf{ti}(sValue, t+1) = \mathsf{ti}(sValue, t)
\end{aligned}
$$

but it is a bad style to define the value of output stream at time $t + 1$ by the value of the same stream at time t. Moreover, if we want to translate later this Focus specification to an AutoFocus model, we need to ovoid such a situation – such kind of definitions cannot be used on the AutoFocus layer at all. Thus, we need to use a local variable to safe this value.

The same holds for the values of pre_1, pre_2, pre_3, pre_4 and pre_5: the specification contains no formula describing the case that the values of pre_1, pre_2, pre_3, pre_4 and pre_5 remain unchanged after setting them to true.

We can refine to the specification *CrCtLogicLoc* by by rewriting the formulas

$$
\begin{aligned}
&\mathsf{ti}(precondition_1, t) \neq \langle\rangle\ \to\ \mathsf{ti}(pre_1, t+1) = \langle \mathsf{true} \rangle \\
&\mathsf{ti}(precondition_2, t) \neq \langle\rangle\ \to\ \mathsf{ti}(pre_2, t+1) = \langle \mathsf{true} \rangle \\
&\mathsf{ti}(precondition_3, t) \neq \langle\rangle\ \to\ \mathsf{ti}(pre_3, t+1) = \langle \mathsf{true} \rangle \\
&\mathsf{ti}(precondition_4, t) \neq \langle\rangle\ \to\ \mathsf{ti}(pre_4, t+1) = \langle \mathsf{true} \rangle \\
&\mathsf{ti}(precondition_5, t) \neq \langle\rangle\ \to\ \mathsf{ti}(pre_5, t+1) = \langle \mathsf{true} \rangle
\end{aligned}
$$

to the following ones

$$
\begin{aligned}
&\mathsf{ti}(precondition_1, t) \neq \langle\rangle\ \to\ (\forall\, i \in \mathbb{N} : t < i \to \mathsf{ti}(pre_1, i) = \langle \mathsf{true} \rangle) \\
&\mathsf{ti}(precondition_2, t) \neq \langle\rangle\ \to\ (\forall\, i \in \mathbb{N} : t < i \to \mathsf{ti}(pre_2, i) = \langle \mathsf{true} \rangle) \\
&\mathsf{ti}(precondition_3, t) \neq \langle\rangle\ \to\ (\forall\, i \in \mathbb{N} : t < i \to \mathsf{ti}(pre_3, i) = \langle \mathsf{true} \rangle) \\
&\mathsf{ti}(precondition_4, t) \neq \langle\rangle\ \to\ (\forall\, i \in \mathbb{N} : t < i \to \mathsf{ti}(pre_4, i) = \langle \mathsf{true} \rangle) \\
&\mathsf{ti}(precondition_5, t) \neq \langle\rangle\ \to\ (\forall\, i \in \mathbb{N} : t < i \to \mathsf{ti}(pre_5, i) = \langle \mathsf{true} \rangle)
\end{aligned}
$$

But to specify the behavior of a component as a step-by-step one, similar to state transition diagram, we need to use a local variable to safe the corresponding value. Thus, we need to use a local variable to represent to safe this value, we refine the *LogicLoc_1* to the specification *LogicLoc*.

LogicLoc — timed

in
$stateInf : StateType;$
$sensor\_signal4, sensor\_signal5, sensor\_signal6 : \mathbb{B}ool$
$precondition_1, precondition_2, precondition_3 : Event$
$precondition_4, precondition_5 : Event$

out $sValue : SValueType;\ pre_1, pre_2, pre_3, pre_4, pre_5 : \mathbb{B}ool$

local $sValue : SValueType;\ Init_1, Init_2, Init_3, Init_4, Init_5 : \mathbb{B}ool$

init $Init_1 = \mathsf{false};\ Init_2 = \mathsf{false};\ Init_3 = \mathsf{false};\ Init_4 = \mathsf{false};\ Init_5 = \mathsf{false}$

asm
$\mathsf{ts}(sensor\_signal4) \ \wedge\ \mathsf{ts}(sensor\_signal5) \ \wedge\ \mathsf{ts}(sensor\_signal6)$
$\mathsf{msg}_1(precondition_1) \ \wedge\ \mathsf{msg}_1(precondition_2) \ \wedge\ \mathsf{msg}_1(precondition_3)$
$\mathsf{msg}_1(precondition_4) \ \wedge\ \mathsf{msg}_1(precondition_5)$
$\mathsf{ts}(stateInf)$

gar

$\mathsf{ti}(sValue, 0) = \langle V_1 \rangle$
$\mathsf{ti}(pre_1, 0) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(pre_2, 0) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(pre_3, 0) = \langle \mathsf{false} \rangle$
$\mathsf{ti}(pre_4, 0) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(pre_5, 0) = \langle \mathsf{false} \rangle$

$\forall\, t \in \mathbb{N} :$
$\mathsf{ti}(precondition_1, t) \neq \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_1, t+1) = \langle \mathsf{true} \rangle \ \wedge\ Init'_1 = \mathsf{true}$
$\mathsf{ti}(precondition_2, t) \neq \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_2, t+1) = \langle \mathsf{true} \rangle \ \wedge\ Init'_2 = \mathsf{true}$
$\mathsf{ti}(precondition_3, t) \neq \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_3, t+1) = \langle \mathsf{true} \rangle \ \wedge\ Init'_3 = \mathsf{true}$
$\mathsf{ti}(precondition_4, t) \neq \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_4, t+1) = \langle \mathsf{true} \rangle \ \wedge\ Init'_4 = \mathsf{true}$
$\mathsf{ti}(precondition_5, t) \neq \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_5, t+1) = \langle \mathsf{true} \rangle \ \wedge\ Init'_5 = \mathsf{true}$

$\mathsf{ti}(precondition_1, t) = \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_1, t+1) = \langle Init_1 \rangle \ \wedge\ Init'_1 = Init_1$
$\mathsf{ti}(precondition_2, t) = \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_2, t+1) = \langle Init_2 \rangle \ \wedge\ Init' = Init_2$
$\mathsf{ti}(precondition_3, t) = \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_3, t+1) = \langle Init_3 \rangle \ \wedge\ Init' = Init_3$
$\mathsf{ti}(precondition_4, t) = \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_4, t+1) = \langle Init_4 \rangle \ \wedge\ Init' = Init_4$
$\mathsf{ti}(precondition_5, t) = \langle\rangle \ \rightarrow\ \mathsf{ti}(pre_5, t+1) = \langle Init_5 \rangle \ \wedge\ Init' = Init_5$

$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{true} \rangle$
$\quad \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle V_1 \rangle \ \wedge\ sValue' = V_1$

$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle$
$\wedge\ \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{true} \rangle$
$\quad \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle V_2 \rangle \ \wedge\ sValue' = V_2$

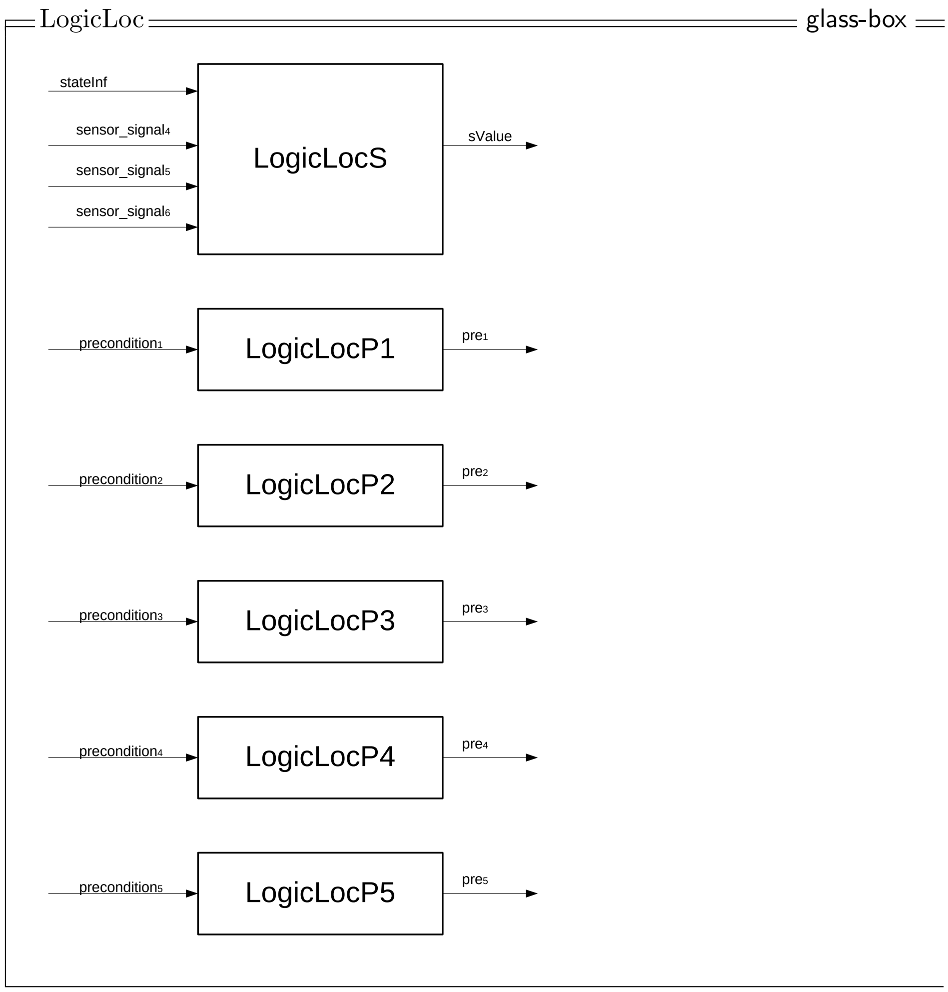
$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle$
$\wedge\ \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{false} \rangle$
$\wedge\ \mathsf{ti}(sensor\_signal6, t) = \langle \mathsf{true} \rangle$
$\quad \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle V_3 \rangle \ \wedge\ sValue' = V_3$

$\neg SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \vee\ (\mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle \ \wedge$
$\quad \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{false} \rangle \ \wedge$
$\quad \mathsf{ti}(sensor\_signal6, t) = \langle \mathsf{false} \rangle)$
$\qquad \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle sValue \rangle \ \wedge\ sValue' = sValue$

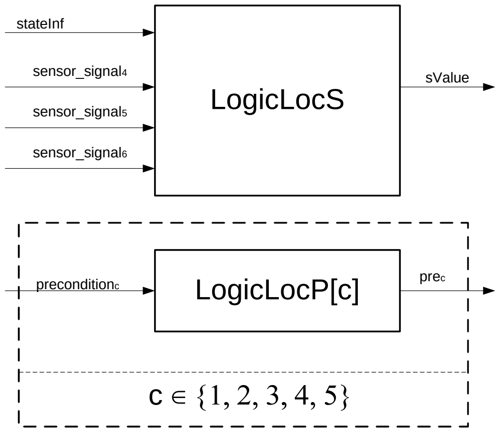
6.1.6 LogicLoc Component: Parallel Decomposition

To have more clear architecture we can decompose the component *LogicLoc* into three subcomponents, *LogicLocS*, *LogicLocP*1, *LogicLocP*2, *LogicLocP*3, *LogicLocP*4 and *LogicLocP*5 by the kind of working with the local variables. Please note that the component *LogicLoc* is a parallel composition of these subcomponents: they work independently.

The components *LogicLocP*1, *LogicLocP*2, *LogicLocP*3, *LogicLocP*4 and *LogicLocP*5 have the same structure and can be specified as two different instances of one component, i.e. represented using specification replication (see below).



LogicLoc — glass-box



LogicLocP — timed

in $precondition : Event$

out $pre : \mathbb{B}ool$

local $p : \mathbb{B}ool$

init $p = \mathsf{false}$

asm $\mathsf{msg}_1(precondition)$

gar

$\mathsf{ti}(pre, 0) = \langle \mathsf{false} \rangle$

$\forall\, t \in \mathbb{N}:$

$\mathsf{ti}(precondition, t) \neq \langle\rangle \;\rightarrow\; \mathsf{ti}(pre, t+1) = \langle \mathsf{true} \rangle \;\wedge\; p' = \mathsf{true}$

$\mathsf{ti}(precondition, t) = \langle\rangle \;\rightarrow\; \mathsf{ti}(pre, t+1) = \langle p \rangle \;\wedge\; p' = p$

LogicLocS — timed

in	$stateInf : StateType$; $sensor\_signal4 : \mathbb{B}ool$ $sensor\_signal5 : \mathbb{B}ool$ $sensor\_signal6 : \mathbb{B}ool$
out	$sValue : SValueType$
local	$sValue : SValueType$
asm	$\mathsf{ts}(sensor\_signal4)$ $\mathsf{ts}(sensor\_signal5)$ $\mathsf{ts}(sensor\_signal6)$ $\mathsf{ts}(stateInf)$

gar

$$\mathsf{ti}(sValue, 0) = \langle V_1 \rangle$$

$$\forall\, t \in \mathbb{N} :$$

$$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{true} \rangle \ \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle V_1 \rangle \ \wedge\ sValue' = V_1$$

$$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{true} \rangle \ \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle V_2 \rangle \ \wedge\ sValue' = V_2$$

$$SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \wedge\ \mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal6, t) = \langle \mathsf{true} \rangle \ \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle V_3 \rangle \ \wedge\ sValue' = V_3$$

$$\neg SystemStateSubset(stateInf^{t}_{\mathsf{ft}}) \ \vee\ (\mathsf{ti}(sensor\_signal4, t) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(sensor\_signal5, t) = \langle \mathsf{false} \rangle \ \wedge\ \mathsf{ti}(sensor\_signal6, t) = \langle \mathsf{false} \rangle) \ \rightarrow\ \mathsf{ti}(sValue, t+1) = \langle sValue \rangle \ \wedge\ sValue' = sValue$$

6.1.7 LogicLoc Subcomponent: Timed State Transition Diagrams

The specification *LogicLocP* is semantically equal to the specification using a timed state transition diagram, which two states, *pFalse* and *pTrue*, according to the value of the local variable p. We take *pFalse* as the initial state, because of to the initial value of this variable.

The formula $\mathsf{ti}(pre, 0) = \langle \mathsf{false} \rangle$ defines the starting output value, where the formulas

$$\mathsf{ti}(precondition, t) \neq \langle\rangle \;\rightarrow\; \mathsf{ti}(pre, t+1) = \langle \mathsf{true} \rangle \;\wedge\; p' = \mathsf{true}$$
$$\mathsf{ti}(precondition, t) = \langle\rangle \;\rightarrow\; \mathsf{ti}(pre, t+1) = \langle p \rangle \;\wedge\; p' = p$$

describe state transitions with corresponding inputs.

We do not have precondition about the value of p on the left part of implication – this means, that both formulas must hold for each state:

$$p' = \mathsf{true} \;\wedge\; \mathsf{ti}(precondition, t) \neq \langle\rangle \;\rightarrow\; \mathsf{ti}(p, t+1) = \langle \mathsf{true} \rangle \;\wedge\; p' = \mathsf{true}$$

$$p' = \mathsf{true} \;\wedge\; \mathsf{ti}(precondition, t) = \langle\rangle \;\rightarrow\; \mathsf{ti}(pre, t+1) = \langle p \rangle \;\wedge\; p' = p$$

$$p' = \mathsf{false} \;\wedge\; \mathsf{ti}(precondition, t) \neq \langle\rangle$$
$$\rightarrow\; \mathsf{ti}(pre, t+1) = \langle \mathsf{true} \rangle \;\wedge\; p' = \mathsf{true}$$

$$p' = \mathsf{false} \;\wedge\; \mathsf{ti}(precondition, t) = \langle\rangle$$
$$\rightarrow\; \mathsf{ti}(pre, t+1) = \langle p \rangle \;\wedge\; p' = p$$

We can easily see that the first two formulas can be simplified to a single one:

$$p' = \mathsf{true} \;\rightarrow\; \mathsf{ti}(pre, t+1) = \langle \mathsf{true} \rangle \;\wedge\; p' = \mathsf{true}$$

The corresponding timed state transition diagram for the component *LogicLocP* is presented on Figure 1.

