Proving Safety Properties of the Steam Boiler Controller

Formal Methods for Industrial Applications: A Case Study

Gunter Leeb
leeb@auto.tuwien.ac.at
Vienna University of Technology
Department for Automation
Treitlstr. 3, A-1040 Vienna, Austria

Nancy Lynch
lynch@lcs.mit.edu
Massachusetts Institute for Technology
Laboratory for Computer Science
Technology Square 545, Cambridge, MA

Abstract

In this paper we model a hybrid system consisting of a continuous steam boiler and a discrete controller. Our model uses the Lynch-Vaandrager Timed Automata model to show formally that certain safety requirements can be guaranteed under the described assumptions and failure model. We prove incrementally that a simple controller model and a controller model tolerating sensor faults preserve the required safety conditions. The specification of the steam boiler and the failure model follow the specification problem for participants of the Dagstuhl Meeting "Methods for Semantics and Specification."

1 Introduction

The number of different formal methods for specifying, designing, and analyzing real-time systems has grown difficult to survey. For the purpose of comparison, some problems have been defined or borrowed from real-life applications. One such benchmark problem is the Steam Boiler Controller problem discussed in this paper. Another representative of this kind of problem is the Generalized Railroad Crossing (GRC) [Hei93]. Various approaches have been applied to the latter, e.g., [Cle93,Jah86,Sha93,Hoa93]. Many steps of the approach described here are similar to the steps described in [Hei94].

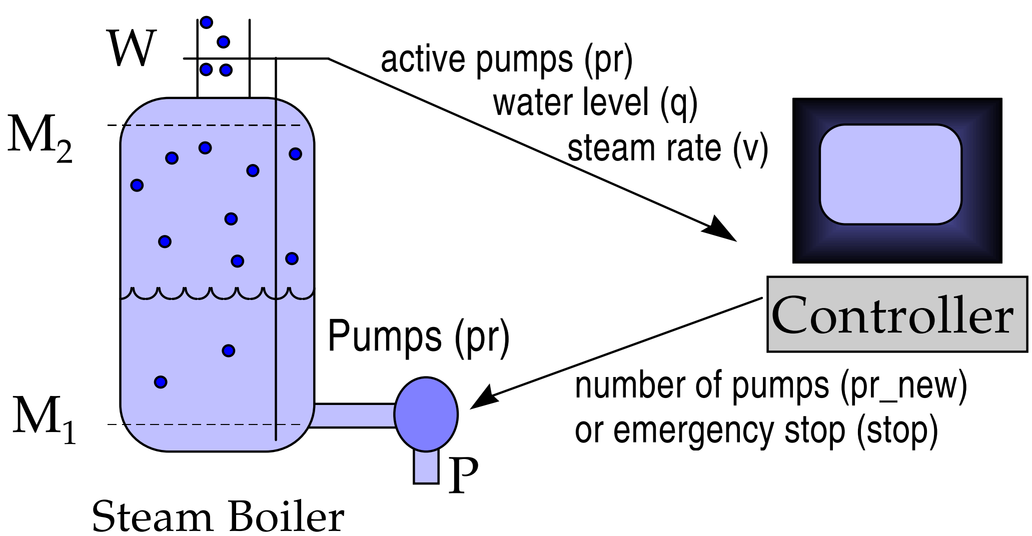


Figure 1: The steam boiler system. This picture shows the information flow between the controller and the steam boiler. It also gives some notion about the capacities of a pump (***P***), the limits for the steam rate (***W***) and the boundaries for the water level (***M_1*** and ***M_2***). A clock periodically states when the pumps are set and the sensors read and the user can shut down the system with the emergency stop button.

However, the Steam Boiler Controller represents a different kind of problem. Basically, it consists of a discrete control loop where several components may fail. We now give a condensed and informal version of the Steam Boiler Controller specification. The original specification can be found in [AS96]. Since even the detailed specification is informal and ambiguous, the following summarizes our interpretation of the described problem. For easier understanding of the following discussion, we include some abbreviations for variables used in the analysis:

> The physical plant consists of a steam boiler. Conceptually, this boiler is heated (e.g., by nuclear fuel) and the water in the boiler evaporates into steam and escapes the boiler to drive, e.g., a generator (this part is of no concern to the problem). The amount of heat and, therefore, the amount of steam changes without any considered control. Nevertheless, the safety of the boiler depends on a bounded water level (**q**) in the boiler and steam rate (**v**) at its exit.* A set of four equal pumps may supply water to compensate for the steam that leaves the boiler. These four pumps can be activated or stopped by the controller system. The controller reacts to the information of two sensors, the water level sensor and the steam rate sensor, and both may fail. Moreover, the controller can deduce from a pump monitor whether the pumps are working correctly. Sensor data are transferred to the controller system periodically. The controller reacts instantaneously with a new setting for the pumps (**pr_new**) or decides to shut-down the boiler system (**stop**).
>
> There are two basic time constants: First, the time between two consecutive sensor readings (denoted ***I***)† and, second, the delay time (***S***) until the reaction of the controller causes consequences in the boiler. The latter delay time usually represents a worst case accumulation of sensor reading delay, calculation time in the controller, message delivery time, reaction time of the pumps, and other minor factors.
>
> The water level has two safety limits, one upper (denoted M_2) and one lower limit (denoted M_1). If the water level reaches either limit, there is just time enough to shut down the system before the probability of a catastrophe gets unacceptably high. The steam rate has an upper limit (denoted ***W***) and, again, if this limit is reached the boiler must be stopped immediately. In addition the human operator has the possibility to activate the shut down anytime.

The above description gives an overview of the essential parts of the problem and a reduction to the central aspects of this problem with the main purpose of resolving some ambiguity in the specification. The specification includes some additional technicalities which we mostly ignore.

The rest of this paper is organized as follows: After presenting an outline of our formal methods (Section 2), we state the assumptions we make for our model and show how the model is related to the physical model (Section 3). The following two sections describe the model of the boiler and a simple controller. In Section 6, we show some key model invariants. In Section 7, we present a similar controller which allows for sensor faults and we show its correctness incrementally based on the simpler controller model.

2 The Formal Framework

Applying formal methods to a system involves three steps: the system requirements specification, the design of an implementation, and the verification that the implementation satisfies the specification. The system requirements specification describes all acceptable system implementations [Hei94]. It has three parts:

1. A formal model describing the environment (e.g., the steam boiler) and its interface
2. A formal model describing the controller system and its interface at an abstraction level
3. Formal statements of the properties that the system must satisfy

---

* Most variable names are according to the original specification in [AS96].
† Capital letters denote constants of the problem.

The formal method we used to specify the steam boiler problem and to develop and verify a solution represents both the controller and the system environment as Timed Automata, according to the definition of Lynch and Vaandrager [Lyn91]. A Timed Automaton is a very general automaton, i.e., a labeled transition system. It is not finite-state: for example, the state can contain real-valued information, such as the current time or the current steam rate. This characteristic makes Timed Automata suitable for modeling not only discrete computer systems but also real-world entities such as the steam boiler. We base our work directly on an automaton model rather than on any particular specification language, programming language, or proof system, so that we may obtain the greatest flexibility in selecting specification and proof methods. The formal definition of a Timed Automaton appears in Appendix A. Appendix B describes the Simulation Mapping method used for incremental reasoning about other increasingly specific instances of the model.

The Timed Automaton model supports the description of systems as collections of Timed Automata, interacting by means of common actions. In our example, we define separate Timed Automata for the steam boiler and the controller system; the common actions are sensors reporting the current state of some parameters of the boiler and actuators controlling the pumps of the boiler.

Actions change the state and, in particular, some variables of the state of an automaton. As a distinction between variables of the pre-state and the post-state, we write variables of the post-state (or the representation of the whole post-state) with a prime. In changing the state, actions perform a step or transition. Such a step or transition defines the change from one state *s* to another state *s'* by an action *a*, which is formally written as *(s, a, s')* or $s_A \xrightarrow{a}_A s'_A$, where the subscript A stands for the name of the particular automaton.

For the communication with other automata, we define input, output and internal actions. Such input actions will be enabled by output actions of another automaton. For example, the actuator output action in the controller model is synchronized with the actuator input action of the steam boiler model. The inherent flexibility of the method allows, for example, the introduction of a new automaton representing channel and message transfer characteristics to be employed in-between the boiler automaton and the controller automaton, interfacing with an input action from the controller and an output action to the steam boiler model. This allows us to model more complex systems without major changes to the previous automata. Furthermore, with this composition, we can reuse information, we gained about the separate automata.

We describe the Timed Automata using precondition-effect notation. The precondition identifies particular states in which the system performs some actions. For any state fulfilling the precondition, the effect part describes how the state is changed by the particular action. This has several advantages. First of all, it is easy to understand. Even more important is that implementations can follow the abstract model description and even allow for simple validity checks in the code. In addition, all the invariants proved represent useful checks to be validated while running the final application. This approach will help to identify rare kinds of faults that are not even considered in the model. In this view, formal verification with Timed Automata is a constructive approach to systems development.

3 Further Considerations for Our Model

For our model, we need to know some more information about the physical behavior. Some of the following assumptions follow the informal specification of [AS96] or are intended to resolve some ambiguity. As suggested by [AS96], to simplify reasoning about the model, we ignore second order effects like the volume expansion of water when heated. This reasoning implies that a unit of water measured as steam can be replaced by pumping in exactly one unit of water.

Most important is some knowledge about how fast the steam rate may change over time. We assume a reasonable worst case situation where the steam rate increases at most with U_1 liters per second per second. In other words, the maximum gradient of increase of the steam rate is U_1 l/s^2. Symmetric to this, we know that the fastest decrease of the steam rate is denoted with U_2 l/s^2.

Furthermore, no pump supplies water unless activated and then it supplies a constant, exactly known amount of water per second denoted with ***P*** liters per second. The delay between reading the sensors and consequently changing the active pumps, denoted with ***S***, is caused mainly by the slow reaction of the physical pumps. As a minor difference to the specification in [AS96], we assume the same delay for the activation and the deactivation of pumps. Since the pumps cause most of the delay ***S***, we assume any boiler shut down is activated instantaneously and the whole process of shutting down the steam boiler is left to a later phase which we do not consider in this model. In the same way, we omit the initialization phase, which should force the boiler state into a particular acceptable set of start states before the boiler becomes fully operational. We assume all parameters of the start state for this model are already in their correct operational ranges. Moreover, we assume that the controller may decide to shut down the boiler any time it sets the new pumps. This assumption includes the possibility that the operator initiates an emergency stop and provides the flexibility to incorporate other reasons to shut down the boiler.

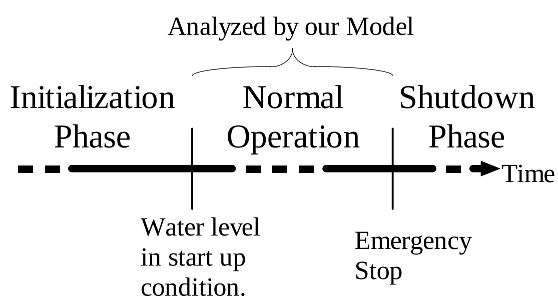


Figure 2: Our model only considers the time of normal operation. At the beginning, the initialization phase provides all parameters in the correct range and the shutdown phase is activated through setting parameter *stop* to ***true***.

Other helpful assumptions are correct and accurate sensor values or the detection of a sensor fault. Perfect fault detection and identification are necessary for our model but will not be available in reality. In this aspect our model might need improvement if it is necessary to study such general cases. For example, the techniques developed for probabilistic Timed Automata [Seg94] seem to be appropriate for a problem requiring the analysis of such probabilistic properties. Probabilistic Timed Automata would allow one to assign probabilities to certain actions, e.g., for a successful error detection, and to prove the probability of a certain system behavior.

As a further simplification, we choose a very simple fault model which, in fact, includes or is close to most common fault conditions. The fault model assumes that every pump may fail and stop pumping water into the boiler. As a minor simplification, we assume for our model that any pump fault only occurs at times when the pumps may be activated or stopped. This happens periodically whenever the parameter *set* equals the current time (*now*). Thus, pumps, when successfully activated, supply water at least to the next instant where pumps might change their behavior. Moreover, we assume that the activation delay, i.e., the time from reading the sensor values until consequently the pumps change their behavior, is smaller than the time between two successive sensor readings ($\boldsymbol{S} < \boldsymbol{I}$).

The goal of modeling the steam boiler and the controller with Timed Automata is to show certain important properties. In this case, we want to verify that our controller model does not violate safety. Therefore, we have to show that neither the steam rate nor the water level crosses its critical limits.

Next, we summarize the information we have about the physical model.

3.1 The Physical Model

We assume the steam rate expressed as a function over time ($sr(t) \geq 0$) is differentiable. Furthermore, we know that

$$-U_2 \leq \dot{sr}(t) \leq U_1$$

and

$$wl(t) = wl(0) + \int_0^t pr(x)dx - \int_0^t sr(x)dx$$

where $\dot{sr}(t)$ represents the derivative of the steam rate function and $wl(t)$ the amount of water in the boiler at the time t and $pr(t)$ (≥ 0) the (discrete) pump rate function over time. We apply the following transformation to this information to make our model easier to follow.

We know $-U_2 \leq \dot{sr}(t)$, which implies $0 \leq \dot{sr}(t) + U_2$ and in general

$$\int \dot{sr}(t) + U_2 dt = sr(t) + t * U_2 + C.$$

Thus, we know that for all Δt,

$$sr(t+\Delta t) + U_2 * \Delta t \geq sr(t)$$

and symmetrically

$$sr(t+\Delta t) - U_1 * \Delta t \leq sr(t).$$

In the following, we use s for $sr(t)$ and s_{new} for $sr(t + \Delta t)$. With a similar straightforward calculation as before, we get

$$wl(t+\Delta t) \geq wl(t) + \int_t^{t+\Delta t} pr(x)dx - \delta_{HIGH}(s, s_{new}, \Delta t)$$

and symmetrically

$$wl(t+\Delta t) \leq wl(t) + \int_t^{t+\Delta t} pr(x)dx - \delta_{LOW}(s, s_{new}, \Delta t)$$

with

$$\delta_{HIGH}\left(s, s_{new}, \Delta t\right) = \left(\frac{2\Delta t U_2 s + 2\Delta t U_1 s_{new} + \Delta t^2 U_1 U_2 - (s - s_{new})^2}{2U_1 + 2U_2}\right)$$

and

$$\delta_{LOW}\left(s, s_{new}, \Delta t\right) = \begin{cases} \left(\frac{2\Delta t U_1 s + 2\Delta t U_2 s_{new} - \Delta t^2 U_1 U_2 + (s - s_{new})^2}{2U_1 + 2U_2}\right) & \text{if} \quad \left(\frac{s}{U_2} + \frac{s_{new}}{U_1}\right) > \Delta t \\ \left(\frac{s^2}{2U_2} + \frac{s_{new}{}^2}{2U_1}\right) & \text{otherwise} \end{cases}$$

δ_{HIGH} describes the maximum amount of water that could evaporate and δ_{LOW} the minimum amount of water. Obviously, δ_{LOW} depends on whether the steam rate might drop to 0 in the interval Δt. Figure 3 represents δ_{HIGH} and δ_{LOW} graphically for an arbitrary interval t. Figure 3 ignores the pump rate, and the shaded areas represent the water evaporated into steam until a certain point in time. In other words, δ_{HIGH} and δ_{LOW} represent the worst case amount of water that could evaporate into steam in interval Δt. Both depend on the knowledge of the steam rate at the beginning and the end of the interval. The basic dependencies shown in the following Lemma 1 are sufficient for all further proofs.

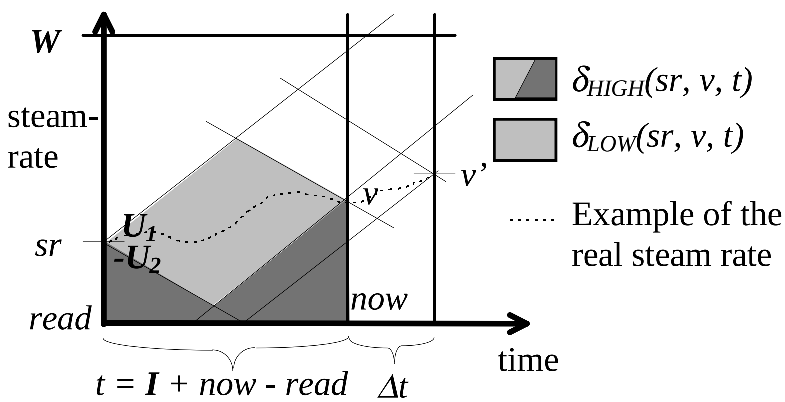


Figure 3: Example of what δ_{HIGH} and δ_{LOW} represent. For different intervals the maximum and minimum amount of water evaporated into steam depends on the steam rate at the beginning of the interval and at the end.

The following Lemma lists all necessary relations about the steam development functions δ_{HIGH} and δ_{LOW}. Some intuition for this lemma can be gained from Figure 3. Obviously, two consecutive intervals can be joined and the minimum and maximum amount of water is smaller and bigger respectively or equal to the minimum/maximum water evaporated in both subintervals.