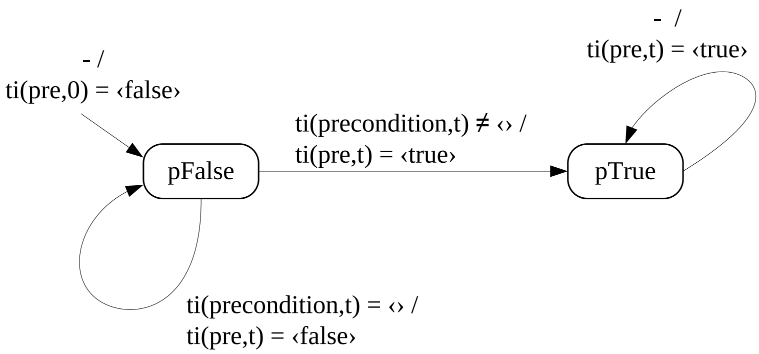


Figure 1: Timed state transition diagram for the component LogicLocP

The specification *LogicLocS* is semantically equal to the specification using a simple state transition diagram, which has three states, let call them V_1, V_2 and V_3. The corresponding timed state transition diagram is presented on Figure 2.

Please note, that we cannot mark here an initial state, because no initial value of the variable *sValue* is given in the specification *LogicLoc* (and as result also in the specification *LogicLocS*).

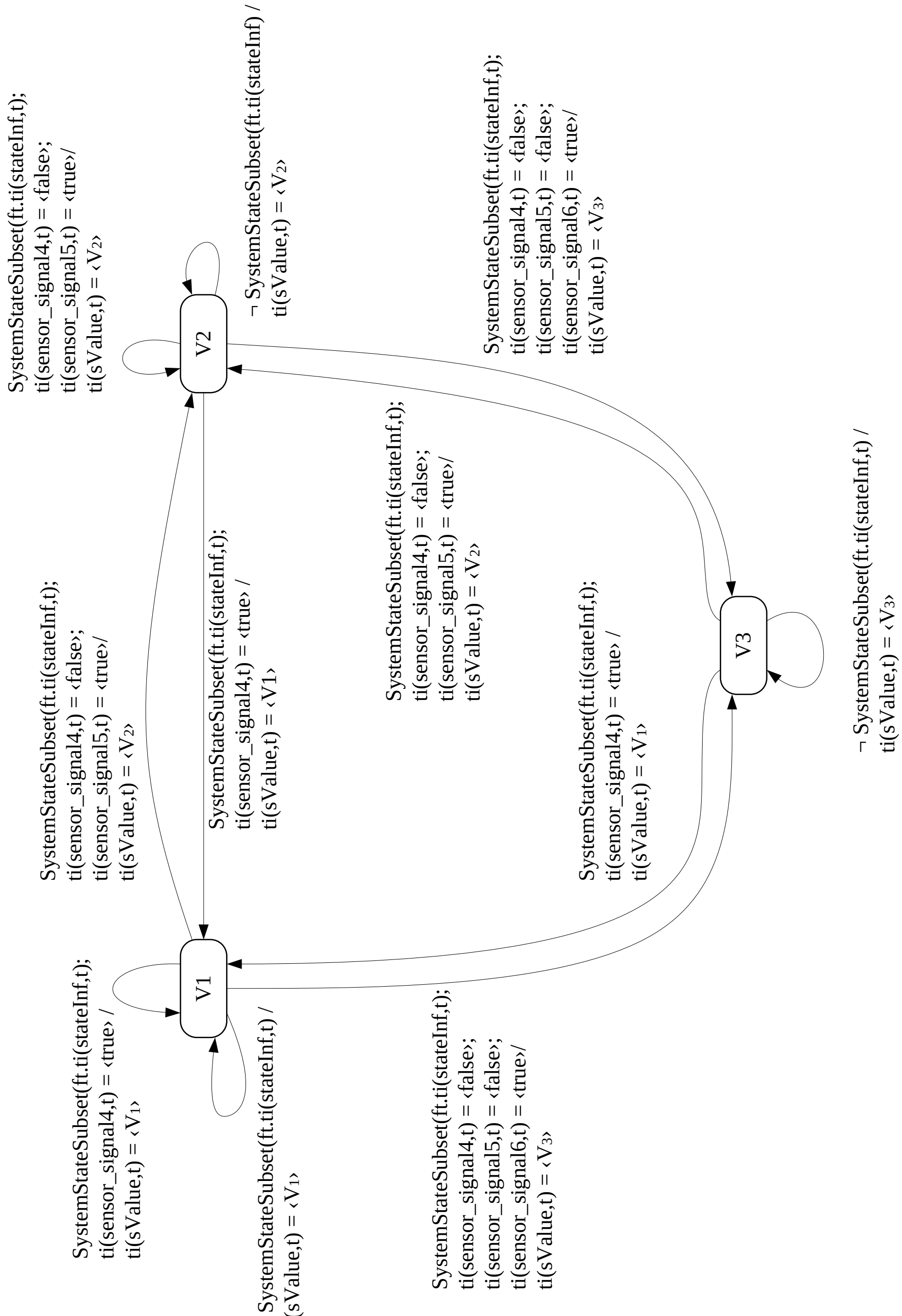


Figure 2: Timed state transition diagram for the component LogicLocS

6.1.8 Decomposition: Outputs That Depends from Inputs

Now we need apply the schema from Section 3.3 to get the components *LogicMain* and *LogicOut* from the component *LogicNew* presented in Section 6.1.

For this purpose we have to extract the specification *LogicNew* according to all the steps from Section 3.2. Now we can see in the specification *LogicNew* that there a number of formulas, describing a number of output streams depend only on the component state, local variables and some inputs, s.t. these formulas do not describe any requirement on the state changes. These are formulas 8 – 11, which describe outputs $event_1$, $event_2$, and $event_3$. The is also the 43d formula that does not describe any requirement on the state changes, but this formula describe requirements on the local variable *targetValue*, therefore it will be no advantage to move this formula out.

1. The formulas to extract from the component *LogicNew* to the component *LogicOut* contain the local variable *SystemState*, but the value of this variable at any time interval t is equal to the value of its output stream *stateInf* at this time interval (see the second formula in the guarantee-part of the specification *LogicNew*). Thus, we do not need any extensions of *LogicNew*, but we need to change the 8th, 9th, 10th and 11th formulas of *LogicNew* as follows:
   - $SystemState \neq S_2$ must be replaced by $stateInf^{t}_{\mathsf{ft}} \neq S_2$,
   - $SystemState = S_2$ must be replaced by $stateInf^{t}_{\mathsf{ft}} = S_2$, and
   - $SystemState' = S_2$ must be replaced by $stateInf^{t+1}_{\mathsf{ft}} = S_2$.
2. The set of input channels of the component *LogicOut* is a subset of the corresponding set of the component *LogicNew*
   - *sValue*,
   - *sensor_signal*1,
   - *sensor_signal*2,
   - *signal*2, and
   - *signal*3

   together with this output of the component *LogicNew* that presents value of local variable *SystemState* of this component:
   - *stateInf*.

   In the notation from [2]:

   $$i_{LogicOut} \subseteq (i_{LogicNew} \cup o_{LogicNew})$$

   where the set of output channels of the component *LogicOut* is only the set of output channels moved from *LogicNew* to*LogicOut*:
   - $event_1$,
   - $event_2$, and
   - $event_3$

We remove these outputs from the definition of *LogicNew*.

3. Add to the specification *LogicOut* all the assumptions about its input streams according to the specification *LogicNew*:

   $\mathsf{ts}(sValue)$

   $\mathsf{ts}(sensor\_signal1)$

   $\mathsf{ts}(sensor\_signal2)$

   $\mathsf{msg}_1(signal_2)$

   $\mathsf{msg}_1(signal_3)$

4. Values of the following input streams of *LogicNew* are used only in the formulas to extract to the component *LogicOut*:

   $sValue : SValueType$

   $signal_2, signal_3 : Event$

   We remove these inputs from interface of the component *LogicNew*.

5. Delete from the specification *LogicNew* all the assumptions about the input streams that are removed according the previous step:

   $\mathsf{ts}(sValue)$

6. Add to the specification *LogicOut* the assumption about all the extra channels:

   $\mathsf{ts}(stateInf)$

7. Move all corresponding formulas from the specification *LogicNew* to the specification *LogicOut*.

8. We do not use in the specification *LogicOut* any parameter of the component *LogicNew* – we do not need to (re)move any parameter.

Now we get the first versions of the components *LogicMain* (see also Section 6.1.11) and *LogicOut* (see also Section 6.1.9), we denote this adding _1 to the specification names.

Please note, that we do not change the enumeration of formulas in the specification *LogicMain_1*, thus, this specification has formulas with the following numbers: 1 – 7, 12, 13, 19 – 43, 47, 48. After that we group the formulas by the current system state and update the enumeration to get the specification *LogicMain_2*.

LogicMain_1(const LogicParam) — timed

in $sensor\_signal1, sensor\_signal2, sensor\_signal3 : \mathbb{B}ool;\ signal_1 : SignalType$
$current\_value, counter1, counter2 : \mathbb{N};\ pre_1, pre_2, pre_3, pre_4, pre_5, power1, power2 : Event$

out $target\_value\_1, target\_value\_2 : \mathbb{N};\ \ stateInf, stateInfOut : StateType$

local $SystemState : StateType;\ targetValue : \mathbb{N}$

init $SystemState = S_0;\ targetValue = 0;$

asm $\mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2) \wedge \mathsf{ts}(sensor\_signal3)$
$\mathsf{msg}_1(signal_1) \wedge \mathsf{ts}(current\_value) \wedge \mathsf{ts}(counter1) \wedge \mathsf{ts}(counter2) \wedge \mathsf{msg}_1(power1) \wedge \mathsf{msg}_1(power2)$
$\mathsf{ts}(pre_1) \wedge \mathsf{ts}(pre_2) \wedge \mathsf{ts}(pre_3) \wedge \mathsf{ts}(pre_4) \wedge \mathsf{ts}(pre_5)$

gar

1 $stateInfOut = stateInf \wedge target\_value\_2 = target\_value\_1$

$\forall\, t \in \mathbb{N}:$

2 $\mathsf{ti}(stateInf, t) = \langle SystemState\rangle \wedge \mathsf{ti}(target\_value\_1, t) = \langle targetValue\rangle$

3 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge \neg\, Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_3$
4 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_2$

5 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true}\rangle \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false}\rangle \rightarrow SystemState' = S_2$

6 $SystemState = S_4 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle \wedge \neg sensor\_signal_{12} \wedge SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_5)$

7 $SystemState = S_4 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle \wedge \neg sensor\_signal_{12} \wedge SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_6)$

12 $SystemState = S_0 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \rightarrow targetValue' = 0 \wedge CrCtSate' = S_1$

13 $\mathsf{ti}(power1, t) = \langle\rangle \rightarrow CrCtSate' = S_0$

19 $SystemState = S_1 \wedge \mathsf{ft.ti}(pre_1, t+1) \wedge \mathsf{ft.ti}(pre_2, t+1) \wedge \mathsf{ft.ti}(pre_3, t+1) \wedge \mathsf{ft.ti}(pre_4, t+1) \wedge \mathsf{ft.ti}(pre_5, t+1) \rightarrow SystemState' = S_2$
20 $SystemState = S_1 \wedge (\neg\, \mathsf{ft.ti}(pre_1, t+1) \vee \neg\, \mathsf{ft.ti}(pre_2, t+1) \vee \neg\, \mathsf{ft.ti}(pre_3, t+1) \vee \neg\, \mathsf{ft.ti}(pre_4, t+1) \vee \neg\, \mathsf{ft.ti}(pre_5, t+1)) \rightarrow SystemState' = S_1$

21 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

22 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

23 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle \wedge \mathsf{ti}(counter2, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

24 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle \wedge \mathsf{ti}(counter1, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

25 $(SystemStateSubset(SystemState) \vee SystemState = S_2) \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{true}\rangle \rightarrow SystemState' = S_7$

26 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$
27 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue > 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle \rightarrow SystemState' = S_4 \wedge targetValue' = targetValue$
28 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue = 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle \rightarrow SystemState' = S_3$
29 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle \rightarrow SystemState' = S_5 \wedge targetValue' = limTargetValue$
30 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle \rightarrow SystemState' = S_6 \wedge targetValue' = limTargetValue$

31 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle \rightarrow SystemState' = S_5$
32 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle \rightarrow SystemState' = S_6$
33 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

34 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} > targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle \wedge \neg\, sensor\_signal_{12}$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

35 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} \leq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle \wedge \neg\, sensor\_signal_{12}$
$\rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

36 $SystemState = S_5 \wedge current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue) \wedge \neg\, sensor\_signal_{12}$
$\rightarrow targetValue' = min(MaxCurrentValue, MaxTargetValue) \wedge SystemState' = S_4$

37 $SystemState = S_5 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}}$
$\rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

38 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} < targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle \wedge \neg\, sensor\_signal_{12} \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

39 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} \geq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle \wedge \neg\, sensor\_signal_{12} \rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

40 $SystemState = S_6 \wedge current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue) \wedge \neg\, sensor\_signal_{12} \rightarrow targetValue' = max(MinCurrentValue, MinTargetValue) \wedge SystemState' = S_4$

41 $SystemState = S_6 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}} \rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

42 $SystemState = S_7 \wedge \mathsf{ti}(power1, t) = \langle\rangle \rightarrow SystemState' = S_7$

43 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal3, t) = \langle \mathsf{true}\rangle \rightarrow targetValue' = 0$

47 $SystemState \neq S_2 \wedge \neg \mathsf{ti}(power1, t) = \langle\rangle \wedge \mathsf{ti}(power2, t) = \langle\rangle \rightarrow SystemState' = S_1$

48 $(SystemState = S_4 \vee SystemState = S_5) \wedge \mathsf{ti}(signal_1, t) = \langle\rangle \rightarrow SystemState' = S_6$

where $sensor\_signal_{12}, limTargetValue$ so that
$sensor\_signal_{12} = sensor\_signal2^t_{\mathsf{ft}} \vee sensor\_signal1^t_{\mathsf{ft}}$
$limTargetValue = LimitedValue(current\_value^t_{\mathsf{ft}}, MinCurrentValue, MinTargetValue, MaxCurrentValue, MaxTargetValue)$