Lemma 1: For all $a, b, c \geq 0$, all constants > 0 and $t, u > 0$:

1) $\delta_{LOW}(a, b, u) \leq \delta_{HIGH}(a, b, u)$
2) $\delta_{LOW}(a, b, u) \geq \begin{cases} a^2/(2*\mathbf{U_2}) & \text{if } a < \mathbf{U_2} * u \\ a * u - \mathbf{U_2}*u^2/2 & \text{otherwise} \end{cases}$
3) $\delta_{LOW}(a, b, u) \geq \begin{cases} b^2/(2*\mathbf{U_1}) & \text{if } b < \mathbf{U_1} * u \\ b * u - \mathbf{U_1}*u^2/2 & \text{otherwise} \end{cases}$
4) $\delta_{LOW}(a, b, u) + \delta_{LOW}(b, c, t) \geq \delta_{LOW}(a, c, t + u)$
5) $(a + b)*u/2 \geq \delta_{LOW}(a, b, u)$
6) $\delta_{HIGH}(a, b, u) \leq (b * u + \mathbf{U_2}*u^2/2)$
7) $\delta_{HIGH}(a, b, u) + \delta_{HIGH}(b, c, t) \leq \delta_{HIGH}(a, c, u + t)$
8) $\delta_{HIGH}(a, b, u) \geq (a + b)*u/2$
9) $\delta_{HIGH}(a, b, u) \leq (a * u + \mathbf{U_1}*u^2/2)$

Proof: 1. - 9.: By calculus.

■

Based on this information, we can now model the steam boiler as a Timed Automaton.

4 The Boiler Model

For providing a formal description of the steam boiler, we first define all constants and the state. For all variables of the state, we provide the type, value range and description. Moreover, we describe the initial state which immediately forces the automaton to read the current sensor values and forwards them to the controller. The controller will provide an appropriate pump setting. The checks in the controller, which is described in the following section, require that there is a certain minimal amount of water between the critical limits or otherwise the controller would stop the steam boiler at once. Thus, a valid start condition of the water level and steam rate must be far enough from the critical boundaries not to force the controller to execute an emergency stop.

Constants

Name	Type	Restriction	Unit	Description
I	positive real	> ***S***	s	time in-between periodical sensor readings
S	positive real	< ***I***	s	delay to activate pumps after the last sensor reading
$\mathbf{U_1}$	positive real		l/s^2	maximum gradient of the increase of the steam rate
$\mathbf{U_2}$	positive real		l/s^2	maximum gradient of the decrease of the steam rate
$\mathbf{M_1}$	real	$\geq 0, < \mathbf{M_2}$	l	minimum amount of water before boiler becomes critical
$\mathbf{M_2}$	positive real	$\leq \mathbf{C}, > \mathbf{M_1}$	l	maximum amount of water before boiler becomes critical
W	positive real		l/s	maximum steam rate before boiler becomes critical
P	positive real		l/s	exact rate at which one active pump supplies water to the boiler
#pumps	positive integer			number of pumps that can supply water to the boiler in parallel
C	positive real	$\geq \mathbf{M_2}$	l	capacity of the boiler

Table 1: Constants and their relation for the boiler and controller models

Variables

Name	Initial Value	Type	Values Range	Unit	Description
now	*0*	real	$[0 \ldots \infty)$	s	current time
pr	*0*	integer	$\{0, \ldots \textbf{\#pumps}\}$		number of pumps actively supplying water to the boiler
q	$\gg M_1$, $\ll M_2$	real	$[0 \ldots C]$	l	actual water level in the boiler
v	*0*	real	$[0 \ldots \infty)$	l/s	steam rate of the steam currently leaving the boiler
pr_new	*0*	integer	$\{0, \ldots \textbf{\#pumps}\}$		number of pumps that are supposed to supply water after the activation delay
error	*0*	integer	$\{0, \ldots \textbf{\#pumps}\}$		number of pumps that fail to supply water to the boiler after activation
do_sensor	***true***	boolean	{***true***, ***false***}		enable a single sensor reading
set	***S***	real	$[0 \ldots \infty)$	s	next time the pumps change to the new settings
read	*0*	real	$[0 \ldots \infty)$	s	next time the sensors will be read
stop	***false***	boolean	{***true***, ***false***}		flag that determines whether emergency shut down is activated

Table 2: Variables of the steam boiler model. Together they represent the (initial) state of the steam boiler.

4.1 The Boiler Automaton

Expressing our interpretation of the informal specification more precisely leads to the following Timed Automaton:

Input Action

actuator (e_stop, pset)

Effect:

pr_new' = pset
stop' = e_stop
do_sensor' = ***true***
read' = now + ***I***

Output Action

sensor (s, w, p)

Precondition:

now = read
do_sensor = ***true***
stop = ***false***
w = q
s = v
p = pr

Effect:

do_sensor' = ***false***

Internal Actions

activate

Precondition:

now = set
stop = ***false***

Effect:

set' = read + ***S***
$0 \leq error' \leq pr\_new$
pr' = pr_new - error'

$\nu(\Delta t)$

Precondition:

stop = ***false***
$now + \Delta t \leq read$
$now + \Delta t \leq set$

Effect:

$v - \mathbf{U_2} * \Delta t \leq v' \leq v + \mathbf{U_1} * \Delta t$
$q + pr * \mathbf{P} * \Delta t - \delta_{HIGH}(v, v', \Delta t) \leq q'$
$q' \leq q + pr * \mathbf{P} * \Delta t - \delta_{LOW}(v, v', \Delta t)$
$now' = now + \Delta t$

This formal description of the steam boiler is easily readable: The steam boiler reads periodically the current water level and the current steam rate and forwards these values to the controller. In addition, the controller learns about the number of pumps that currently actually supply water to the boiler. The controller evaluates the data and through the actuator supplies a new pump setting or enables the shut-down phase. After the activation delay, all non-faulty pumps of the new setting supply water to the boiler. In the meantime, water evaporates into steam unpredictably but limited by its worst case rules.

With the ***actuator*** action the boiler receives the new pump setting requested by the controller and learns whether the controller shuts down the boiler. Furthermore, it schedules and enables the next reading of the sensor values. After an emergency stop is executed by setting the variable ***stop*** to ***true***, our model ignores any further development.

As an internal action, the boiler changes the steam rate and the water level unpredictably over time. The purpose of the ***time-passage action*** denoted with $\nu(\Delta t)$ is to provide a method for describing formally a time-dependent process. Δt represents an arbitrary, non-empty interval of time. A possible value for the parameter Δt depends on the precondition. Obviously, Δt may be arbitrary as long as the next activation of the pumps and the next sensor reading occur. Formally, the time-passage action must follow some rules as described in the Appendix A, which we are going to verify in the next section.

The ***activate action*** occurs after the pump activation delay. It sets the new pump rate with respect to an arbitrary number of pumps that fail, expressed as *error*. We chose this rather strong fault model where all pumps might fail at the activation time regardless whether such a pump was already supplying water before. This can be as much as all pumps that should supply water for the next cycle. Finally, it schedules the next activation time. Periodically, the ***sensor action*** forwards the current amount of water, the current steam rate and the number of active pumps to the controller. To prevent the sensor action from happening multiple times, it disables itself by setting *do_sensor* = ***false***.

4.2 Checking the Model

As described formally in Appendix A (the complete definition can be found in [Lyn91]), each Timed Automaton has to follow five axioms. We have to show that the Boiler Model satisfies these axioms. Overall, these axioms are used to define the concept of time in Timed Automata. The first three simply state that the current time denoted with the *now* variable starts at 0 in the initial state and only increases with the time-passage action. We would like to note that all non-time-passage actions occur "instantaneously". The fourth axiom enforces transitivity in the representation of time, i.e., transitivity of the time passage action. Whenever it is possible to describe a development over time with several succeeding time-passage steps it must be possible to describe this change in a single time-passage step. The fifth axiom describes trajectory consistency. Whenever the change from one state to another with the time-passage action can be expressed as a trajectory (or function), the change between any two states in this interval follows the same trajectory.

Basically, with these axioms fulfilled the Timed Automaton model allows us to combine automata through their input and output actions. We will combine the boiler model with a controller model, which we present in the next section. In the following, we show that our model fulfills these axioms. The first three are trivially true.

4.2.1 Axiom [A4]: Transitivity of $\nu(\Delta t)$

We have to show that if $(s, \nu(\Delta t_1), s')$ and $(s', \nu(\Delta t_2), s'')$ are steps (or transitions) then $(s, \nu(\Delta t_1+\Delta t_2), s'')$ is also a valid transition in our model.

Precondition (*read*, *set* and *stop* are unchanged): Since the time-passage action does not change *stop*, we know *stop = stop' = stop''* and the transitivity fulfilled. Moreover, $now + \Delta t_1 \leq read$ and $now' + \Delta t_2 \leq read'$. Since $now + \Delta t_1 + \Delta t_2 = now' + \Delta t_2$, we get $now + \Delta t_1 + \Delta t_2 \leq read''$. Analogously, we can show $now + \Delta t \leq set$ is transitive.

Effect:

a) Steam rate: We know $v - \boldsymbol{U_2}*\Delta t_1 \leq v' \leq v + \boldsymbol{U_1}*\Delta t_1$ and $v' - \boldsymbol{U_2}*\Delta t_2 \leq v'' \leq v' + \boldsymbol{U_1}*\Delta t_2$. Obviously, these statements can be combined to $v - \boldsymbol{U_2}*\Delta t_1 - \boldsymbol{U_2}*\Delta t_2 \leq v'' \leq v + \boldsymbol{U_1}*\Delta t_1 + \boldsymbol{U_1}*\Delta t_2$.

b) Water level lower bound: We know $q - \delta_{HIGH}(v, v', \Delta t_1) + pr*\Delta t_1 \leq q'$ and $q' - \delta_{HIGH}(v', v'', \Delta t_2) + pr*\Delta t_2 \leq q''$. These statements can be combined to $q'' \geq q - \delta_{HIGH}(v, v', \Delta t_1) + pr*(\Delta t_1+\Delta t_2) - \delta_{HIGH}(v', v'', \Delta t_2)$ and since (Lemma 1.7) $\delta_{HIGH}(a, b, u) + \delta_{HIGH}(b, c, t) \leq \delta_{HIGH}(a, c, u + t)$, we get $q - \delta_{HIGH}(v, v'', \Delta t_1+\Delta t_2) + pr*(\Delta t_1+\Delta t_2) \leq q''$.

c) Water level upper bound: We know $q' \leq q - \delta_{LOW}(v, v', \Delta t_1) + pr*\Delta t_1$ and $q'' \leq q' - \delta_{LOW}(v', v'', \Delta t_2) + pr*\Delta t_2$. Obviously, these statements can be combined to $q'' \leq q - \delta_{LOW}(v, v', \Delta t_1) + pr*(\Delta t_1+\Delta t_2) - \delta_{LOW}(v', v'', \Delta t_2)$.
Since $\delta_{LOW}(a, b, u) + \delta_{LOW}(b, c, t) \geq \delta_{LOW}(a, c, u + t)$ (Lemma 1.4) this is equivalent to $q'' \leq q - \delta_{LOW}(v, v'', \Delta t_1+\Delta t_2) + pr*(\Delta t_1+\Delta t_2)$.

d) Clock: From $now' = now + \Delta t_1$ and $now'' = now' + \Delta t_2$ follows $now'' = now + \Delta t_1+\Delta t_2$.

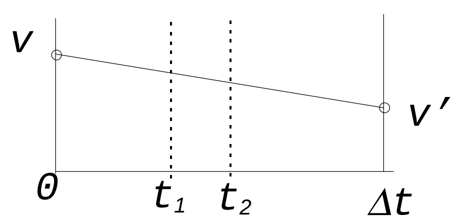
Thus, we have proved the transitivity of $\nu(\Delta t)$ for the boiler automaton.

■

4.2.2 Axiom [A5]: Trajectory Consistency of $\nu(\Delta t)$

We want to show that in-between any time-passage step the variables follow a trajectory.

We assume the time-passage action is enabled for the step $(s, \nu(\Delta t), s')$ and choose a (simple) trajectory $w(t)$ for which $w(0)= s$ and $w(\Delta t)= s'$ *for any* $t \in [0 \ldots \Delta t]$:



We define:

$$w(t) = \begin{cases} now_t = now + t \\ v_t = v + \dfrac{(v' - v)*t}{\Delta t} \\ q_t = q - \dfrac{(v + v_t)*t}{\Delta t} \\ \text{all other remain unchanged} \end{cases}$$

for any $t \in [0 \ldots \Delta t]$.

We have to show that our model is consistent for any t_1 and $t_2 \in [0 \dots \Delta t]$.

We define $\Delta t_t = t_2 - t_1$ and get:

a) Steam rate: We know $-U_2 * \Delta t \le v' - v \le U_1 * \Delta t$. Using the trajectory w(t) we know

$$- U_2 * \Delta t_t \le \frac{(v'-v) * \Delta t_t}{\Delta t} \le U_1 * \Delta t_t$$

and this is equivalent to

$$\frac{(v'-v) * t_1}{\Delta t} + v - U_2 * \Delta t_t \le \frac{(v'-v) * t_1}{\Delta t} + v + \frac{(v'-v) * \Delta t_t}{\Delta t} \le \frac{(v'-v) * t_1}{\Delta t} + v + U_1 * \Delta t_t$$

and a simple algebraic transformation and the trajectory definition lead to the desired result:
$v_{t_1} - U_2 * \Delta t_t \le v_{t_2} \le v_{t_1} + U_1 * \Delta t_t$.

b) Water level lower bound: Since we know $\delta_{HIGH}(a, b, u) \ge (a + b) * u/2$ (Lemma 1.8) we know

$$\delta_{HIGH}\left(v_{t_1}, v_{t_2}, \Delta t_t\right) \ge \frac{(v + v') * \Delta t}{2}$$

and this is equivalent to

$$q - \frac{\left(v + v_{t_1}\right) * t_1}{2} - \delta_{HIGH}\left(v_{t_1}, v_{t_2}, \Delta t_t\right) + pr * \left(\Delta t_t + t_1\right) \le q - \frac{\left(v + v_{t_2}\right) * \left(t_1 + \Delta t_t\right)}{2} + pr * \left(t_1 + \Delta t_t\right).$$

Since this is equivalent to $q_{t1} - \delta_{HIGH}(v_{t1}, v_{t2}, \Delta t_t) + pr * \Delta t_t \le q_{t2}$ we have proved the trajectory consistency of the lower bound of any new water level.

c) Water level upper bound is symmetrical to the lower bound and the proof is analogous to the previous case but uses Lemma 1.5 instead.

d) Time: $now = now$ this is equivalent to $now + t_2 = now + t_1 + (t_2 - t_1)$ and this to $now_{t2} = now_{t1} + \Delta t_t$.

Thus, we have proved the trajectory consistency for the time-passage action.

■

4.3 Properties of the Boiler

Based on the automaton description, we can derive the following useful information about the boiler system. These intermediate results can be favorably employed for fault detection and consistency checks in any actual boiler implementation based on this model. This information is expressed in the form of logic expressions invariant in all possible executions of this boiler model. Therefore, these expressions are called *invariants*. In other words, no order of steps will produce a state in which any of these logical expressions is not true. All proofs are by induction on the steps of the automaton.

For all following proofs, variables that do not change in a particular step will not be written differently in the pre-state and post-state. Such variables represent constants for the particular transition considered. For more clarification in the proofs, we usually give for each action all involved variables which do not change in parentheses.

The following simple proof shows that the next sensor reading and pumps activation time is always in the future.

Lemma 2: In all reachable states of boiler,

1) $read \geq now$
2) $set \geq now$

Proof.

1. For $a \in$ {sensor, activate} and in the initial state this lemma is trivially true. Otherwise we get for
   A) *a* = actuator (*now* unchanged): Sets *read'* = *now* + ***I***.
   B) *a* = time-passage (*read* is unchanged):
   We know $now' = now + \Delta t$ and from the precondition $now + \Delta t \leq read$. Thus, $now' \leq read$.
2. For $a \in$ {sensor, actuator} and in the initial state this lemma is trivially true. Otherwise we get for
   A) *a* = time-passage (*stopmode* and *read* are unchanged):
   We know $now' = now + \Delta t$ and from the precondition $now + \Delta t \leq set$. Thus, $now' \leq set$.
   B) *a* = activate (*now* and *read* unchanged):
   We know $read \geq now$ from Lemma 2.1 and $set' = read + \boldsymbol{S}$ from the effect thus this lemma is true.

■

5 The Controller Model

In order to solve the steam boiler problem, we have to find a controller that guarantees the required safety properties. For this purpose, we take advantage of a characteristic of the Timed Automaton model. First, we will show that a simple controller that cannot tolerate sensor faults guarantees the safety properties under described assumptions. Then, the Simulation Mapping technique is used to show incrementally that a different controller which allows for sensor failures preserves the safety properties.

Obviously, it is most important that the controller identifies water levels and steam rates that might cross their critical limits before the next sensor values arrive. In case such sensor values are identified the controller will enable the shut-down phase. In a non-critical case, the controller chooses an appropriate new setting for the pumps to adjust the water level and compensate for the amount of steam leaving the boiler.