LogicMain_2(const LogicParam) — timed

in: $sensor\_signal1, sensor\_signal2, sensor\_signal3 : \mathbb{B}\text{ool};\ signal_1 : SignalType$
$current\_value, counter1, counter2 : \mathbb{N};\ pre_1, pre_2, pre_3, pre_4, pre_5, power1, power2 : Event$

out: $target\_value\_1, target\_value\_2 : \mathbb{N};\ \ stateInf, stateInfOut : StateType$

local: $SystemState : StateType;\ targetValue : \mathbb{N}$

init: $SystemState = S_0;\ targetValue = 0;$

asm: $\mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2) \wedge \mathsf{ts}(sensor\_signal3)$
$\mathsf{msg}_1(signal_1) \wedge \mathsf{ts}(current\_value) \wedge \mathsf{ts}(counter1) \wedge \mathsf{ts}(counter2) \wedge \mathsf{msg}_1(power1) \wedge \mathsf{msg}_1(power2)$
$\mathsf{ts}(pre_1) \wedge \mathsf{ts}(pre_2) \wedge \mathsf{ts}(pre_3) \wedge \mathsf{ts}(pre_4) \wedge \mathsf{ts}(pre_5)$

gar

1 $stateInfOut = stateInf \wedge target\_value\_2 = target\_value\_1$

$\forall t \in \mathbb{N}:$

2 $\mathsf{ti}(stateInf, t) = \langle SystemState \rangle \wedge \mathsf{ti}(target\_value\_1, t) = \langle targetValue \rangle$

3 $\mathsf{ti}(power1, t) = \langle\rangle \rightarrow CrCtSate' = S_0$

4 $SystemState = S_0 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \rightarrow targetValue' = 0 \wedge CrCtSate' = S_1$

5 $SystemState = S_1 \wedge \mathsf{ft.ti}(pre_1, t+1) \wedge \mathsf{ft.ti}(pre_2, t+1) \wedge \mathsf{ft.ti}(pre_3, t+1) \wedge \mathsf{ft.ti}(pre_4, t+1) \wedge \mathsf{ft.ti}(pre_5, t+1) \rightarrow SystemState' = S_2$
6 $SystemState = S_1 \wedge (\neg\, \mathsf{ft.ti}(pre_1, t+1) \vee \neg\, \mathsf{ft.ti}(pre_2, t+1) \vee \neg\, \mathsf{ft.ti}(pre_3, t+1) \vee \neg\, \mathsf{ft.ti}(pre_4, t+1) \vee \neg\, \mathsf{ft.ti}(pre_5, t+1)) \rightarrow SystemState' = S_1$

7 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge \neg\, Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_3$
8 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_2$

9 $(SystemStateSubset(SystemState) \vee SystemState = S_2) \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{true} \rangle \rightarrow SystemState' = S_7$

10 $SystemState \neq S_2 \wedge \neg\mathsf{ti}(power1, t) = \langle\rangle \wedge \mathsf{ti}(power2, t) = \langle\rangle \rightarrow SystemState' = S_1$

11 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$
12 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue > 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4 \rangle \rightarrow SystemState' = S_4 \wedge targetValue' = targetValue$
13 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue = 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4 \rangle \rightarrow SystemState' = S_3$
14 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \rightarrow SystemState' = S_5 \wedge targetValue' = limTargetValue$
15 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \rightarrow SystemState' = S_6 \wedge targetValue' = limTargetValue$

16 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

17 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_5)$

18 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

19 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge \mathsf{ti}(counter2, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

20 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4 \wedge targetValue' = ChangeTargetValue(targetValue, SignalA_6)$

21 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

22 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge \mathsf{ti}(counter1, t) > 0$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

23 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \rightarrow SystemState' = S_5$
24 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \rightarrow SystemState' = S_6$
25 $(SystemState = S_4 \vee SystemState = S_5) \wedge \mathsf{ti}(signal_1, t) = \langle\rangle \rightarrow SystemState' = S_6$

26 $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} > targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle$
$\rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

27 $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \leq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle$
$\rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

28 $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue)$
$\rightarrow targetValue' = min(MaxCurrentValue, MaxTargetValue) \wedge SystemState' = S_4$

29 $SystemState = S_5 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}}$
$\rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

30 $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} < targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \rightarrow targetValue' = limTargetValue \wedge SystemState' = S_4$

31 $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \geq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \rightarrow SystemState' = S_4 \wedge targetValue' \neq 0$

32 $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue) \rightarrow targetValue' = max(MinCurrentValue, MinTargetValue) \wedge SystemState' = S_4$

33 $SystemState = S_6 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}} \rightarrow targetValue' = targetValue \wedge SystemState' = S_2$

34 $SystemState = S_7 \wedge \mathsf{ti}(power1, t) = \langle\rangle \rightarrow SystemState' = S_7$

35 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal3, t) = \langle \mathsf{true} \rangle \rightarrow targetValue' = 0$

36 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true} \rangle \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle \rightarrow SystemState' = S_2$

where $sensor\_signal_{12}, limTargetValue$ so that
$sensor\_signal_{12} = sensor\_signal2^t_{\mathsf{ft}} \vee sensor\_signal1^t_{\mathsf{ft}}$
$limTargetValue = LimitedValue(current\_value^t_{\mathsf{ft}}, MinCurrentValue, MinTargetValue, MaxCurrentValue, MaxTargetValue)$

LogicOut_1 — timed

in	$stateInf : StateType;\ sValue : SValueType;$ $signal_2, signal_3 : Event;$ $sensor\_signal1, sensor\_signal2 : \mathbb{B}ool$
out	$event_1, event_2, event_3 : Event$
asm	$\mathsf{ts}(stateInf) \wedge \mathsf{ts}(sValue) \wedge \mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2)$ $\mathsf{msg}_1(signal_2) \wedge \mathsf{msg}_1(signal_3)$

gar

$$stateInf^t_{\mathsf{ft}} \neq S_2 \wedge stateInf^{t+1}_{\mathsf{ft}} = S_2 \wedge sValue^t_{\mathsf{ft}} = V_1 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \to \mathsf{ti}(event_3, t+1) = \langle event \rangle$$

$$stateInf^t_{\mathsf{ft}} \neq S_2 \wedge stateInf^{t+1}_{\mathsf{ft}} = S_2 \wedge sValue^t_{\mathsf{ft}} = V_2 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \to \mathsf{ti}(event_1, t+1) = \langle event \rangle$$

$$stateInf^t_{\mathsf{ft}} \neq S_2 \wedge stateInf^{t+1}_{\mathsf{ft}} = S_2 \wedge sValue^t_{\mathsf{ft}} = V_3 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \to \mathsf{ti}(event_2, t+1) = \langle event \rangle$$

$$stateInf^t_{\mathsf{ft}} = S_2 \wedge \mathsf{ti}(signal_2, t) \neq \langle\rangle \wedge \mathsf{ti}(signal_3, t) \neq \langle\rangle \to \mathsf{ti}(event_3, t+1) = \langle event \rangle$$

6.1.9 LogicOut Component

As we can easily see now, the specification *LogicOut_1* is only weak causal: its values of its output streams at the time interval $t+1$ depend on the values of the input stream *stateInf* at the same time interval. Because we want to have this component as a causal one, we need to change the first three formulas as follows:[5]

$$stateInf^t_{\mathsf{ft}} \neq S_2 \wedge stateInf^{t+1}_{\mathsf{ft}} = S_2 \wedge sValue^{t+1}_{\mathsf{ft}} = V_1 \wedge sensor\_signal2^{t+1}_{\mathsf{ft}} \wedge \neg sensor\_signal1^{t+1}_{\mathsf{ft}} \to \mathsf{ti}(event_3, t+2) = \langle event \rangle$$

$$stateInf^t_{\mathsf{ft}} \neq S_2 \wedge stateInf^{t+1}_{\mathsf{ft}} = S_2 \wedge sValue^{t+1}_{\mathsf{ft}} = V_2 \wedge sensor\_signal2^{t+1}_{\mathsf{ft}} \wedge \neg sensor\_signal1^{t+1}_{\mathsf{ft}} \to \mathsf{ti}(event_1, t+2) = \langle event \rangle$$

$$stateInf^t_{\mathsf{ft}} \neq S_2 \wedge stateInf^{t+1}_{\mathsf{ft}} = S_2 \wedge sValue^{t+1}_{\mathsf{ft}} = V_3 \wedge sensor\_signal2^{t+1}_{\mathsf{ft}} \wedge \neg sensor\_signal1^{t+1}_{\mathsf{ft}} \to \mathsf{ti}(event_2, t+2) = \langle event \rangle$$

[5] These changes have no contradiction with the initial requirement specification.

To be consistent with the output stream $event_3$ we also need to change the last formula:

$$stateInf_{\mathsf{ft}}^{t+1} = Off \ \wedge\ \mathsf{ti}(signal_2, t+1) \neq \langle\rangle \ \wedge\ \mathsf{ti}(signal_3, t+1) \neq \langle\rangle$$
$$\rightarrow\ \mathsf{ti}(event_3, t+2) = \langle event\rangle$$

After these changes the specification *LogicOut* will be strong causal, but another problem still exists: we argue here about the input values of the input stream *stateInf* within two different time intervals, t and $t+1$. More natural way to represent this situation is to use a local variable to save the value of $stateInf_{\mathsf{ft}}^{t}$ (its initial value must be S_0, because this is the initial system state).

Using this solution we can also simplify the component delay definition (see the specification *LogicOut_2* below).

LogicOut_2 — timed

in
$stateInf : StateType;\ sValue : SValueType;$
$signal_2, signal_3 : Event;$
$sensor\_signal1, sensor\_signal2 : \mathbb{B}ool$

out
$event_1, event_2, event_3 : Event$

local $oldState : StateType;$

init $oldState = S_0;$

asm
$\mathsf{ts}(stateInf) \ \wedge\ \mathsf{ts}(sValue) \ \wedge\ \mathsf{ts}(sensor\_signal1) \ \wedge\ \mathsf{ts}(sensor\_signal2)$
$\mathsf{msg}_1(signal_2) \ \wedge\ \mathsf{msg}_1(signal_3)$

gar

$$oldState' = stateInf_{\mathsf{ft}}^{t}$$

$$oldState \neq S_2 \ \wedge\ stateInf_{\mathsf{ft}}^{t} = S_2 \ \wedge\ sValue_{\mathsf{ft}}^{t} = V_1 \ \wedge\ sensor\_signal2_{\mathsf{ft}}^{t} \ \wedge\ \neg sensor\_signal1_{\mathsf{ft}}^{t} \rightarrow\ \mathsf{ti}(event_3, t+1) = \langle event\rangle$$

$$oldState \neq S_2 \ \wedge\ stateInf_{\mathsf{ft}}^{t} = S_2 \ \wedge\ sValue_{\mathsf{ft}}^{t} = V_2 \ \wedge\ sensor\_signal2_{\mathsf{ft}}^{t} \ \wedge\ \neg sensor\_signal1_{\mathsf{ft}}^{t} \rightarrow\ \mathsf{ti}(event_1, t+1) = \langle event\rangle$$

$$oldState \neq S_2 \ \wedge\ stateInf_{\mathsf{ft}}^{t} = S_2 \ \wedge\ sValue_{\mathsf{ft}}^{t} = V_3 \ \wedge\ sensor\_signal2_{\mathsf{ft}}^{t} \ \wedge\ \neg sensor\_signal1_{\mathsf{ft}}^{t} \rightarrow\ \mathsf{ti}(event_2, t+1) = \langle event\rangle$$

$$stateInf_{\mathsf{ft}}^{t} = S_2 \ \wedge\ \mathsf{ti}(signal_2, t) \neq \langle\rangle \ \wedge\ \mathsf{ti}(signal_3, t) \neq \langle\rangle \rightarrow\ \mathsf{ti}(event_3, t+1) = \langle event\rangle$$

To have more clear definition, for which cases the value of the output stream $event_3$ is specified, we can join the first and the fourth formulas:

$$(oldState \neq S_2 \;\wedge\; stateInf^{t}_{\mathsf{ft}} = S_2 \;\wedge\; sValue^{t}_{\mathsf{ft}} = V_1 \;\wedge$$
$$sensor\_signal2^{t}_{\mathsf{ft}} \;\wedge\; \neg sensor\_signal1^{t}_{\mathsf{ft}}$$
$$\vee$$
$$stateInf^{t}_{\mathsf{ft}} = S_2 \;\wedge\; \mathsf{ti}(signal_2, t) \neq \langle\rangle \;\wedge\; \mathsf{ti}(signal_3, t) \neq \langle\rangle)$$
$$\rightarrow\; \mathsf{ti}(event_3, t+1) = \langle event \rangle$$

This formula can also be reformulated as follows:

$$stateInf^{t}_{\mathsf{ft}} = S_2 \;\wedge$$
$$(oldState \neq S_2 \;\wedge\; sValue^{t}_{\mathsf{ft}} = V_1 \;\wedge\; sensor\_signal2^{t}_{\mathsf{ft}} \;\wedge\; \neg sensor\_signal1^{t}_{\mathsf{ft}}$$
$$\vee$$
$$\mathsf{ti}(signal_2, t) \neq \langle\rangle \;\wedge\; \mathsf{ti}(signal_3, t) \neq \langle\rangle)$$
$$\rightarrow\; \mathsf{ti}(event_3, t+1) = \langle event \rangle$$

We change it to get a new version of the specification, let call it LogicOut_3. In the specification LogicOut_3 we have corrected also the following underspecification: in the specification *LogicOut_2* there is no information that the streams $event_1$, $event_2$ and $event_3$ are disjoint – at every time interval only one of them can contain the *event* message:

$$\mathsf{disj}^{\mathsf{inf}}(event_1, event_2, event_3)$$

We need to add this information, but the formula above is again too abstract, wee need to specify that these streams have empty time intervals in all cases that were underspecified until now. The new version of the specification is presented below. In the next section we discuss a representation of this specification as a timed state transition diagram.

LogicOut_3 — timed

in
$stateInf : StateType;\ sValue : SValueType;$
$signal_2, signal_3 : Event;$
$sensor\_signal1, sensor\_signal2 : \mathbb{B}ool$

out
$event_1, event_2, event_3 : Event$

local $oldState : StateType;$

init $oldState = S_0;$

asm
$\mathsf{ts}(stateInf) \wedge \mathsf{ts}(sValue) \wedge \mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2)$
$\mathsf{msg}_1(signal_2) \wedge \mathsf{msg}_1(signal_3)$

gar

$$oldState' = stateInf^t_{\mathsf{ft}}$$

$$stateInf^t_{\mathsf{ft}} = S_2 \wedge oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_2 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_3, t+1) = \langle\rangle \wedge \mathsf{ti}(event_1, t+1) = \langle event\rangle \wedge \mathsf{ti}(event_2, t+1) = \langle\rangle$$

$$stateInf^t_{\mathsf{ft}} = S_2 \wedge oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_3 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \rightarrow \mathsf{ti}(event_3, t+1) = \langle\rangle \wedge \mathsf{ti}(event_1, t+1) = \langle\rangle \wedge \mathsf{ti}(event_2, t+1) = \langle event\rangle$$

$$stateInf^t_{\mathsf{ft}} = S_2 \wedge (oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_1 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \vee \mathsf{ti}(signal_2, t) \neq \langle\rangle \wedge \mathsf{ti}(signal_3, t) \neq \langle\rangle) \rightarrow \mathsf{ti}(event_3, t+1) = \langle event\rangle \wedge \mathsf{ti}(event_1, t+1) = \langle\rangle \wedge \mathsf{ti}(event_2, t+1) = \langle\rangle$$

Now we can easily see in the specification *LogicOut_3*, that all the output streams are defined only for the case $stateInf^t_{\mathsf{ft}} = S_2$, moreover, with a number of restrictions. Thus we extend the component definition by the following formula and do a number of logical simplifications:

$$\neg(stateInf^t_{\mathsf{ft}} = S_2 \wedge (oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_1 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \vee \mathsf{ti}(signal_2, t) \neq \langle\rangle \wedge \mathsf{ti}(signal_3, t) \neq \langle\rangle))$$
$$\wedge$$
$$\neg(stateInf^t_{\mathsf{ft}} = S_2 \wedge oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_2 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}})$$
$$\wedge$$
$$\neg(stateInf^t_{\mathsf{ft}} = S_2 \wedge oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_3 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}})$$
$$\rightarrow \mathsf{ti}(event_1, t+1) = \langle\rangle \wedge \mathsf{ti}(event_2, t+1) = \langle\rangle \wedge \mathsf{ti}(event_3, t+1) = \langle\rangle$$

This is equal to the following formulas:

$$
\begin{array}{l}
(stateInf^{t}_{\mathsf{ft}} \neq S_2 \;\; \vee \\
(oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_1 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}}) \\
\quad \wedge\; (\mathsf{ti}(signal_2, t) = \langle\rangle \;\vee\; \mathsf{ti}(signal_3, t) = \langle\rangle)) \\
\wedge \\
(stateInf^{t}_{\mathsf{ft}} \neq S_2 \;\; \vee \;\; (oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_2 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}})) \\
\wedge \\
(stateInf^{t}_{\mathsf{ft}} \neq S_2 \;\; \vee \\
\;\; (oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_3 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}})) \\
\rightarrow\; \mathsf{ti}(event_1, t+1) = \langle\rangle \;\wedge\; \mathsf{ti}(event_2, t+1) = \langle\rangle \;\wedge\; \mathsf{ti}(event_3, t+1) = \langle\rangle \\
\equiv \\
stateInf^{t}_{\mathsf{ft}} \neq S_2 \\
\;\vee \\
((oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_1 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}}) \\
\wedge \\
(\mathsf{ti}(signal_2, t) = \langle\rangle \;\vee\; \mathsf{ti}(signal_3, t) = \langle\rangle) \\
\wedge \\
(oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_2 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}}) \\
\wedge \\
(oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_3 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}})) \\
\rightarrow\; \mathsf{ti}(event_1, t+1) = \langle\rangle \;\wedge\; \mathsf{ti}(event_2, t+1) = \langle\rangle \;\wedge\; \mathsf{ti}(event_3, t+1) = \langle\rangle \\
\equiv \\
stateInf^{t}_{\mathsf{ft}} \neq S_2 \\
\;\vee\; ((oldState = S_2 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}}) \\
\qquad \wedge\; (\mathsf{ti}(signal_2, t) = \langle\rangle \;\vee\; \mathsf{ti}(signal_3, t) = \langle\rangle)) \\
\rightarrow \mathsf{ti}(event_1, t+1) = \langle\rangle \;\wedge\; \mathsf{ti}(event_2, t+1) = \langle\rangle \;\wedge\; \mathsf{ti}(event_3, t+1) = \langle\rangle
\end{array}
$$

Please note that the stream *sValue* is time-synchronous and its type contains only three values: V_1, V_2 and V_3, therefore we can simplify the expression

$$
\begin{array}{l}
(oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_1 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}}) \\
\wedge \\
(oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_2 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}}) \\
\wedge \\
(oldState = S_2 \;\vee\; sValue^{t}_{\mathsf{ft}} \neq V_3 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}})
\end{array}
$$

to the expression

$$
oldState = S_2 \;\vee\; \neg sensor\_signal2^{t}_{\mathsf{ft}} \;\vee\; sensor\_signal1^{t}_{\mathsf{ft}}
$$

It easy to see that to find out all the presented inconsistencies and underspecifications within a large specification like *Logic*, is much more difficult than after the decomposition.

As result we get the following specification of the component *LogicOut*.

LogicOut — timed

in
$stateInf : StateType;\ sValue : SValueType;$
$signal_2, signal_3 : Event;$
$sensor\_signal1, sensor\_signal2 : \mathbb{B}ool$

out
$event_1, event_2, event_3 : Event$

local $oldState : StateType;$

init $oldState = S_0;$

asm
$\mathsf{ts}(stateInf) \wedge \mathsf{ts}(sValue) \wedge \mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2)$
$\mathsf{msg}_1(signal_2) \wedge \mathsf{msg}_1(signal_3)$

gar

$$oldState' = stateInf^t_{\mathsf{ft}}$$

$$\begin{array}{l} stateInf^t_{\mathsf{ft}} = S_2 \wedge oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_2 \wedge \\ sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \\ \quad \to \mathsf{ti}(event_1, t+1) = \langle event \rangle \wedge \\ \qquad \mathsf{ti}(event_2, t+1) = \langle\rangle \wedge \mathsf{ti}(event_3, t+1) = \langle\rangle \end{array}$$

$$\begin{array}{l} stateInf^t_{\mathsf{ft}} = S_2 \wedge oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_3 \wedge \\ sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \\ \quad \to \mathsf{ti}(event_1, t+1) = \langle\rangle \wedge \\ \qquad \mathsf{ti}(event_2, t+1) = \langle event \rangle \wedge \mathsf{ti}(event_3, t+1) = \langle\rangle \end{array}$$

$$\begin{array}{l} stateInf^t_{\mathsf{ft}} = S_2 \wedge \\ (oldState \neq S_2 \wedge sValue^t_{\mathsf{ft}} = V_1 \wedge \\ sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \\ \vee \\ \mathsf{ti}(signal_2, t) \neq \langle\rangle \wedge \mathsf{ti}(signal_3, t) \neq \langle\rangle) \\ \quad \to \mathsf{ti}(event_1, t+1) = \langle\rangle \wedge \\ \qquad \mathsf{ti}(event_2, t+1) = \langle\rangle \wedge \mathsf{ti}(event_3, t+1) = \langle event \rangle \end{array}$$

$$\begin{array}{l} stateInf^t_{\mathsf{ft}} \neq S_2 \\ \vee \\ (\ (\mathsf{ti}(signal_2, t) = \langle\rangle \vee \mathsf{ti}(signal_3, t) = \langle\rangle) \\ \wedge\ (oldState = S_2 \vee \neg sensor\_signal2^t_{\mathsf{ft}} \vee sensor\_signal1^t_{\mathsf{ft}})) \\ \qquad \to \mathsf{ti}(event_1, t+1) = \langle\rangle \wedge \mathsf{ti}(event_2, t+1) = \langle\rangle \wedge \mathsf{ti}(event_3, t+1) = \langle\rangle \end{array}$$

6.1.10 LogicOut Component: Timed State Transition Diagram

The specification *LogicOut* is semantically equal to the specification using a simple state transition diagram (see Figure 3), which two states, $NonS2$ and $S2$, according to the value of the local variable *oldState*. We take $NonS2$ as the initial state, because of to the initial value of this variable.

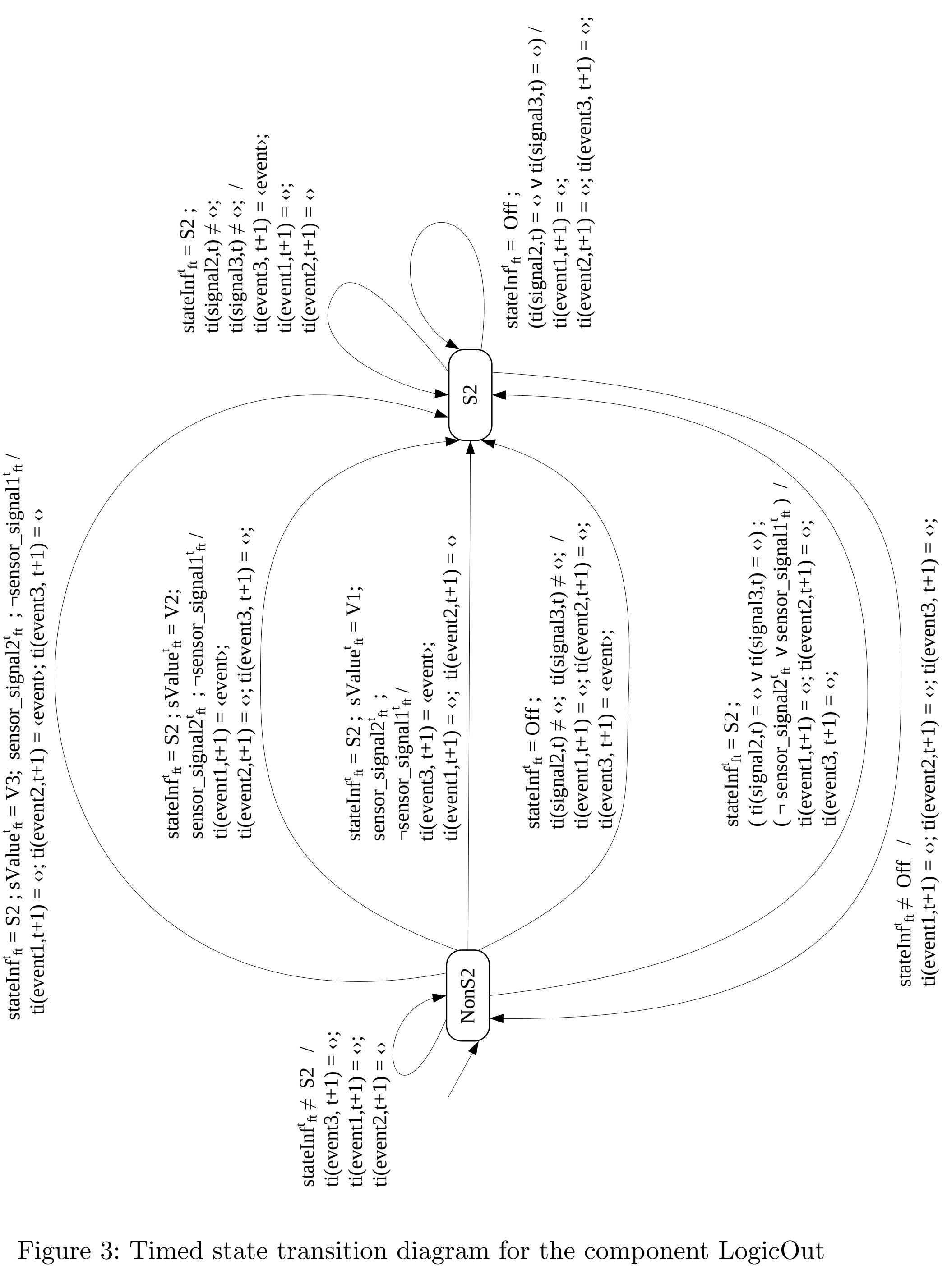


Figure 3: Timed state transition diagram for the component LogicOut

6.1.11 LogicMain Component

The specification *LogicMain_2* was obtained in Section 6.1.8 by decomposition of the specification *LogicNew*. To make it more readable we moved some formulas inside it to group them by value of the variable *SystemState* at the time interval t and to put them in the order which can be realistic in a state transition diagram – the intermediate result is presented by the specification *LogicMain_3*.

This specification is now much readable than the *Logic* specification – some inconsistencies and undefined cases can be found. Let discuss all the formulas that describe system behavior grouping them to the system states.

Now we can see, that the local variable *targetValue* is indeed in strong relation with the system state, but comparing, e.g., the formulas 11 – 15 with the 35th formula, we find out more possibilities and also benefits to try to move the computation of this variable to separate formulas. This separation also allows us to present the timed state transition diagram of the *LogicMain* component (see Section 6.2) in a simplified way, omitting the local variables calculation – this representation is more readable for the case one want to understand the main state transitions.

The result of the splitting of formulas is presented by the specification, where the formulas to split were: 4, 11, 12, 14, 15, 16 – 22, 26 – 33. After that we group all the formulas about the local variable *targetValue* after the main formulas about state changes. We also can see that

- the formulas 16 – 22 describe the case in which the system is at the state S_4 and will stay at this state at the next time unit, thus we can group these formulas together;
- the formulas 26 – 28 describe the case in which the system is in the state S_5 and will move to the state S_4 at this state at the next time unit, thus we can group them together;
- the formulas 30 – 32 describe the case in which the system is in the state S_6 and will move to the state S_4 at this state at the next time unit, thus we can group them together;

The result of the optimization is presented by the specification *LogicMain_4*.

The result of the optimization according to Sections 6.1.12 – 6.1.19 is presented by the specification *LogicMain*.

LogicMain_3(const LogicParam) — timed

in $sensor\_signal1, sensor\_signal2, sensor\_signal3 : \mathbb{B}\mathrm{ool};\ signal_1 : SignalType$
$current\_value, counter1, counter2 : \mathbb{N};\ pre_1, pre_2, pre_3, pre_4, pre_5, power1, power2 : Event$

out $target\_value\_1, target\_value\_2 : \mathbb{N};\ stateInf, stateInfOut : StateType$

local $SystemState : StateType;\ targetValue : \mathbb{N}$

init $SystemState = S_0;\ targetValue = 0;$

asm $\mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2) \wedge \mathsf{ts}(sensor\_signal3)$
$\mathsf{msg}_1(signal_1) \wedge \mathsf{ts}(current\_value) \wedge \mathsf{ts}(counter1) \wedge \mathsf{ts}(counter2) \wedge \mathsf{msg}_1(power1) \wedge \mathsf{msg}_1(power2)$
$\mathsf{ts}(pre_1) \wedge \mathsf{ts}(pre_2) \wedge \mathsf{ts}(pre_3) \wedge \mathsf{ts}(pre_4) \wedge \mathsf{ts}(pre_5)$

gar
1 $stateInfOut = stateInf \wedge target\_value\_2 = target\_value\_1$

$\forall\, t \in \mathbb{N}:$
2 $\mathsf{ti}(stateInf, t) = \langle SystemState \rangle \wedge \mathsf{ti}(target\_value\_1, t) = \langle targetValue \rangle$

3 $\mathsf{ti}(power1, t) = \langle\rangle \rightarrow CrCtSate' = S_0$

4 $SystemState = S_0 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \rightarrow CrCtSate' = S_1$
4t $SystemState = S_0 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \rightarrow targetValue' = 0$

5 $SystemState = S_1 \wedge \mathsf{ft.ti}(pre_1, t+1) \wedge \mathsf{ft.ti}(pre_2, t+1) \wedge \mathsf{ft.ti}(pre_3, t+1) \wedge \mathsf{ft.ti}(pre_4, t+1) \wedge \mathsf{ft.ti}(pre_5, t+1) \rightarrow SystemState' = S_2$
6 $SystemState = S_1 \wedge (\neg\, \mathsf{ft.ti}(pre_1, t+1) \vee \neg\, \mathsf{ft.ti}(pre_2, t+1) \vee \neg\, \mathsf{ft.ti}(pre_3, t+1) \vee \neg\, \mathsf{ft.ti}(pre_4, t+1) \vee \neg\, \mathsf{ft.ti}(pre_5, t+1)) \rightarrow SystemState' = S_1$

7 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge \neg\, Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_3$
8 $SystemState = S_2 \wedge (\neg sensor\_signal_{12} \wedge Signal1Precondition(\mathsf{ti}(signal_1, t))) \rightarrow SystemState' = S_2$

9 $(SystemStateSubset(SystemState) \vee SystemState = S_2) \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{true} \rangle \rightarrow SystemState' = S_7$

10 $SystemState \neq S_2 \wedge \neg \mathsf{ti}(power1, t) = \langle\rangle \wedge \mathsf{ti}(power2, t) = \langle\rangle \rightarrow SystemState' = S_1$

11 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow SystemState' = S_4$
11t $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow targetValue' = limTargetValue$
12 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue > 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4 \rangle \rightarrow SystemState' = S_4$
12t $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue > 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4 \rangle \rightarrow targetValue' = targetValue$
13 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge targetValue = 0 \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_4 \rangle \rightarrow SystemState' = S_3$
14 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \rightarrow SystemState' = S_5$
14t $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \rightarrow targetValue' = limTargetValue$
15 $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \rightarrow SystemState' = S_6$
15t $SystemState = S_3 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \rightarrow targetValue' = limTargetValue$