5.1 The Controller Model

Definitions

Name	Type	Unit	Value	Description
max_pumps_after_set	integer		***#pumps***	maximum number of pumps that can supply water to the boiler after the delay considering the pump failure model
min_pumps_after_set	integer		*0*	minimum number of pumps that can supply water to the boiler after the delay considering the chosen pump failure model. For a different pump failure model, e.g., in which pumps might fail when activated or stopped, this constant may actually be a function of the change in the number of pumps.
min_steam_water(sr)	real	l	$sr^2/(2\,\boldsymbol{U_2})$ if $sr < \boldsymbol{I} * \boldsymbol{U_2}$ $(sr - \boldsymbol{U_2} * \boldsymbol{I}/2) * \boldsymbol{I}$ otherwise	minimum amount of water that can evaporate into steam until the next sensor reading
max_steam_water(sr)	real	l	$(sr + \boldsymbol{U_1} * \boldsymbol{I}/2) * \boldsymbol{I}$	maximum amount of water that can evaporate into steam until the next sensor reading
min_steam_water_est(sr)	real	l	$sr^2/(2\,\boldsymbol{U_1})$ if $sr < \boldsymbol{I} * \boldsymbol{U_1}$ $(sr - \boldsymbol{U_1} * \boldsymbol{I}/2) * \boldsymbol{I}$ otherwise	estimated minimum amount of water that has evaporated since the next sensor reading

Table 3: Definitions and abbreviations for the controller model

Variables

Name	Initial Value	Type	Value Range	Unit	Description
do_output	***false***	boolean	{***true, false***}		flag that enables the output. This represents a kind of program counter.
stopmode	***true***	boolean	{***true, false***}		flag to activate the shut down, initially true, since condition is not checked yet.
wl	*q*	real	[*0* ... ***C***]	l	current water level reading
sr	*0*	real	[*0* ... ***W***]	l/s	current steam rate reading
now	*0*	real	[*0* ... ∞)	s	current time
pumps	*0*	integer	{*0* ... ***#pumps***}		number of currently active pumps supplying water to the boiler
px	*0*	integer	{*0* ... ***#pumps***}		number of pumps that shell supply water next

Table 4: The state of the controller including all variables and their initial values

5.2 The Simple Controller Automaton

The input and output actions are complementary to the input and output actions of the steam boiler model.

Input Actions

sensor (s, w, p)

Effect:

sr' = s
wl' = w
pumps' = p
do_output' = ***true***

 # safety checks:
if sr' $\geq$ ***W*** - $\boldsymbol{U_1}$ * ***I*** *or*
wl' $\geq$ $\boldsymbol{M_2}$ - ***P*** * *(pumps'* * ***S*** + *(max_pumps_after_set)*
* *(****I*** - ***S****)) + min_steam_water(sr) or*
wl' $\leq$ $\boldsymbol{M_1}$ - ***P*** * *(pumps'* * ***S*** + *(min_pumps_after_set)*
* *(****I*** - ***S****)) + max_steam_water(sr)*
then
stopmode' = ***true***
else
stopmode' = {***true, false***} *arbitrary*

Internal Actions

controller

Precondition:

true

Effect:

$0 \leq px' \leq$ ***#pumps***

ν(Δt)

Precondition:

true

Effect:

now' = now + Δt

Output Actions

actuator (e_stop, pset)

Precondition:

do_output = ***true***
pset = px
e_stop = stopmode

Effect:

do_output' = ***false***

With the ***sensor action***, the controller receives periodically the current steam rate, water level and number of activated pumps. Its primary purpose is to test if the current sensor values are "close" to either critical limit. In such a case the sensor action sets a flag for the actuator to initiate the shut-down. Likewise, external critical conditions are modeled by non-deterministically setting *stopmode* to true. Furthermore, the sensor action enables the actuator action. The test for what is "close" depends on the particular fault model used and controller capabilities. The controller can try to start all pumps every period and our fault model allows up to all pumps to fail. The point in time for the decision how many pumps actually supply water to the boiler is every *set* time. Therefore, we must choose all pumps for *max_pumps_after_set*. On the other hand, all pumps could fail and therefore *min_pumps_after_set* equals 0. Similarly, *min_steam_water* and *max_steam_water* express the minimum and maximum amounts of water that can evaporate into steam in the following period starting with given current steam rate, respectively. The test simply calculates the worst case situations for the water level and steam rate and compares the results with the critical limits $\boldsymbol{M_1}$, $\boldsymbol{M_2}$ and ***W***.

The ***controller action*** chooses an appropriate new pump setting. Actually, it can choose any pump setting. For our approach, we are not particularly interested in the performance of the controller. On the other hand, we are interested in generality. Therefore, we chose a controller model that can incorporate any possible control algorithm for setting the pumps. As a consequence, our results concerning the safety are valid for an arbitrary control algorithm. Although the choice of a new setting for the pumps is irrelevant to the safety of the steam boiler system, for a performance analysis the pump setting would be of major importance. The ***time-passage action*** ($\nu(\Delta t)$) allows time to pass. For the following proofs, we ignore these two actions, since they do not provide additional information and are irrelevant to the proofs.

Finally, the ***actuator action*** forwards the new pump setting and whether the boiler must be stopped to the boiler environment. Furthermore, it disables itself, by setting *do_output* back to false.

As suggested in the original specification, this controller model acts instantaneously. Therefore, the time-passage action is trivial and all five axioms for Timed Automata are satisfied. Moreover, there is no useful information gained from the controller model alone. So far the proofs have involved only either the steam boiler model or the controller model. Next, we use the composition property of Timed Automata for combining the two automata, and we prove the required safety properties.

6 Properties of the Combined Steam Boiler System

Following, we show in several steps that the combined model (formally a composition), consisting of the steam boiler model and the simple controller model together, guarantee the safety conditions. The first safety property requires that the steam rate must always stay below ***W***. Before the steam rate can cross this limit, the boiler must be shut down. Expressing this in terms of the state of the steam boiler system, we have to show

S1) $v < \boldsymbol{W}$ *or stop* = ***true***

The second safety property requires that the water level must always stay between its critical limits $\boldsymbol{M_1}$ and $\boldsymbol{M_2}$. Before the water level can cross either limit, the boiler must be stopped. Thus, we have to show

S2) $\boldsymbol{M_1} < q < \boldsymbol{M_2}$ *or stop* = ***true***

6.1 Combined Steam Boiler System Automaton

Following, we present the composed steam boiler + controller automaton. This should clarify the interaction between the different actions and make it easier to follow the proofs.

actuator (e_stop, pset)

Precondition:

do_output = ***true***
pset = *px*
e_stop = *stopmode*

Effect:

do_output' = ***false***
do_sensor' = ***true***
pr_new' = *pset*
stop' = *e_stop*
read' = *now* + ***I***

controller

Precondition:

true

Effect:

$0 \leq px' \leq$ ***#pumps***

ν(Δt)

Precondition:

stop = ***false***
$now + \Delta t \leq read$
$now + \Delta t \leq set$

Effect:

$v - \mathbf{U_2} * \Delta t \leq v' \leq v + \mathbf{U_1} * \Delta t$
$q + pr * \mathbf{P} * \Delta t - \delta_{HIGH}(v, v', \Delta t) \leq q'$
$q' \leq q + pr * \mathbf{P} * \Delta t - \delta_{LOW}(v, v', \Delta t)$
$now' = now + \Delta t$

sensor (s, w, p)

Precondition:

now = *read*
do_sensor = ***true***
stop = ***false***
w = *q*
s = *v*
p = *pr*

Effect:

pumps' = *p*
do_sensor' = ***false***
do_output' = ***true***
sr' = *s*
wl' = *w*

if $sr' \geq \mathbf{W} - \mathbf{U_1} * \mathbf{I}$ *or*
$wl' \geq \mathbf{M_2} - \mathbf{P} * (pumps' * \mathbf{S} + (max\_pumps\_after\_set) * (\mathbf{I} - \mathbf{S})) + min\_steam\_water(sr)$ *or*
$wl' \leq \mathbf{M_1} - \mathbf{P} * (pumps' * \mathbf{S} + (min\_pumps\_after\_set) * (\mathbf{I} - \mathbf{S})) + max\_steam\_water(sr)$
then stopmode' = ***true***
else stopmode' = {***true, false***} *arbitrary*

activate

Precondition:

now = *set*
stop = ***false***

Effect:

set' = *read* + ***S***
$0 \leq error' \leq pr\_new$
pr' = *pr_new* - *error'*

6.2 Steam Boiler System Properties

The following lemmas lead us step-by-step toward proving the safety conditions. Coming up with the right invariants that lead to showing the safety properties is the most complicated task in working with Timed Automata. On the other hand, the proofs themselves are usually straightforward and follow well-established, stylized methods and the usual pattern for proving by induction. The main work for proving the safety properties is done by means of these invariants. All the proofs for our model are by induction on the model and can easily be verified using current mechanical proof technology.

The following lemma describes the conditions when the controller decides that the boiler needs to be emergency-stopped.

Lemma 3: In all reachable states of the controller model,

1) $\boldsymbol{M_2} > wl + \boldsymbol{P} * (pumps * \boldsymbol{S} + \boldsymbol{\#pumps} * (\boldsymbol{I} - \boldsymbol{S})) - min\_steam\_water(sr)$ *or stopmode* = ***true***
2) $\boldsymbol{M_1} < wl + \boldsymbol{P} * pumps * \boldsymbol{S} - (sr * \boldsymbol{I} + \boldsymbol{U_1} * \boldsymbol{I}^2/2)$ *or stopmode* = ***true***
3) $sr + \boldsymbol{U_1} * \boldsymbol{I} < \boldsymbol{W}$ *or stopmode* = ***true***

Proof. All three statements are true in the initial state and the correctness of the induction step follows directly from the sensor action which is the only action changing any of the variables.

■

The following lemma states the controller's knowledge about the current situation in the environment after reading the sensors.