16 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow SystemState' = S_4$
16t $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_3 \rangle \rightarrow targetValue' = limTargetValue$

17 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}}) \rightarrow SystemState' = S_4$
17t $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow targetValue' = ChangeTargetValue(targetValue, SignalA_5)$

18 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl \rightarrow SystemState' = S_4$
18t $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl \rightarrow targetValue' = limTargetValue$

19 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge \mathsf{ti}(counter2, t) > 0 \rightarrow SystemState' = S_4$
19t $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_5 \rangle \wedge \mathsf{ti}(counter2, t) > 0 \rightarrow targetValue' = limTargetValue$

20 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow SystemState' = S_4$
20t $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow targetValue' = ChangeTargetValue(targetValue, SignalA_6)$

21 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl \rightarrow SystemState' = S_4$
21t $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl \rightarrow targetValue' = limTargetValue$

22 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge \mathsf{ti}(counter1, t) > 0 \rightarrow SystemState' = S_4$
22t $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_6 \rangle \wedge \mathsf{ti}(counter1, t) > 0 \rightarrow targetValue' = limTargetValue$

23 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \rightarrow SystemState' = S_5$
24 $SystemState = S_4 \wedge \neg sensor\_signal_{12} \wedge \mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \rightarrow SystemState' = S_6$
25 $(SystemState = S_4 \vee SystemState = S_5) \wedge \mathsf{ti}(signal_1, t) = \langle\rangle \rightarrow SystemState' = S_6$

26 $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} > targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle \rightarrow SystemState' = S_4$
26t $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} > targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle \rightarrow targetValue' = limTargetValue$

27 $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \leq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle \rightarrow SystemState' = S_4$
27t $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \leq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle \rightarrow targetValue' \neq 0$

28 $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue) \rightarrow SystemState' = S_4$
28t $SystemState = S_5 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue) \rightarrow targetValue' = min(MaxCurrentValue, MaxTargetValue)$

29 $SystemState = S_5 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}} \rightarrow SystemState' = S_2$
29t $SystemState = S_5 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}} \rightarrow targetValue' = targetValue$

30 $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} < targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \rightarrow SystemState' = S_4$
30t $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} < targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \rightarrow targetValue' = limTargetValue$

31 $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \geq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \rightarrow SystemState' = S_4$
31t $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \geq targetValue \wedge \mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle \rightarrow targetValue' \neq 0$

32 $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue) \rightarrow SystemState' = S_4$
32t $SystemState = S_6 \wedge \neg\, sensor\_signal_{12} \wedge current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue) \rightarrow targetValue' = max(MinCurrentValue, MinTargetValue)$

33 $SystemState = S_6 \wedge sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}} \wedge \neg sensor\_signal3^t_{\mathsf{ft}} \rightarrow SystemState' = S_2$

34 $SystemState = S_7 \wedge \mathsf{ti}(power1, t) = \langle\rangle \rightarrow SystemState' = S_7$

35 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal3, t) = \langle \mathsf{true} \rangle \rightarrow targetValue' = 0$

36 $SystemStateSubset(SystemState) \wedge \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true} \rangle \wedge \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle \rightarrow SystemState' = S_2$

where $sensor\_signal_{12}, limTargetValue$ so that
$sensor\_signal_{12} = sensor\_signal2^t_{\mathsf{ft}} \vee sensor\_signal1^t_{\mathsf{ft}}$
$limTargetValue = LimitedValue(current\_value^t_{\mathsf{ft}}, MinCurrentValue, MinTargetValue, MaxCurrentValue, MaxTargetValue)$

LogicMain_4(const LogicParam) — timed

in $sensor\_signal1, sensor\_signal2, sensor\_signal3 : \mathbb{B}\text{ool};\ signal_1 : SignalType$
$current\_value, counter1, counter2 : \mathbb{N};\ pre_1, pre_2, pre_3, pre_4, pre_5, power1, power2 : Event$

out $target\_value\_1, target\_value\_2 : \mathbb{N};\ \ stateInf, stateInfOut : StateType$

local $SystemState : StateType;\ targetValue : \mathbb{N}$

init $SystemState = S_0;\ targetValue = 0;$

asm $\mathsf{ts}(sensor\_signal1)\ \wedge\ \mathsf{ts}(sensor\_signal2)\ \wedge\ \mathsf{ts}(sensor\_signal3)$
$\mathsf{msg}_1(signal_1)\ \wedge\ \mathsf{ts}(current\_value)\ \wedge\ \mathsf{ts}(counter1)\ \wedge\ \mathsf{ts}(counter2)\ \wedge\ \mathsf{msg}_1(power1)\ \wedge\ \mathsf{msg}_1(power2)$
$\mathsf{ts}(pre_1)\ \wedge\ \mathsf{ts}(pre_2)\ \wedge\ \mathsf{ts}(pre_3)\ \wedge\ \mathsf{ts}(pre_4)\ \wedge\ \mathsf{ts}(pre_5)$

gar

1 $stateInfOut = stateInf\ \wedge\ target\_value\_2 = target\_value\_1$

$\forall\, t \in \mathbb{N}:$

2 $\mathsf{ti}(stateInf, t) = \langle SystemState\rangle\ \wedge\ \mathsf{ti}(target\_value\_1, t) = \langle targetValue\rangle$

3 $\mathsf{ti}(power1, t) = \langle\rangle\ \rightarrow\ CrCtState' = S_0$

4 $SystemState = S_0\ \wedge\ \mathsf{ti}(power1, t) \neq \langle\rangle\ \rightarrow\ CrCtState' = S_1$

5 $SystemState = S_1\ \wedge\ \mathsf{ft.ti}(pre_1, t+1)\ \wedge\ \mathsf{ft.ti}(pre_2, t+1)\ \wedge\ \mathsf{ft.ti}(pre_3, t+1)\ \wedge\ \mathsf{ft.ti}(pre_4, t+1)\ \wedge\ \mathsf{ft.ti}(pre_5, t+1)\ \rightarrow\ SystemState' = S_2$
6 $SystemState = S_1\ \wedge\ (\neg\ \mathsf{ft.ti}(pre_1, t+1)\ \vee\ \neg\ \mathsf{ft.ti}(pre_2, t+1)\ \vee\ \neg\ \mathsf{ft.ti}(pre_3, t+1)\ \vee\ \neg\ \mathsf{ft.ti}(pre_4, t+1)\ \vee\ \neg\ \mathsf{ft.ti}(pre_5, t+1))\ \rightarrow\ SystemState' = S_1$

7 $SystemState = S_2\ \wedge\ (\neg sensor\_signal_{12}\ \wedge\ \neg\ Signal1Precondition(\mathsf{ti}(signal_1, t)))\ \rightarrow\ SystemState' = S_3$
8 $SystemState = S_2\ \wedge\ (\neg sensor\_signal_{12}\ \wedge\ Signal1Precondition(\mathsf{ti}(signal_1, t)))\ \rightarrow\ SystemState' = S_2$

9 $(SystemStateSubset(SystemState)\ \vee\ SystemState = S_2)\ \wedge\ \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{true}\rangle\ \rightarrow\ SystemState' = S_7$

10 $SystemState \neq S_2\ \wedge\ \neg\mathsf{ti}(power1, t) = \langle\rangle\ \wedge\ \mathsf{ti}(power2, t) = \langle\rangle\ \rightarrow SystemState' = S_1$

11 $SystemState = S_3\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle\ \rightarrow\ SystemState' = S_4$
12 $SystemState = S_3\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ targetValue > 0\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle\ \rightarrow\ SystemState' = S_4$
13 $SystemState = S_3\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ targetValue = 0\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle\ \rightarrow\ SystemState' = S_3$
14 $SystemState = S_3\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle\ \rightarrow\ SystemState' = S_5$
15 $SystemState = S_3\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle \rightarrow\ SystemState' = S_6$

16 $SystemState = S_4\ \wedge\ \neg sensor\_signal_{12}\ \wedge$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle\ \wedge\ SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}}))\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle\ \wedge\ ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl)\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle\ \wedge\ \mathsf{ti}(counter2, t) > 0)\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle\ \wedge\ SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}}))\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle\ \wedge\ ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl))\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle\ \wedge\ \mathsf{ti}(counter1, t) > 0))$
$\rightarrow\ SystemState' = S_4$

17 $SystemState = S_4\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle\ \rightarrow\ SystemState' = S_5$
18 $SystemState = S_4\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle\ \rightarrow\ SystemState' = S_6$
19 $(SystemState = S_4 \vee SystemState = S_5)\ \wedge\ \mathsf{ti}(signal_1, t) = \langle\rangle\ \rightarrow\ SystemState' = S_6$

20 $SystemState = S_5\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge$
$((current\_value^t_{\mathsf{ft}} > targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle)\ \vee$
$(current\_value^t_{\mathsf{ft}} \leq targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle)\ \vee$
$current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue))$
$\rightarrow\ SystemState' = S_4$

21 $SystemState = S_5\ \wedge\ sensor\_signal2^t_{\mathsf{ft}}\ \wedge\ \neg sensor\_signal1^t_{\mathsf{ft}}\ \wedge\ \neg sensor\_signal3^t_{\mathsf{ft}}\ \rightarrow\ SystemState' = S_2$

22 $SystemState = S_6\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge$
$((current\_value^t_{\mathsf{ft}} < targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle)\ \vee$
$(current\_value^t_{\mathsf{ft}} \geq targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle)\ \vee$
$current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue))$
$\rightarrow\ SystemState' = S_4$

23 $SystemState = S_6\ \wedge\ sensor\_signal2^t_{\mathsf{ft}}\ \wedge\ \neg sensor\_signal1^t_{\mathsf{ft}}\ \wedge\ \neg sensor\_signal3^t_{\mathsf{ft}}\ \rightarrow\ SystemState' = S_2$

24 $SystemState = S_7\ \wedge\ \mathsf{ti}(power1, t) = \langle\rangle\ \rightarrow\ SystemState' = S_7$

25 $SystemStateSubset(SystemState)\ \wedge\ \mathsf{ti}(sensor\_signal2, t) = \langle\mathsf{true}\rangle\ \wedge\ \mathsf{ti}(sensor\_signal1, t) = \langle\mathsf{false}\rangle\ \rightarrow\ SystemState' = S_2$

26 $SystemState = S_0\ \wedge\ \mathsf{ti}(power1, t) \neq \langle\rangle\ \rightarrow\ targetValue' = 0$

27 $SystemStateSubset(SystemState)\ \wedge\ \mathsf{ti}(sensor\_signal3, t) = \langle\mathsf{true}\rangle\ \rightarrow\ targetValue' = 0$

28 $SystemState = S_3\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ (\mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle \vee \mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle \vee \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle)\ \rightarrow\ targetValue' = limTargetValue$
29 $SystemState = S_3\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ targetValue > 0\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle\ \rightarrow\ targetValue' = targetValue$

30 $SystemState = S_4\ \wedge\ \neg sensor\_signal_{12}\ \wedge$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle\ \wedge\ ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl)\ \vee\ (\mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle\ \wedge\ \mathsf{ti}(counter2, t) > 0)\ \vee$
$(\mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle\ \wedge\ ModSubtraction(current\_value^t_{\mathsf{ft}}, targetValue) > X\_Appl)\ \vee\ (\mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle\ \wedge\ \mathsf{ti}(counter1, t) > 0))$
$\rightarrow\ targetValue' = limTargetValue$

31 $SystemState = S_4\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_5\rangle\ \wedge\ SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow\ targetValue' = ChangeTargetValue(targetValue, SignalA_5)$

32 $SystemState = S_4\ \wedge\ \neg sensor\_signal_{12}\ \wedge\ \mathsf{ti}(signal_1, t) = \langle SignalA_6\rangle\ \wedge\ SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, targetValue, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow\ targetValue' = ChangeTargetValue(targetValue, SignalA_6)$

33 $SystemState = S_5\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge\ current\_value^t_{\mathsf{ft}} > targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle\ \rightarrow\ targetValue' = limTargetValue$
34 $SystemState = S_5\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge\ current\_value^t_{\mathsf{ft}} \leq targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_7\rangle\ \rightarrow\ targetValue' \neq 0$
35 $SystemState = S_5\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge\ current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue)\ \rightarrow\ targetValue' = min(MaxCurrentValue, MaxTargetValue)$
36 $(SystemState = S_5\ \vee\ SystemState = S_6)\ \wedge\ sensor\_signal2^t_{\mathsf{ft}}\ \wedge\ \neg sensor\_signal1^t_{\mathsf{ft}}\ \wedge\ \neg sensor\_signal3^t_{\mathsf{ft}}\ \rightarrow\ targetValue' = targetValue$
37 $SystemState = S_6\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge\ current\_value^t_{\mathsf{ft}} < targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle\ \rightarrow\ targetValue' = limTargetValue$
38 $SystemState = S_6\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge\ current\_value^t_{\mathsf{ft}} \geq targetValue\ \wedge\ \mathsf{ti}(signal_1, t) \neq \langle SignalA_8\rangle\ \rightarrow\ targetValue' \neq 0$
39 $SystemState = S_6\ \wedge\ \neg\ sensor\_signal_{12}\ \wedge\ current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue)\ \rightarrow\ targetValue' = max(MinCurrentValue, MinTargetValue)$

where $sensor\_signal_{12}, limTargetValue$ so that
$sensor\_signal_{12} = sensor\_signal2^t_{\mathsf{ft}} \vee sensor\_signal1^t_{\mathsf{ft}}$
$limTargetValue = LimitedValue(current\_value^t_{\mathsf{ft}}, MinCurrentValue, MinTargetValue, MaxCurrentValue, MaxTargetValue)$

6.1.12 State S_0

The explicit description of the system behavior at this state is given only in the 4th formula, where the reaction on the nonempty signal $power1$ is presented. This nonempty signal is the only way to change the system state from S_0 to S_1. The system behavior at the state S_0 for the case of the empty signal $power1$ is given by the 3rd formula: if at any state the signal $power1$ is empty, the system goes to the state S_0, thus, if the system was at the state S_0 at this time unit, it does not change its state.

Value of the local variable $targetValue$ at the time interval $t+1$ for the case the system is at the state S_0 at the time interval t is defined only for the nonempty signal $power1$ (formula 26), but this underspecification has no influence on the system behavior – value of this variable at the state S_0 is unimportant and moving to the state S_1 the system will set the value to 0 (according to the formula 26).

6.1.13 State S_1

The description of the system behavior at this state is given by 5th and 6th formulas: if all the streams $pre_1, \dots, pre_5$ have at the time unit t true-values, the system state will be changed to S_2, otherwise the system state will be unchanged.

This is a contradiction to the 3rd formula: if the system is on the state S_1 and the signal $power1$ is empty, then according to the 3rd formula the system must change its state to S_0, but according to the 5th and 6th formulas the system state at the next time unit must be either S_1 or S_2. Therefore, we need to extend the 5th and 6th formulas to correct this underspecification: their must hold only for the case $\mathsf{ti}(power1, t) \neq \langle\rangle$.

$$
\begin{array}{l}
SystemState = S_1 \;\wedge\; \mathsf{ti}(power1, t) \neq \langle\rangle \;\wedge \\
\mathsf{ft.ti}(pre_1, t+1) \;\wedge\; \mathsf{ft.ti}(pre_2, t+1) \;\wedge\; \mathsf{ft.ti}(pre_3, t+1) \;\wedge \\
\mathsf{ft.ti}(pre_4, t+1) \;\wedge\; \mathsf{ft.ti}(pre_5, t+1) \\
\rightarrow\; SystemState' = S_2
\end{array}
$$

$$
\begin{array}{l}
SystemState = S_1 \;\wedge\; \mathsf{ti}(power1, t) \neq \langle\rangle \;\wedge \\
(\neg\, \mathsf{ft.ti}(pre_1, t+1) \;\vee\; \neg\, \mathsf{ft.ti}(pre_2, t+1) \;\vee\; \neg\, \mathsf{ft.ti}(pre_3, t+1) \;\vee \\
\neg\, \mathsf{ft.ti}(pre_4, t+1) \;\vee\; \neg\, \mathsf{ft.ti}(pre_5, t+1)) \\
\rightarrow\; SystemState' = S_1
\end{array}
$$

But also after this correction we still have contradictions, because the behavior at the system state S_1 is also implicit described by the 10th formula that says: if the system does not be at the state S_2, the signal $power1$ does not be empty, but the signal $power2$ is empty, then the system state at the next time unit must be changed to S_1. After analyzing the system we restrict the 10th formula to hold only at the states $S_3, \dots, S_6$, i.e. only at the states for which the predicate $SystemStateSubset$ holds and unify the syntax:

$$\mathit{SystemStateSubset}(\mathit{SystemState}) \ \wedge \ \mathsf{ti}(power1, t) \neq \langle\rangle \ \wedge \ \mathsf{ti}(power2, t) = \langle\rangle$$
$$\rightarrow \mathit{SystemState}' = S_1$$

Value of the local variable *targetValue* at the time interval $t+1$ is undefined for the case the system is at the state S_1 at the time interval t, thus, analyzing the system we add new formula $(*)$ that specify this value to be unchanged for both possible situations: system state will be changes to S_2 or will be unchanged.

$$\mathit{SystemState} = S_1 \ \rightarrow \ \mathit{targetValue}' = \mathit{targetValue} \qquad (*)$$

We can also see that the local variable *targetValue* is undefined for the case described by the corrected 10th formula. We specify explicitly that the value of this variable must be unchanged for this case:

$$\mathit{SystemStateSubset}(\mathit{SystemState}) \ \wedge \ \mathsf{ti}(power1, t) \neq \langle\rangle \ \wedge \ \mathsf{ti}(power2, t) = \langle\rangle$$
$$\rightarrow \mathit{targetValue}' = \mathit{targetValue}$$

We join this formula with $(*)$:

$$(\mathit{SystemState} = S_1 \ \vee$$
$$(\mathit{SystemStateSubset}(\mathit{SystemState}) \ \wedge \ \mathsf{ti}(power1, t) \neq \langle\rangle \ \wedge \ \mathsf{ti}(power2, t) = \langle\rangle))$$
$$\rightarrow \ \mathit{targetValue}' = \mathit{targetValue} \qquad (*')$$

6.1.14 State S_2

The description of the system behavior at the state S_2 is given by 7th, 8th and 9th formulas, and analyzing them we find an underspecification: they do not cover the case $\mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true} \rangle$, the 9th formula hold only if $\mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{true} \rangle$, where 7th and 8th formulas hold only for the case $\mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle \wedge \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{false} \rangle$, because

$$\neg sensor\_signal_{12} =$$
$$\neg(sensor\_signal1^t_{\mathsf{ft}} \ \vee \ sensor\_signal2^t_{\mathsf{ft}}) =$$
$$\neg sensor\_signal1^t_{\mathsf{ft}} \ \wedge \ \neg sensor\_signal2^t_{\mathsf{ft}} =$$
$$\mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle \ \wedge \ \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{false} \rangle$$

Analyzing the system we find the underspecified formula $(**)$:

$$\mathit{SystemState} = S_2 \ \wedge \ \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true} \rangle \ \wedge \ \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle$$
$$\rightarrow \ \mathit{SystemState}' = S_2$$

This formula is very similar to the 36th formula, thus, we extend the 25th formula by $(**)$:

$$(\mathit{SystemStateSubset}(\mathit{SystemState}) \ \vee \ \mathit{SystemState} = S_2)$$
$$\wedge \ \mathsf{ti}(sensor\_signal2, t) = \langle \mathsf{true} \rangle \ \wedge \ \mathsf{ti}(sensor\_signal1, t) = \langle \mathsf{false} \rangle$$
$$\rightarrow \ \mathit{SystemState}' = S_2$$

Value of the local variable $targetValue$ at the time interval $t+1$ is undefined for the case the system is at the state S_2 at the time interval t, thus, analyzing the system we need new formula that specify this value to be unchanged for the tree possible situations: system state will be changes to S_3 or S_7, or system state will be unchanged.

$$SystemState = S_2 \;\rightarrow\; targetValue' = targetValue$$

We join this formula with the formula $(*')$ defined in the previous section:

$$\begin{array}{l}(SystemState = S_1 \;\vee\; SystemState = S_2 \;\vee \\ (SystemStateSubset(SystemState) \;\wedge\; \mathsf{ti}(power1, t) \neq \langle\rangle \;\wedge\; \mathsf{ti}(power2, t) = \langle\rangle)) \\ \rightarrow\; targetValue' = targetValue \end{array} \qquad (*'')$$

Moreover, all these formulas have a contradiction to the 3rd formula: if the system is on the state S_2 and the signal $power1$ is empty, then according to the 3rd formula the system must change its state to S_0. Therefore, we need to extend these formulas to correct this underspecification: their must hold only for the case $\mathsf{ti}(power1, t) \neq \langle\rangle$.

6.1.15 State S_3

The description of the system behavior at this state is given by the 9th – 15th and the 25th formulas. It is easy to see the following underspecification at the formulas 11 – 15: they must be extended by the conjunct

$$\mathsf{ti}(power1, t) \neq \langle\rangle \;\wedge\; \mathsf{ti}(power2, t) \neq \langle\rangle$$

To make the correction result more readable, we define a new abbreviation *power_sensor_signal* as follows

$$power\_sensor\_signal = \neg sensor\_signal_{12} \;\wedge\; \mathsf{ti}(power1, t) \neq \langle\rangle \;\wedge\; \mathsf{ti}(power2, t) \neq \langle\rangle$$

and use it in these formulas instead of $sensor\_signal_{12}$. Therefore, we get:

$$\begin{array}{l}SystemState = S_3 \;\wedge\; power\_sensor\_signal \;\wedge\; \mathsf{ti}(signal_1, t) = \langle SignalA_3\rangle \\ \rightarrow\; SystemState' = S_4\end{array}$$

$$\begin{array}{l}SystemState = S_3 \;\wedge\; power\_sensor\_signal \;\wedge\; targetValue > 0 \;\wedge\; \mathsf{ti}(signal_1, t) = \langle SignalA_4\rangle \\ \rightarrow\; SystemState' = S_4\end{array}$$

$$\begin{array}{l}SystemState = S_3 \;\wedge\; power\_sensor\_signal \;\wedge\; \mathsf{ti}(sensor\_signal3, t) = \langle \mathsf{false}\rangle \;\wedge \\ \mathsf{ti}(signal_1, t) = \langle SignalA_7\rangle \\ \rightarrow\; SystemState' = S_5\end{array}$$

$$\begin{array}{l}SystemState = S_3 \;\wedge\; power\_sensor\_signal \;\wedge\; \mathsf{ti}(sensor\_signal3, t) = \langle \mathsf{false}\rangle \;\wedge \\ \mathsf{ti}(signal_1, t) = \langle SignalA_8\rangle \\ \rightarrow\; SystemState' = S_6\end{array}$$

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ \mathit{targetValue} = 0 \ \wedge \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle \\ \rightarrow \ \mathit{SystemState}' = S_3 \end{array}$$

Now we can also see that these formulas does not cover the case

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_3 \rangle \ \wedge \ \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_4 \rangle \ \wedge \\ \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_7 \rangle \ \wedge \ \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_8 \rangle \end{array}$$

or, more explicit,

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_1 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_2 \rangle \ \vee \\ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \rangle) \end{array}$$

The cases $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_1 \rangle$ and $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_2 \rangle$ can be omitted, because if we analyse the whole system, we get that this cases imply also $\mathit{sensor\_signal}_{12}$, thus these cases will be covered either by the 9th or by the 25st formula. Thus, we need to add a new formula to the specification:

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \rangle \\ \rightarrow \ \mathit{SystemState}' = S_3 \end{array}$$

The same case must be added to specify value of the local variable *targetValue* (let call this formula $(***)$):

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \rangle \\ \rightarrow \ \mathit{targetValue}' = \mathit{targetValue} \end{array}$$

The values of this variable are defined now by this formula as well as by the formulas 27 – 29, and we can easily see that the formulas $(***)$, 28 and 29 are in contradiction to the 27th formula. To correct this, we need to add new conjunct $\mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle$ to the formulas $(***)$, 28 and 29:

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \rangle) \\ \rightarrow \ \mathit{targetValue}' = \mathit{targetValue} \end{array}$$

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_8 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_3 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_7 \rangle) \\ \rightarrow \ \mathit{targetValue}' = \mathit{limTargetValue} \end{array}$$

$$\begin{array}{l} \mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \ \wedge \\ \mathit{targetValue} > 0 \ \wedge \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle \\ \rightarrow \ \mathit{targetValue}' = \mathit{targetValue} \end{array}$$

To get more readable specification we join now the formulas $(***)$ and 28:

$$\begin{aligned}&\mathit{SystemState} = S_3 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \ \mathsf{ti}(\mathit{sensor\_signal}3, t) = \langle \mathsf{false} \rangle \ \wedge \\ &((\mathit{targetValue} > 0 \ \wedge \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle) \ \vee \\ &\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \rangle)) \\ &\rightarrow \ \mathit{targetValue}' = \mathit{targetValue}\end{aligned}$$

6.1.16 State S_4

The description of the system behavior at this state is given by 9th, 10th, 16th – 19th and 25th formulas. In the formulas 16 – 18 we need the same changes as described in Section 6.1.15: replace $\neg \mathit{sensor\_signal}_{12}$ by $\mathit{power\_sensor\_signal}$. We also need to add this conjunct to the 19th formula.

We can easily see that in the specification there is no formula describing the following cases: $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_1 \rangle$, $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_2 \rangle$, and $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle$. We add the case $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle$ to the 16th formula, because the system state must be unchanged in this situations. The cases $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_1 \rangle$ and $\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_2 \rangle$ can be omitted, because if we analyse the whole system, we get that this cases imply also $\mathit{sensor\_signal}_{12}$, thus these cases will be covered either by the 9th or by the 25st formula.

Analyzing the specified reactions to the signal $\mathit{SignalA}_5$ as well as to the signal $\mathit{SignalA}_6$ for the case that no switch-off-condition occurs and compare these definitions with the system behavior, we can find out that according to these signal the system state will be unchanged independently from the conjuncts
$\mathit{SignalAccepted}(\mathsf{true}, \mathit{current\_value}^t_{\mathsf{ft}}, \mathit{targetValue}, \mathit{counter}1^t_{\mathsf{ft}}, \mathit{counter}2^t_{\mathsf{ft}})$,
$\mathit{ModSubtraction}(\mathit{current\_value}^t_{\mathsf{ft}}, \mathit{targetValue}) > X\_Appl$,
$\mathsf{ti}(\mathit{counter}1, t) > 0$ and $\mathsf{ti}(\mathit{counter}2, t) > 0$. All these conjunct influence only on the value of the local variable *targetValue* and we can simplify the 16th formula as follows:

$$\begin{aligned}&\mathit{SystemState} = S_4 \ \wedge \ \mathit{power\_sensor\_signal} \ \wedge \\ &(\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_3 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle \ \vee \\ &\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \ \vee \ \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle) \\ &\rightarrow \ \mathit{SystemState}' = S_4\end{aligned}$$

In the formulas 30 – 32, which describe the corresponding value of the variable *targetValue*, we also need to replace $\neg \mathit{sensor\_signal}_{12}$ by $\mathit{power\_sensor\_signal}$.

The manual analyze whether we cover all the possible cases in the definition of *targetValue* for the state S_4 is not sufficient, if the semiautomated proof of the system properties is planed.

6.1.17 State S_5

The description of the system behavior at the state S_5 is given by 9th, 10th, 20th and 21st formulas.

In the formula 20 we need to replace $\neg sensor\_signal_{12}$ by $power\_sensor\_signal$. Analyzing this formula we can see that in the case of $\mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle$ we have no dependencies on $current\_value$. We simplify it as follows:

$$SystemState = S_5 \;\wedge\; power\_sensor\_signal \;\wedge$$
$$(\mathsf{ti}(signal_1, t) \neq \langle SignalA_7 \rangle \;\vee\; current\_value^t_{\mathsf{ft}} \geq min(MaxCurrentValue, MaxTargetValue))$$
$$\rightarrow\; SystemState' = S_4$$

The situation described by the 21st formula is covered by more general 25th formula, thus, we can simply remove the 21st formula from the specification. We also need to add a formula which describes what happens if none of the cases of the 20th formula is applicable:

$$SystemState = S_5 \;\wedge\; power\_sensor\_signal \;\wedge$$
$$(\mathsf{ti}(signal_1, t) = \langle SignalA_7 \rangle \;\wedge\; current\_value^t_{\mathsf{ft}} < min(MaxCurrentValue, MaxTargetValue))$$
$$\rightarrow\; SystemState' = S_5$$

The manual analyze whether we cover all the possible cases in the definition of $targetValue$ for the state S_5 is not sufficient, if the semiautomated proof of the system properties is planed.

6.1.18 State S_6

The description of the system behavior at the state S_5 is given by the 9th, 10th, 22nd and 23rd formulas. In the formula 22 we need to replace $\neg sensor\_signal_{12}$ by $power\_sensor\_signal$. We can see that in the case of $\mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle$ we have no dependencies on $current\_value$ and simplify it as follows:

$$SystemState = S_6 \;\wedge\; power\_sensor\_signal \;\wedge$$
$$(\mathsf{ti}(signal_1, t) \neq \langle SignalA_8 \rangle) \;\vee\; current\_value^t_{\mathsf{ft}} \leq max(MinCurrentValue, MinTargetValue))$$
$$\rightarrow\; SystemState' = S_4$$

The situation described by the 23rd formula is covered by more general 25th formula, thus, we can simply remove the 23rd formula from the specification. We also need to add a formula which describes what happens if none of the cases of the 22nd formula is applicable:

$$SystemState = S_6 \;\wedge\; power\_sensor\_signal \;\wedge$$
$$(\mathsf{ti}(signal_1, t) = \langle SignalA_8 \rangle \;\wedge\; current\_value^t_{\mathsf{ft}} > min(MaxCurrentValue, MaxTargetValue))$$
$$\rightarrow\; SystemState' = S_6$$

The manual analyze whether we cover all the possible cases in the definition of $targetValue$ for the state S_6 is not sufficient, if the semiautomated proof of the system properties is planed.

6.1.19 State S_7

The description of the system behavior at the state S_7 is given explicitly by the 24th formula, which is in contradiction with the 3rd formula because of typo: the correct conjunct must be $\mathsf{ti}(power1, t) \neq \langle\rangle$. The system behavior at the state S_7 for the case of the empty signal $power1$ is given by the 3rd formula.