Lemma 4: In all reachable states of the combined steam boiler system,
if do_output then now = read and sr = v and wl = q

Proof. We distinguish on the cases for the action *a*. In the initial state this lemma is true. For $a \in$ {actuator, activate} this lemma is trivially true. For

A) *a* = sensor (*now* and *read* are unchanged):
From the precondition we know *now* = *read* and from the effect *do_output'* = ***true***, *sr'* = *v* and *wl'* = *q*. Thus, this lemma is true for the sensor action.

B) *a* = time-passage (*do_output, sr, wl* and *read* are not changed):
We know from the precondition that $\Delta t \leq read - now$ and $\Delta t > 0$ per definition, we know $now \neq read$. It remains *do_output* = ***false***. Since *do_output* is not changed this lemma is fulfilled.

■

Lemma 5 concludes that the next time the pumps will be activated can only be either the constant delay after or before the next sensor reading.

Lemma 5: In all reachable states of the combined steam boiler system,
$set = read + \boldsymbol{S}$ *or* $set = read - \boldsymbol{I} + \boldsymbol{S}$

Proof. In the initial state this lemma is true. This lemma is trivially true for $a \in$ {sensor, time-passage, activate}. For *a* = actuator (*set* is unchanged) we know from the precondition *do_output* = ***true*** and *if do_output then now* = *read* (Lemma 4). We get two cases:
Case 1) We assume $set = read - \boldsymbol{I} + \boldsymbol{S}$ in the precondition. From the effect we get $read' = now + \boldsymbol{I}$ from which we can infer $set = read' + \boldsymbol{S}$ and this case is true.
Case 2) We can assume $set = read + \boldsymbol{S}$ in the pre-state. This assumption contradicts *now* = *read* and $now \leq read$ and $now \leq set$ (Lemma 2). Thus, this lemma is true.

■

This lemma helps us later to show that whenever the sensors are read (or, at the same instant, the new pumps settings sent to the boiler) the pumps are activated exactly after the delay ***S***, as specified.

Lemma 6: In all reachable states of the combined steam boiler system,
$now \leq read - \boldsymbol{I} + \boldsymbol{S}$ *or* $set = read + \boldsymbol{S}$

Proof. We distinguish on the cases for the action *a*. In the initial state and for $a \in$ {sensor, activate} this lemma is trivially true.

A) *a* = actuator (*now* is unchanged):
We know from the precondition *do_output* = ***true*** and from Lemma 4 *if do_output then now = read*. From the effect we know $read' = now + \boldsymbol{I}$ which implies $now \leq read' - \boldsymbol{I} + \boldsymbol{S}$ and this lemma is obviously satisfied.

B) *a* = time-passage (*read* and *set* unchanged):
In case $set = read + \boldsymbol{S}$ obviously this lemma is true. Otherwise, we get from the precondition $now + \Delta t \leq set$, from Lemma 5 $set = read + \boldsymbol{S}$ *or* $set = read - \boldsymbol{I} + \boldsymbol{S}$ and we can conclude $set = read - \boldsymbol{I} + \boldsymbol{S}$ and $now + \Delta t \leq read - \boldsymbol{I} + \boldsymbol{S}$. Since $now' = now + \Delta t$ from the effect this lemma is true.

■

The following lemma claims that as long as the sensor reading time is not reached, the output of a new pump setting is disabled.

Lemma 7: In all reachable states of the combined steam boiler system,
if $now < read$ *then do_output* = ***false***

Proof. We distinguish on the cases for the action *a*. In the initial state this lemma is true. This lemma is trivially true for $a \in$ {sensor, actuator, activate}. For *a* = time-passage we get from the precondition $now + \Delta t \leq read$ and $\Delta t > 0$ per definition. We know *do_output* = ***false*** which is not changed by the effect.

■

The following is a base for Lemma 10. Lemma 10 expresses that at the time the sensors are read the representation of the active pumps in the controller are equal to the pumps actually supplying water to the boiler. This lemma is partially redundant but yields some new knowledge.

Lemma 8: In all reachable states of the combined steam boiler system,
if do_output then pumps = pr and now = read

Proof. We distinguish on the cases for the action *a*. In the initial state and for $a \in$ {sensor, actuator} this lemma is trivially true.

A) *a* = time-passage (*do_output*, *set*, *pumps*, *pr* and *read* are not changed):
We know from the precondition that $\Delta t \leq read - now$ and from Lemma 7 *if* $now < read$ *then do_output* = ***false***, besides $\Delta t > 0$ per definition. Thus, we can conclude *do_output* = ***false***. Since *do_output* is not changed this lemma is fulfilled.

B) *a* = activate (*do_output*, *now* and *pumps* are unchanged):
We know from the precondition $now = set$ and from Lemma 5 $set = read + \boldsymbol{S}$ *or* $set = read - \boldsymbol{I} + \boldsymbol{S}$. Since $now \leq read$ (Lemma 2), we know $now = set = read - \boldsymbol{I} + \boldsymbol{S}$. From Lemma 7 we know *if* $now < read$ *then do_output* = ***false*** for the precondition. Thus *do_output* = ***false*** and remains false and this lemma is fulfilled.

■

In general, either the controller wants to read some values or send some new parameter to the boiler system. We need this information only for the next lemma.

Lemma 9: In all reachable states of the controller model,
do_sensor xor do_output

Proof. In the start condition this Lemma is true. We distinguish on the cases for the action *a*. For $a \in$ {time-passage, activate} this lemma is trivially true. For

A) *a* = sensor: We get from the effect *do_sensor'* = ***false*** and *do_output'* = ***true***.

B) *a* = actuator: We get from the effect *do_sensor'* = ***true*** and *do_output'* = ***false***. Thus this lemma is true.

■

During the entire operation of the boiler system the number of pumps supplying water is either the number requested by the controller minus some faulty pumps or equal to the status sensed at the last reading point after the pumps were activated.

Lemma 10: In all reachable states of the combined steam boiler system,
if set = read + ***S*** *and do_output =* ***false*** *then pr = pr_new - error else pr = pumps*

Proof. We distinguish on the cases for the action *a*. In the initial state and for *a* = time-passage this lemma is trivially true.

A) *a* = sensor (*set*, *read*, *pr* and *pr_new* are unchanged):
We know *do_sensor* = ***true*** from the precondition, *do_output xor do_sensor* (Lemma 9). Thus *do_output* = ***false***. Moreover, we know *now* = *read* from the precondition and from Lemma 6 *now* ≤ *read* - ***I*** + ***S*** *or set* = *read* + ***S***. Since ***I*** > ***S*** per definition, it must be *set* = *read* + ***S***. From the effect we know *do_output'* = ***true*** and *if do_output then pumps* = *pr* (Lemma 8) which is true for the post-state. Thus, *if set* = *read* + ***S*** *and do_output'* = ***false*** *then pr* = *pr_new* - *error else pr* = *pumps'* is true for the sensor action.

B) *a* = actuator (*set*, *pr* and *pumps* are unchanged):
We know from the precondition *do_output* = ***true*** . Thus, *pr* = *pumps* from the assumption and from Lemma 8 *if do_output then pumps* = *pr and now* = *read* . Since we know *now* ≤ *read* - ***I*** + ***S*** *or set* = *read* + ***S*** from Lemma 5 and ***I*** > ***S*** per definition, it must be *set* = *read* + ***S***. From the effect we know *read'* = *now* + ***I*** and thus *set* = *read'* - ***I*** + ***S*** and this lemma is true.

C) *a* = activate (*pumps*, *do_output* and *pr_new* are unchanged):
We know from the precondition *now* = *set* and from Lemma 5 *set* = *read* + ***S*** *or set* = *read* - ***I*** + ***S***. Since *now* ≤ *read* (Lemma 2), we know *now* = *set* = *read* - ***I*** + ***S***. From Lemma 7 we know *if now* < *read then do_output* = ***false*** for the precondition and remains false. From the effect we get *set'* = *read* + ***S*** and *pr'* = *pr_new* - *error*. Thus, this lemma is fulfilled.

■

Using the test conditions in Lemma 5, we can now prove that the actual steam rate will stay under a certain limit depending on how long it takes until the next sensor reading.

Lemma 11: In all reachable states of the combined steam boiler system,

$v + \boldsymbol{U_1}*(read - now) < \boldsymbol{W}$ *or stop* = ***true***

Proof. The basis is vacuously satisfied. We distinguish on the cases for the action *a*. For $a \in$ {sensor, activate} this lemma is trivially true. Otherwise we get:

A) *a* = actuator (*v*, *stop* and *now* are unchanged):

We know $sr + \boldsymbol{U_1}*\boldsymbol{I} < \boldsymbol{W}$ *or stopmode* = ***true*** (Lemma 3.3), *do_output* = ***true*** from the precondition and *if do_output then now = read and sr = v* (Lemma 4). From this we can infer $v + \boldsymbol{U_1}*(now + \boldsymbol{I} - now) < \boldsymbol{W}$ *or stopmode* = ***true***. Moreover, we get *stop' = e_stop = stopmode* and $read' = now + \boldsymbol{I}$ from the effect and thus, we know $v + \boldsymbol{U_1}*(read' - now) < \boldsymbol{W}$ *or stop'* = ***true***.