LogicMain(const LogicParam) — timed

in $\mathit{sensor\_signal1}, \mathit{sensor\_signal2}, \mathit{sensor\_signal3} : \mathbb{B}\mathrm{ool};\ \mathit{signal}_1 : \mathit{SignalType}$
$\mathit{current\_value}, \mathit{counter1}, \mathit{counter2} : \mathbb{N};\ \mathit{pre}_1, \mathit{pre}_2, \mathit{pre}_3, \mathit{pre}_4, \mathit{pre}_5, \mathit{power1}, \mathit{power2} : \mathit{Event}$

out $\mathit{target\_value\_1}, \mathit{target\_value\_2} : \mathbb{N};\ \ \mathit{stateInf}, \mathit{stateInfOut} : \mathit{StateType}$

local $\mathit{SystemState} : \mathit{StateType};\ \mathit{targetValue} : \mathbb{N}$

init $\mathit{SystemState} = S_0;\ \mathit{targetValue} = 0;$

asm $\mathsf{ts}(\mathit{sensor\_signal1}) \wedge \mathsf{ts}(\mathit{sensor\_signal2}) \wedge \mathsf{ts}(\mathit{sensor\_signal3})$
$\mathsf{msg}_1(\mathit{signal}_1) \wedge \mathsf{ts}(\mathit{current\_value}) \wedge \mathsf{ts}(\mathit{counter1}) \wedge \mathsf{ts}(\mathit{counter2}) \wedge \mathsf{msg}_1(\mathit{power1}) \wedge \mathsf{msg}_1(\mathit{power2})$
$\mathsf{ts}(\mathit{pre}_1) \wedge \mathsf{ts}(\mathit{pre}_2) \wedge \mathsf{ts}(\mathit{pre}_3) \wedge \mathsf{ts}(\mathit{pre}_4) \wedge \mathsf{ts}(\mathit{pre}_5)$

gar

1 $\mathit{stateInfOut} = \mathit{stateInf} \wedge \mathit{target\_value\_2} = \mathit{target\_value\_1}$

$\forall\, t \in \mathbb{N}:$

2 $\mathsf{ti}(\mathit{stateInf}, t) = \langle \mathit{SystemState} \rangle \wedge \mathsf{ti}(\mathit{target\_value\_1}, t) = \langle \mathit{targetValue} \rangle$

3 $\mathsf{ti}(\mathit{power1}, t) = \langle\rangle \rightarrow \mathit{CrCtSate}' = S_0$

4 $\mathit{SystemState} = S_0 \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \rightarrow \mathit{CrCtSate}' = S_1$

5 $\mathit{SystemState} = S_1 \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge \mathsf{ft.ti}(\mathit{pre}_1, t+1) \wedge \mathsf{ft.ti}(\mathit{pre}_2, t+1) \wedge \mathsf{ft.ti}(\mathit{pre}_3, t+1) \wedge \mathsf{ft.ti}(\mathit{pre}_4, t+1) \wedge \mathsf{ft.ti}(\mathit{pre}_5, t+1) \rightarrow \mathit{SystemState}' = S_2$
6 $\mathit{SystemState} = S_1 \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge (\neg\, \mathsf{ft.ti}(\mathit{pre}_1, t+1) \vee \neg\, \mathsf{ft.ti}(\mathit{pre}_2, t+1) \vee \neg\, \mathsf{ft.ti}(\mathit{pre}_3, t+1) \vee \neg\, \mathsf{ft.ti}(\mathit{pre}_4, t+1) \vee \neg\, \mathsf{ft.ti}(\mathit{pre}_5, t+1)) \rightarrow \mathit{SystemState}' = S_1$

7 $\mathit{SystemState} = S_2 \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge (\neg \mathit{sensor\_signal}_{12} \wedge \neg\, \mathit{Signal1Precondition}(\mathsf{ti}(\mathit{signal}_1, t))) \rightarrow \mathit{SystemState}' = S_3$
8 $\mathit{SystemState} = S_2 \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge (\neg \mathit{sensor\_signal}_{12} \wedge \mathit{Signal1Precondition}(\mathsf{ti}(\mathit{signal}_1, t))) \rightarrow \mathit{SystemState}' = S_2$

9 $(\mathit{SystemStateSubset}(\mathit{SystemState}) \vee \mathit{SystemState} = S_2) \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge \mathsf{ti}(\mathit{sensor\_signal1}, t) = \langle \mathsf{true} \rangle \rightarrow \mathit{SystemState}' = S_7$

10 $\mathit{SystemStateSubset}(\mathit{SystemState}) \wedge \neg \mathsf{ti}(\mathit{power1}, t) = \langle\rangle \wedge \mathsf{ti}(\mathit{power2}, t) = \langle\rangle \rightarrow \mathit{SystemState}' = S_1$

11 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_3 \rangle \rightarrow \mathit{SystemState}' = S_4$
12 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{targetValue} > 0 \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle \rightarrow \mathit{SystemState}' = S_4$
13 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{targetValue} = 0 \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle \rightarrow \mathit{SystemState}' = S_3$
14 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_7 \rangle \rightarrow \mathit{SystemState}' = S_5$
15 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_8 \rangle \rightarrow \mathit{SystemState}' = S_6$
16 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{false} \rangle \wedge (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle\rangle \rightarrow \mathit{SystemState}' = S_3$

16 $\mathit{SystemState} = S_4 \wedge \mathit{power\_sensor\_signal} \wedge$
$(\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_3 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle) \rightarrow \mathit{SystemState}' = S_4$

17 $\mathit{SystemState} = S_4 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_7 \rangle \rightarrow \mathit{SystemState}' = S_5$
18 $\mathit{SystemState} = S_4 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_8 \rangle \rightarrow \mathit{SystemState}' = S_6$
19 $(\mathit{SystemState} = S_4 \vee \mathit{SystemState} = S_5) \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle\rangle \rightarrow \mathit{SystemState}' = S_6$

20 $\mathit{SystemState} = S_5 \wedge \mathit{power\_sensor\_signal} \wedge (\mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_7 \rangle \vee \mathit{current\_value}^t_{\mathsf{ft}} \geq \mathit{min}(\mathit{MaxCurrentValue}, \mathit{MaxTargetValue})) \rightarrow \mathit{SystemState}' = S_4$
21 $\mathit{SystemState} = S_5 \wedge \mathit{power\_sensor\_signal} \wedge (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_7 \rangle \wedge \mathit{current\_value}^t_{\mathsf{ft}} < \mathit{min}(\mathit{MaxCurrentValue}, \mathit{MaxTargetValue})) \rightarrow \mathit{SystemState}' = S_5$

22 $\mathit{SystemState} = S_6 \wedge \mathit{power\_sensor\_signal} \wedge (\mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_8 \rangle) \vee \mathit{current\_value}^t_{\mathsf{ft}} \leq \mathit{max}(\mathit{MinCurrentValue}, \mathit{MinTargetValue})) \rightarrow \mathit{SystemState}' = S_4$
23 $\mathit{SystemState} = S_6 \wedge \mathit{power\_sensor\_signal} \wedge (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_8 \rangle \wedge \mathit{current\_value}^t_{\mathsf{ft}} > \mathit{min}(\mathit{MaxCurrentValue}, \mathit{MaxTargetValue})) \rightarrow \mathit{SystemState}' = S_6$

24 $\mathit{SystemState} = S_7 \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \rightarrow \mathit{SystemState}' = S_7$

25 $(\mathit{SystemStateSubset}(\mathit{SystemState}) \vee \mathit{SystemState} = S_2) \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge \mathsf{ti}(\mathit{sensor\_signal2}, t) = \langle \mathsf{true} \rangle \wedge \mathsf{ti}(\mathit{sensor\_signal1}, t) = \langle \mathsf{false} \rangle \rightarrow \mathit{SystemState}' = S_2$

26 $\mathit{SystemState} = S_0 \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \rightarrow \mathit{targetValue}' = 0$

27 $(\mathit{SystemState} = S_1 \vee \mathit{SystemState} = S_2 \vee (\mathit{SystemStateSubset}(\mathit{SystemState}) \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge \mathsf{ti}(\mathit{power2}, t) = \langle\rangle)) \rightarrow \mathit{targetValue}' = \mathit{targetValue}$

28 $\mathit{SystemStateSubset}(\mathit{SystemState}) \wedge \mathsf{ti}(\mathit{sensor\_signal3}, t) = \langle \mathsf{true} \rangle \rightarrow \mathit{targetValue}' = 0$

28 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_3 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_7 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_8 \rangle) \rightarrow \mathit{targetValue}' = \mathit{limTargetValue}$
29 $\mathit{SystemState} = S_3 \wedge \mathit{power\_sensor\_signal} \wedge$
$((\mathit{targetValue} > 0 \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_4 \rangle) \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \vee \mathsf{ti}(\mathit{signal}_1, t) = \langle\rangle) \rightarrow \mathit{targetValue}' = \mathit{targetValue}$

30 $\mathit{SystemState} = S_4 \wedge \mathit{power\_sensor\_signal} \wedge$
$(\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_3 \rangle \vee$
$(\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \wedge \mathit{ModSubtraction}(\mathit{current\_value}^t_{\mathsf{ft}}, \mathit{targetValue}) > \mathit{X\_Appl}) \vee (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \wedge \mathsf{ti}(\mathit{counter2}, t) > 0) \vee$
$(\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \wedge \mathit{ModSubtraction}(\mathit{current\_value}^t_{\mathsf{ft}}, \mathit{targetValue}) > \mathit{X\_Appl}) \vee (\mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \wedge \mathsf{ti}(\mathit{counter1}, t) > 0))$
$\rightarrow \mathit{targetValue}' = \mathit{limTargetValue}$

31 $\mathit{SystemState} = S_4 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_5 \rangle \wedge \mathit{SignalAccepted}(\mathsf{true}, \mathit{current\_value}^t_{\mathsf{ft}}, \mathit{targetValue}, \mathit{counter1}^t_{\mathsf{ft}}, \mathit{counter2}^t_{\mathsf{ft}})$
$\rightarrow \mathit{targetValue}' = \mathit{ChangeTargetValue}(\mathit{targetValue}, \mathit{SignalA}_5)$

32 $\mathit{SystemState} = S_4 \wedge \mathit{power\_sensor\_signal} \wedge \mathsf{ti}(\mathit{signal}_1, t) = \langle \mathit{SignalA}_6 \rangle \wedge \mathit{SignalAccepted}(\mathsf{false}, \mathit{current\_value}^t_{\mathsf{ft}}, \mathit{targetValue}, \mathit{counter1}^t_{\mathsf{ft}}, \mathit{counter2}^t_{\mathsf{ft}})$
$\rightarrow \mathit{targetValue}' = \mathit{ChangeTargetValue}(\mathit{targetValue}, \mathit{SignalA}_6)$

33 $\mathit{SystemState} = S_5 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{current\_value}^t_{\mathsf{ft}} > \mathit{targetValue} \wedge \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_7 \rangle \rightarrow \mathit{targetValue}' = \mathit{limTargetValue}$
34 $\mathit{SystemState} = S_5 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{current\_value}^t_{\mathsf{ft}} \leq \mathit{targetValue} \wedge \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_7 \rangle \rightarrow \mathit{targetValue}' \neq 0$
35 $\mathit{SystemState} = S_5 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{current\_value}^t_{\mathsf{ft}} \geq \mathit{min}(\mathit{MaxCurrentValue}, \mathit{MaxTargetValue}) \rightarrow \mathit{targetValue}' = \mathit{min}(\mathit{MaxCurrentValue}, \mathit{MaxTargetValue})$
36 $(\mathit{SystemState} = S_5 \vee \mathit{SystemState} = S_6) \wedge \mathit{sensor\_signal2}^t_{\mathsf{ft}} \wedge \neg \mathit{sensor\_signal1}^t_{\mathsf{ft}} \wedge \neg \mathit{sensor\_signal3}^t_{\mathsf{ft}} \rightarrow \mathit{targetValue}' = \mathit{targetValue}$
37 $\mathit{SystemState} = S_6 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{current\_value}^t_{\mathsf{ft}} < \mathit{targetValue} \wedge \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_8 \rangle \rightarrow \mathit{targetValue}' = \mathit{limTargetValue}$
38 $\mathit{SystemState} = S_6 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{current\_value}^t_{\mathsf{ft}} \geq \mathit{targetValue} \wedge \mathsf{ti}(\mathit{signal}_1, t) \neq \langle \mathit{SignalA}_8 \rangle \rightarrow \mathit{targetValue}' \neq 0$
39 $\mathit{SystemState} = S_6 \wedge \mathit{power\_sensor\_signal} \wedge \mathit{current\_value}^t_{\mathsf{ft}} \leq \mathit{max}(\mathit{MinCurrentValue}, \mathit{MinTargetValue}) \rightarrow \mathit{targetValue}' = \mathit{max}(\mathit{MinCurrentValue}, \mathit{MinTargetValue})$

where $\mathit{sensor\_signal}_{12}, \mathit{power\_sensor\_signal}, \mathit{limTargetValue}$ so that

$\mathit{sensor\_signal}_{12} = \mathit{sensor\_signal2}^t_{\mathsf{ft}} \vee \mathit{sensor\_signal1}^t_{\mathsf{ft}}$
$\mathit{power\_sensor\_signal} = \neg \mathit{sensor\_signal}_{12} \wedge \mathsf{ti}(\mathit{power1}, t) \neq \langle\rangle \wedge \mathsf{ti}(\mathit{power2}, t) \neq \langle\rangle$
$\mathit{limTargetValue} = \mathit{LimitedValue}(\mathit{current\_value}^t_{\mathsf{ft}}, \mathit{MinCurrentValue}, \mathit{MinTargetValue}, \mathit{MaxCurrentValue}, \mathit{MaxTargetValue})$

6.2 LogicMain Component: Timed State Transition Diagram

The specification *LogicMain* is semantically equal to the specification using a timed state transition diagram (TSTD), with 8 states, which correspond to the values of the local variable *SystemState*: $S_0, \ldots, S_7$. We take S_0 as the initial state, because of to the initial value of the variable *SystemState*.

Please note that according to [10] for the TSTDs the following rules hold:

- The argumentation is over time intervals, the "current" time interval number is t, $t \in \mathbb{N}$.
- For any stream y from the *input* channels used in the TSTD: if an expression of the form $\mathsf{ti}(y, t) = \mathit{SomeTimeInterval}$ is omitted, the value of the tth time interval of the stream y can be arbitrary.
- For any stream z from the *output* channels used in the TSTD: all expression of the form $\mathsf{ti}(z, t) = \langle\rangle$ are omitted.
- For any local variable l all expression of the form $l' = l$ are omitted.

The init-part of the specification defines the starting output values, where the 1st formula of the body-part of the specification *LogicMain*

$$\mathit{stateInfOut} = \mathit{stateInf} \ \wedge \ \mathit{target\_value\_2} = \mathit{target\_value\_1}$$

specifies a general equality on the outputs, which must be added to each transition in the same manner as the equalities from the 2nd formula of *LogicMain*:

$$\mathsf{ti}(\mathit{stateInf}, t) = \langle \mathit{SystemState} \rangle \ \wedge \ \mathsf{ti}(\mathit{target\_value\_1}, t) = \langle \mathit{targetValue} \rangle$$

After translation these formulas, which operates with a single current state, to the state transitions, we get a TSTD that is relatively readable, but we need also to add to the TSTD the transitions that represents formulas that do not operate with a single current state, but with a number of states, which correspond to some properties.

The formula 3 has no information about the current state. This implies that the corresponding transition must be added to each state. Therefore, we need to add eight transitions to our TSTD . For better readability we mark them green.

The formula 10 holds for four state (according to the definition of the predicate *SystemStateSubset*) – we need to add four corresponding transitions to our TSTD. For better readability we mark them purple.