B) *a* = time-passage (*read* and *stop* are unchanged):

We know from the precondition *stop* = ***false*** and $v + \boldsymbol{U_1}*(read - now) < \boldsymbol{W}$ from the assumption. This is equivalent to $v + \boldsymbol{U_1}*(read - now - \Delta t + \Delta t) < \boldsymbol{W}$ and it follows $v + \boldsymbol{U_1}*\Delta t + \boldsymbol{U_1} * (read - now - \Delta t) < \boldsymbol{W}$. Since we know from the effect $v' \leq v + \boldsymbol{U_1} * \Delta t$ and $now' = now + \Delta t$, finally, this is equivalent to $v' + \boldsymbol{U_1}*(read - now') < \boldsymbol{W}$.

■

The following lemma describes the amount of water remaining above the lower limit depending on the current steam rate and minimum pump rate.

Lemma 12: In all reachable states of the combined steam boiler system,

if do_output = ***false*** *then*

if $set = read - \boldsymbol{I} + \boldsymbol{S}$ *then*

$\boldsymbol{M_1} < q + \boldsymbol{P}*pumps*(set-now) - (v * (read-now) + \boldsymbol{U_1}*(read-now)^2/2)$ *or stop* = ***true***

else $\boldsymbol{M_1} < q - (v * (read-now) + \boldsymbol{U_1}*(read-now)^2/2)$ *or stop* = ***true***

Proof. In the initial state this Lemma is true. We distinguish on the cases for the action *a*: For the sensor action this lemma is trivially true.

A) *a* = actuator (*set*, *q*, *v*, *pumps* and *now* are unchanged):

We know $\boldsymbol{M_1} < wl + \boldsymbol{P}*pumps*\boldsymbol{S} - (sr * \boldsymbol{I} + \boldsymbol{U_1}*\boldsymbol{I}^2/2)$ *or stopmode* = ***true*** (Lemma 3.2) and Lemma 4: *if do_output then now = read and sr = v and wl = q*. Since *do_output* = ***true*** in the precondition, we know *now = read*, *sr = v* and *wl = q*. Since $now \leq read - \boldsymbol{I} + \boldsymbol{S}$ *or* $set = read + \boldsymbol{S}$ (Lemma 6), $now \leq read$ (Lemma 2), we know $set = read + \boldsymbol{S}$ and, since $read' = now + \boldsymbol{I}$ from the effect, $set = read' - \boldsymbol{I} + \boldsymbol{S}$. Moreover, we know *stop' = e_stop = stopmode* from the effect and thus, $\boldsymbol{M_1} < q + \boldsymbol{P}*pumps*(set-now) - (v * (read'- now) + \boldsymbol{U_1}*(read'-now)^2/2)$ *or stop'* = ***true***. Actuator sets *do_output'* = ***false*** and this lemma is true for the actuator action.

B) *a* = time-passage (*do_output*, *set*, *read*, *stop* and *pumps* are unchanged):

We know *do_output* = ***false*** from *if now < read then do_output* = ***false*** (Lemma 7), from the precondition ($now + \Delta t \leq read$) and $\Delta t > 0$.

Based on $set = read + \boldsymbol{S}$ *or* $set = read - \boldsymbol{I} + \boldsymbol{S}$ (Lemma 5), we can distinguish two cases:

1. Case $set = read - \boldsymbol{I} + \boldsymbol{S}$:

   We know from the assumption $\boldsymbol{M_1} < q + \boldsymbol{P}*pumps*(set-now-\Delta t+\Delta t) - (v * (read-now-\Delta t+\Delta t) + \boldsymbol{U_1}*(read-now-\Delta t+\Delta t)^2/2)$ *or stop* = ***true***. This is equivalent to $\boldsymbol{M_1} < q + \boldsymbol{P}*pumps*\Delta t - (v*\Delta t +$

$\boldsymbol{U_1}*\Delta t^2/2)$ + $\boldsymbol{P}$*pumps*(set-now-Δt) - (v * (read-now-Δt) + $\boldsymbol{U_1}$*Δt *(read-now-Δt) + $\boldsymbol{U_1}$*(read-now-Δt)^2/2) *or stopmode* = ***true***. Since *v*(read-now-Δt)* + $\boldsymbol{U_1}$*Δt*(read-now-Δt) = (v + $\boldsymbol{U_1}$*Δt) * (read-now-Δt) and *now' = now +Δt, v' ≤ v +* $\boldsymbol{U_1}$ * Δ*t* from the effect, we get $\boldsymbol{M_1}$ < *q +* $\boldsymbol{P}$*pumps*Δt - (v*Δt + $\boldsymbol{U_1}$*Δt^2/2)+ $\boldsymbol{P}$*pumps*(set-now') - (v' * (read-now') + $\boldsymbol{U_1}$*(read-now')^2/2) *or stop* = ***true***. Since δ_{HIGH}*(a, b, u) ≤ (a*u +* $\boldsymbol{U_1}$*u^2/2) from Lemma 1.9, *pumps = pr* from Lemma 10: *if set = read +* ***S*** *and do_output =* ***false*** *then pr = pr_new - error else pr = pumps* and *q + pr* * $\boldsymbol{P}$ * Δ*t* - δ_{HIGH}*(v, v', Δt) ≤ q'* from the effect, we get $\boldsymbol{M_1}$ < *q' +* $\boldsymbol{P}$*pumps*(set-now') - (v * (read-now') + $\boldsymbol{U_1}$*(read-now')^2/2) *or stop* = ***true*** and this case true.

2. Case *set = read +* ***S***:

   We know from the assumption $\boldsymbol{M_1}$ < *q - (v * (read-now-Δt+Δt) +* $\boldsymbol{U_1}$*(read-now-Δt+Δt)^2/2) *or stop* = ***true***. This is equivalent to $\boldsymbol{M_1}$ < *q - (v*Δt +* $\boldsymbol{U_1}$*Δt^2/2) - (v * (read-now-Δt) + $\boldsymbol{U_1}$*Δt *(read-now-Δt) + $\boldsymbol{U_1}$*(read-now-Δt)^2/2) *or stop* = ***true***. Since *v * (read-now-Δt) +* $\boldsymbol{U_1}$*Δt *(read-now-Δt) = (v + $\boldsymbol{U_1}$*Δt)*(read-now-Δt) and *now' = now +Δt, v' ≤ v +* $\boldsymbol{U_1}$ * Δ*t* from the effect, we get $\boldsymbol{M_1}$ < *q - (v*Δt +* $\boldsymbol{U_1}$*Δt^2/2) - (v' * (read-now') + $\boldsymbol{U_1}$*(read-now')^2/2) *or stop* = ***true***. Since δ_{HIGH}*(a, b, u) ≤ (a*u +* $\boldsymbol{U_1}$*u^2/2) from Lemma 1.9, *0 ≤ pr ** $\boldsymbol{P}$ * Δ*t* and *q + pr ** $\boldsymbol{P}$ * Δ*t* - δ_{HIGH}*(v, v', Δt) ≤ q'* from the effect, we get $\boldsymbol{M_1}$ < *q' - (v * (read-now') +* $\boldsymbol{U_1}$*(read-now')^2/2) *or stop* = ***true*** and this case true.

C) *a* = activate (only *set* is changed):
If *do_output* = ***true*** this lemma is trivially true. Since we get *set = now* from the precondition, *now ≤ read* (Lemma 2) and *set = read +* ***S*** *or set = read -* ***I*** *+* ***S*** (Lemma 5), we know *set = read -* ***I*** *+* ***S*** and we get from the assumption $\boldsymbol{M_1}$ < *q - (v * (read-now) +* $\boldsymbol{U_1}$*(read-now)^2/2) *or stop* = ***true***. Since the effect sets *set' = read +* ***S*** this lemma is true.

■

The following lemma describes the amount of water remaining to the upper water level limit depending on the current steam rate and the maximum pump rate.

Lemma 13: In all reachable states of the combined steam boiler system

if do_output = ***false*** *then*

if set = read - ***I*** *+* ***S*** *then*

$\boldsymbol{M_2}$ > *q +* $\boldsymbol{P}$*(pumps*(set-now) + ***#pumps****(***I-S***)) - steam or stop* = ***true***

else $\boldsymbol{M_2}$ > *q +* $\boldsymbol{P}$****#pumps****(read - now) - steam or stop* = ***true***

$$\text{with } steam = \begin{cases} v^2/2*\boldsymbol{U_2} & \text{if } v < \boldsymbol{U_2}(read\text{-}now) \\ (v*(read\text{-}now) - \boldsymbol{U_2}*(read\text{-}now)^2/2) & \text{otherwise} \end{cases}$$

Proof. In the initial state this Lemma is true. We distinguish on the cases for the action *a*: For *a* = sensor this lemma is trivially true.