The formulas 9 and 25 hold for five states (the S_2-state and the four states, for which holds the predicate *SystemStateSubset*). Therefore, we need to add five corresponding transitions to our TSTD for each of these two cases. For better readability we mark them blue and red correspondingly.

As result we get the complete TSTD for the *LogicMain* component.

Here we present a simplified version of the TSTD – *LogicMainTSTD* (see Figure 4), omitting the local variables calculation – this representation is more readable for the case one want to understand the main state transitions.

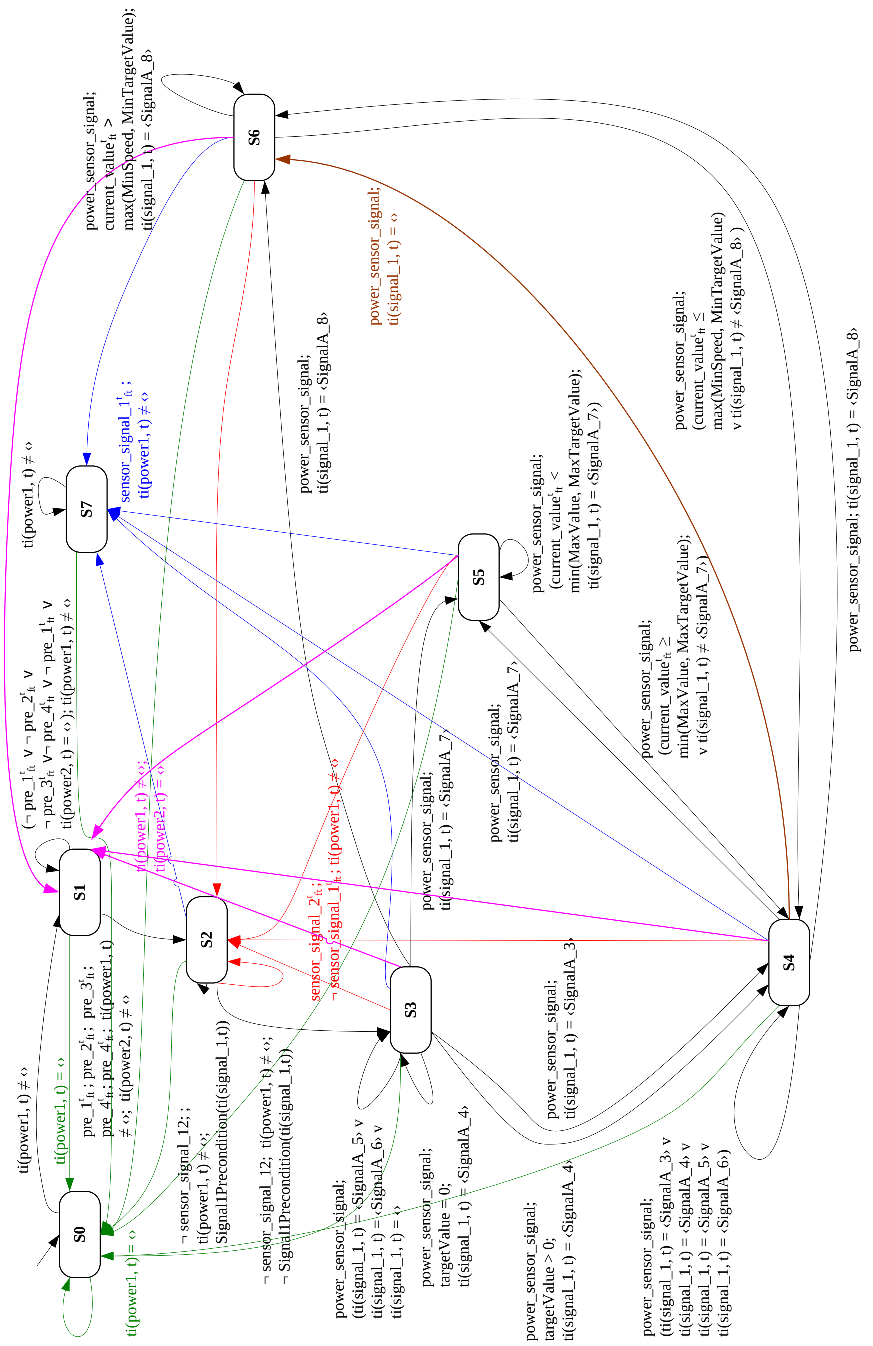


Figure 4: Timed state transition diagram *LogicMainTSTD* for the component LogicMain

7 Specification of the System Requirements

In this section we define the requirements on the component *LogicComp*.

Please note, that the specification *LogicCompReq* contains not all requirements which can be important for the component *LogicComp* – our aim here was to present a number of examples for such kind of specifications.

LogicCompReq(const LogicParam) — timed

in
$sensor\_signal1, sensor\_signal2, sensor\_signal3, sensor\_signal4, sensor\_signal5, sensor\_signal6 : \mathbb{B}ool$
$signal_1 : SignalType; \quad current\_value, counter1, counter2 : \mathbb{N};$
$precondition_1, precondition_2, precondition_3, precondition_4, precondition_5 : Event$
$signal_2, signal_3, power1, power2 : Event$

out
$target\_value\_1, target\_value\_2 : \mathbb{N}; \quad stateInfOut : StateType;$
$SignalA7Action, SignalA8Action, S_4Action, event_1, event_2, event_3 : Event;$
$indicator\_lamp\_is\_On : \mathbb{B}ool; \; request, systemSignal1 : \mathbb{N};$

asm
$\mathsf{ts}(sensor\_signal1) \wedge \mathsf{ts}(sensor\_signal2) \wedge \mathsf{ts}(sensor\_signal3)$
$\mathsf{ts}(sensor\_signal4) \wedge \mathsf{ts}(sensor\_signal5) \wedge \mathsf{ts}(sensor\_signal6)$
$\mathsf{msg}_1(signal_1) \wedge \mathsf{ts}(current\_value) \wedge \mathsf{ts}(counter1) \wedge \mathsf{ts}(counter2)$
$\mathsf{msg}_1(precondition_1) \wedge \mathsf{msg}_1(precondition_2) \wedge \mathsf{msg}_1(precondition_3)$
$\mathsf{msg}_1(precondition_4) \wedge \mathsf{msg}_1(precondition_5)$
$\mathsf{msg}_1(power1) \wedge \mathsf{msg}_1(power2) \wedge \mathsf{msg}_1(signal_3) \wedge \mathsf{msg}_1(signal_2)$

gar
$\forall\, t \in \mathbb{N}:$

$stateInfOut^t_{\mathsf{ft}} = S_2 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle$
$\rightarrow stateInfOut^{t+1}_{\mathsf{ft}} = S_2 \vee stateInfOut^{t+1}_{\mathsf{ft}} = S_3 \vee stateInfOut^{t+1}_{\mathsf{ft}} = S_7$

$stateInfOut^t_{\mathsf{ft}} = S_4 \rightarrow \mathsf{ti}(targetValue_1, t) \neq \langle 0\rangle$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_5 \wedge stateInfOut^t_{\mathsf{ft}} = S_4 \wedge$
$SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, target\_value\_1^t_{\mathsf{ft}}, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow \mathsf{ti}(target\_value\_1, t+1) = \langle ChangeTargetValue(target\_value\_1^t_{\mathsf{ft}}, SignalA_5)\rangle$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_5 \wedge stateInfOut^t_{\mathsf{ft}} = S_4 \wedge$
$\neg\, SignalAccepted(\mathsf{true}, current\_value^t_{\mathsf{ft}}, target\_value\_1^t_{\mathsf{ft}}, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow \mathsf{ti}(target\_value\_1, t+1) = \langle limTargetValue\rangle$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_6 \wedge stateInfOut^t_{\mathsf{ft}} = S_4 \wedge$
$SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, target\_value\_1^t_{\mathsf{ft}}, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow \mathsf{ti}(target\_value\_1, t+1) = \langle ChangeTargetValue(targetValueInf^t_{\mathsf{ft}}, SignalA_6)\rangle$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_6 \wedge stateInfOut^t_{\mathsf{ft}} = S_4 \wedge$
$\neg\, SignalAccepted(\mathsf{false}, current\_value^t_{\mathsf{ft}}, target\_value\_1^t_{\mathsf{ft}}, counter1^t_{\mathsf{ft}}, counter2^t_{\mathsf{ft}})$
$\rightarrow \mathsf{ti}(target\_value\_1, t+1) = \langle limTargetValue\rangle$

$\neg power\_sensor\_signal \wedge SystemStateSubset(stateInfOut^t_{\mathsf{ft}})$
$\rightarrow (\mathsf{ti}(stateInfOut, t+1) = \langle S_0\rangle \vee \langle S_1\rangle \vee \langle S_2\rangle \vee \mathsf{ti}(stateInfOut, t+1) = \langle S_7\rangle)$

$stateInfOut^t_{\mathsf{ft}} = S_7 \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \rightarrow (\mathsf{ti}(stateInfOut, t+1) = \langle S_7\rangle)$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_3 \wedge (stateInfOut^t_{\mathsf{ft}} = S_3 \vee stateInfOut^t_{\mathsf{ft}} = S_4)$
$\rightarrow \mathsf{ti}(stateInfOut, t+1) = \langle S_4\rangle \wedge \mathsf{ti}(target\_value\_1, t+1) = \langle limTargetValue\rangle$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_4 \wedge stateInfOut^t_{\mathsf{ft}} = S_3 \wedge \mathsf{ft.ti}(target\_value\_1, t) > 0$
$\rightarrow \mathsf{ti}(stateInfOut, t+1) = S_4$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_4 \wedge stateInfOut^t_{\mathsf{ft}} = S_3 \wedge \mathsf{ti}(target\_value\_1, t) = \langle 0\rangle$
$\rightarrow \mathsf{ti}(stateInfOut, t+1) = S_3$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_7 \wedge (stateInfOut^t_{\mathsf{ft}} = S_3 \vee stateInfOut^t_{\mathsf{ft}} = S_4)$
$\rightarrow \mathsf{ti}(stateInfOut, t+1) = S_5$

$power\_sensor\_signal \wedge \mathsf{ft.ti}(signal_1, t) = SignalA_8 \wedge (stateInfOut^t_{\mathsf{ft}} = S_3 \vee stateInfOut^t_{\mathsf{ft}} = S_4)$
$\rightarrow \mathsf{ti}(stateInfOut, t+1) = S_5$

where $power\_sensor\_signal, limTargetValue$ so that
$power\_sensor\_signal =$
$\neg sensor\_signal2^t_{\mathsf{ft}} \wedge \neg sensor\_signal1^t_{\mathsf{ft}}) \wedge \mathsf{ti}(power1, t) \neq \langle\rangle \wedge \mathsf{ti}(power2, t) \neq \langle\rangle$
$limTargetValue =$
$LimitedValue(current\_value^t_{\mathsf{ft}}, MinCurrentValue, MinTargetValue, MaxCurrentValue, MaxTargetValue)$

8 Summary

In this paper we have presented a part of specification and verification process developed within the Verisoft-XT project. The purpose of this project is to integrate verification techniques in real industrial development processes – from specification and analysis of requirements to a verified implementation.

Ones of the main points in this paper are system architecture and system decomposition. The main contribution of our decomposition methodology is that it was developed for such a system architecture, where we know systems (components) properties and need to decompose this whole properties collection to a number of subcomponent. Thus, the presented methodology allows us to decompose component architecture decomposition exactly on this point where we see that the component specification becomes too large and too complex. In addition, our methodology helps to perform the next modeling step – translation to the case tool representation and deployment.

We can also see this methodology as an extension of the approach "Focus on Isabelle" – it is integrated into a seamless development process, which covers both specification and verification, starts from informal specification and finishes by the corresponding verified C code.

The starting point of presented approach is a semiformal requirement specification – according to it we represent the system in Focus according to the approach "Focus on Isabelle". Using this approach one can validate the refinement relation between two given systems, as well as make automatic correctness proofs of syntactic interfaces for specified system components. Having a Focus specification, we can schematically translate it to a specification in Hight-Order Logic and verify properties of the specified system.

We present and discuss here the Focus specifications of an imaginary case study that is an anonymization of the of the case study [11] from Robert Bosch GmbH: the used data types, constants, auxiliary functions and predicates, the general system architecture and the system components as well as their decomposition according the presented methodology, as well as the system requirements.

Given a system, represented in Focus, one can verify its properties by translating the specification to a Higher-Order Logic and subsequently using the theorem prover Isabelle/HOL or the point of disagreement can be found. This must be done as the next step of the methodology. As an other next step we can schematically translate the Focus specification to a model in the n AutoFocus 3 case tool and analyze the structure and behavior of the system, simulate it, prove its properties using model checking and also using its translation to Isabelle/HOL, as well as we gan generate C code from it.

References

[1] AutoFOCUS 3. `http://af3.in.tum.de`.

[2] M. Broy and K. Stølen. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement.* Springer-Verlag, 2001.

[3] Manfred Broy, Franz Huber, and Bernhard Schätz. AutoFocus – Ein Werkzeugprototyp zur Entwicklung eingebetteter Systeme. *Informatik Forschung und Entwicklung*, (13(3)):121–134, 1999.

[4] David Bettencourt da Cruz and Birgit Penzenstadler. Designing, Documenting, and Evaluating Software Architecture. Technical Report TUM-INFO-06-I0818-0/1.-FI, Technische Universität München, Institut für Informatik, Boltzmannstr. 3, 85748 Garching, GERMANY, jun 2008. available at http://www.in.tum.de/forschung/publikationen/index.html.de.

[5] Andreas Fleischmann. *Model-based formalization of requirements of embedded automotive systems.* PhD thesis, Technische Universität München, 2008.

[6] Christoph Hofmann, Eckart Horn, Wolfgang Keller, Klaus Renzel, Monika Schmidt, Wirth Horn, and Bereiter Anger. Approaches to software architecture.

[7] T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[8] Jan Philipps and Bernhard Rumpe. Refinement of Pipe-and-Filter Architectures. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM'99 – Formal Methods, Proceedings of the World Congress on Formal Methods in the Development of Computing System*, number LNCS 1708, pages 96 – 115. Springer, 1999.

[9] Bernhard Schätz and Franz Huber. Integrating Formal Description Techniques. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *FM'99 – Formal Methods, Proceedings of the World Congress on Formal Methods in the Development of Computing Systems*, volume 1709 of *Lecture Notes in Computer Science*, pages 1206 – 1225. Springer Verlag, sep 1999.

[10] M. Spichkova. *Specification and Seamless Verification of Embedded Real-Time Systems: FOCUS on Isabelle.* PhD thesis, Technische Universität München, 2007.

[11] Maria Spichkova. Architecture: Methodology of Decomposition. Specification of the Cruise Control System. Case Study. Technical report, Robert Bosch GmbH, 2010.

[12] Maria Spichkova. Verisoft XT Automotive Application: Semiformal Specification for Cruise Control System. Technical report, Robert Bosch GmbH, 2010.

[13] Verisoft Project. `http://www.verisoft.de`.

[14] Verisoft-XT Project. `http://www.verisoftxt.de`.

[15] M. Wenzel. *The Isabelle/Isar Reference Manual*. Technische Universität München, 2004.

[16] Doris Wild, Andreas Fleischmann, Judith Hartmann, Christian Pfaller, Martin Rappl, and Sabine Rittmann. An Architecture-Centric Approach towards the Construction of Dependable Automotive Software. In *Proceedings of the SAE 2006 World Congress*, 2006.