A) *a* = actuator (*set*, *q*, *v*, *pumps* and *now* are unchanged):

We know $\boldsymbol{M_2} > wl + \boldsymbol{P}*(pumps*\boldsymbol{S} + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - min\_steam\_water(sr)$ *or stopmode* = ***true***

with $min\_steam\_water(sr) = \begin{cases} sr^2/(2*\boldsymbol{U_2}) & \text{if } sr < \boldsymbol{U_2}*\boldsymbol{I} \\ (sr*\boldsymbol{I} - \boldsymbol{U_2}*\boldsymbol{I}^2/2) & \text{otherwise} \end{cases}$

(Lemma 3.1) and Lemma 4: *if do_output then now = read and sr = v and wl = q*. Since *output* = ***true*** in the precondition, we know *now = read*, *sr = v* and *wl = q*. Since *now ≤ read* - ***I*** + ***S*** *or set = read* + ***S*** (Lemma 6), *now ≤ read* (Lemma 2), we know *set = read* + ***S*** and, since *read' = now* + ***I*** from the effect, *set = read'* - ***I*** +***S***. Since *stop' = e_stop = stopmode* from the effect, we know $\boldsymbol{M_2} > q + \boldsymbol{P}*(pumps*(set - now) + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - min\_steam\_water(v)$ *or stop'* = ***true*** with

$min\_steam\_water(sr) = \begin{cases} v^2/2*\boldsymbol{U_2} & \text{if } v < \boldsymbol{U_2}*(read'\text{-}now) \\ (v*(read'\text{-}now) - \boldsymbol{U_2}*(read'\text{-}now)^2/2) & \text{otherwise} \end{cases}$

The actuator action sets *do_output'* = ***false*** and this lemma is true for the actuator action.

B) *a* = time-passage (*do_output*, *set*, *read*, *stop* and *pumps* are unchanged):

We know *do_output* = ***false*** from (Lemma 7) *if now < read then do_output* = ***false***, from the precondition ($now + \Delta t \leq read$) and $\Delta t > 0$. Since we know *set = read* + ***S*** *or set = read* - ***I*** + ***S*** (Lemma 5), we can distinguish two cases:

a. Case *set = read* - ***I*** + ***S***:

We know from the assumption $\boldsymbol{M_2} > q + \boldsymbol{P}*(pumps*(set\text{-}now\text{-}\Delta t+\Delta t) + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - steam$ *or stop* = ***true*** which is equivalent to $\boldsymbol{M_2} > q + \boldsymbol{P} * pumps*\Delta t - \delta_{LOW}(v, v', \Delta t) + \boldsymbol{P}*(pumps*(set\text{-}now\text{-}\Delta t) + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - steam + \delta_{LOW}(v, v', \Delta t)$ *or stop* = ***true***. Moreover, we know from the effect that $now' = now + \Delta t$, $q + \boldsymbol{P} * pr * \Delta t - \delta_{LOW}(v, v', \Delta t) \geq q'$, and *pumps = pr* from Lemma 10: *if set = read* + ***S*** *and do_output* = ***false*** *then pr = pr_new - error else pr = pumps*. Thus, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*(pumps*(set\text{-}now') + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - steam + \delta_{LOW}(v, v', \Delta t)$ *or stop* = ***true*** with

$steam = \begin{cases} v^2/2*\boldsymbol{U_2} & \text{if } v < \boldsymbol{U_2}*(read\text{-}now) \\ v(read\text{-}now' + \Delta t) - \boldsymbol{U_2}*(read\text{-}now' + \Delta t)^2/2) & \text{otherwise} \end{cases}$

Based on the steam rate condition and Lemma 1.2:

$\delta_{LOW}(a, b, u) \geq \begin{cases} a^2/(2*\boldsymbol{U_2}) & \text{if } a < \boldsymbol{U_2} * u \\ a * u - \boldsymbol{U_2}*u^2/2 & \text{otherwise} \end{cases}$

we distinguish following cases:

1. Sub-case $v < \boldsymbol{U_2}(read\text{-}now)$ and $v < \boldsymbol{U_2} * \Delta t$:

   Since $\delta_{LOW}(v, v', \Delta t) \geq v^2/(2*\boldsymbol{U_2})$ and $v'^2/2*\boldsymbol{U_2} > 0$, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*(pumps*(set\text{-}now') + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - v'^2/(2*\boldsymbol{U_2})$ *or stop* = ***true*** and this case true.

2. Sub-case $v < \boldsymbol{U_2}(read\text{-}now)$ and $v \geq \boldsymbol{U_2} * \Delta t$:

   Here, we know $\boldsymbol{M_2} > q' + \boldsymbol{P}*(pumps*(set\text{-}now') + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - v^2/(2*\boldsymbol{U_2}) + (v *\Delta t - \boldsymbol{U_2}*\Delta t^2/2)$ *or stop* = ***true*** and since $v^2/(2*\boldsymbol{U_2}) - (v *\Delta t - \boldsymbol{U_2}*\Delta t^2/2) = (v - \boldsymbol{U_2}*\Delta t)^2/2*\boldsymbol{U_2}$ and $v - \boldsymbol{U_2}*\Delta t \leq v'$, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*(pumps*(set\text{-}now') + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - v'^2 /2*\boldsymbol{U_2}$ *or stop* = ***true*** and this case true.

3. Sub-case $v \geq \boldsymbol{U_2}(read\text{-}now)$:

   Since $now + \Delta t \leq read$ from the precondition, we know $v \geq \boldsymbol{U_2}*\Delta t$ and using Lemma 1.2, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*(pumps*(set\text{-}now') + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - v*\Delta t - (v*(read\text{-}now') - \boldsymbol{U_2}*\Delta t*(read\text{-}now') - \boldsymbol{U_2}*(read\text{-}now')^2/2) + \boldsymbol{U_2}*\Delta t^2/2 + (v * \Delta t - \boldsymbol{U_2}*\Delta t^2/2)$ *or stop* = ***true***. Since $v*(read\text{-}now') - \boldsymbol{U_2}*\Delta t*(read\text{-}now') = (v - \boldsymbol{U_2}*\Delta t)*(read\text{-}now') - \boldsymbol{U_2}*\Delta t*(read\text{-}now')$ and $v - \boldsymbol{U_2}*\Delta t \leq v'$ from the effect, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*(pumps*(set\text{-}now') + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - (v'*(read\text{-}now') - \boldsymbol{U_2}*(read\text{-}now')^2/2)$ *or stop* = ***true***.

   This case is obviously true.

b. Case $set = read + \boldsymbol{S}$:

   Since $\boldsymbol{\#pumps} \geq pr$ per definition, we know from the assumption $\boldsymbol{M_2} > q + \boldsymbol{P}*pr*\Delta t - \delta_{LOW}(v, v', \Delta t) + \boldsymbol{P}*\boldsymbol{\#pumps}*(read - now - \Delta t) - steam + \delta_{LOW}(v, v', \Delta t)$ *or stop* = ***true*** with

$$steam = \begin{cases} v^2 /2*\boldsymbol{U_2} & \text{if } v < \boldsymbol{U_2}*(read\text{-}now) \\ (read\text{-}now\text{-}\Delta t + \Delta t) - \boldsymbol{U_2}*(read\text{-}now\text{-}\Delta t + \Delta t)^2/2) & \text{otherwise} \end{cases}$$

   Moreover, we know from the effect that $now' = now + \Delta t$, $q + \boldsymbol{P} * pr * \Delta t - \delta_{LOW}(v, v', \Delta t) \geq q'$. Thus, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*\boldsymbol{\#pumps}*(read - now') - steam + \delta_{LOW}(v, v', \Delta t)$ *or stop* = ***true***. Based on the steam rate condition and Lemma 1.2:

$$\delta_{LOW}(a, b, u) \geq \begin{cases} a^2/(2*\boldsymbol{U_2}) & \text{if } a < \boldsymbol{U_2} * u \\ a * u - \boldsymbol{U_2}*u^2/2 & \text{otherwise} \end{cases}$$

   we distinguish in following cases:

1. Sub-case $v < \boldsymbol{U_2}(read\text{-}now)$ and $v < \boldsymbol{U_2} * \Delta t$:

   Since $\delta_{LOW}(v, v', \Delta t) \geq v^2/(2*\boldsymbol{U_2})$ and $v'^2/(2*\boldsymbol{U_2}) > 0$, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*\boldsymbol{\#pumps}*(read - now') - v'^2/(2*\boldsymbol{U_2})$ and this case true.

2. Sub-case $v < \boldsymbol{U_2}(read\text{-}now)$ and $v \geq \boldsymbol{U_2} * \Delta t$:

   Here, we know $\boldsymbol{M_2} > q' + \boldsymbol{P}*\boldsymbol{\#pumps}*(read - now') - v^2 /2*\boldsymbol{U_2} + (v *\Delta t - \boldsymbol{U_2}*\Delta t^2/2)$ and since $v^2/(2*\boldsymbol{U_2}) - (v *\Delta t - \boldsymbol{U_2}*\Delta t^2/2) = (v - \boldsymbol{U_2}*\Delta t)^2/(2*\boldsymbol{U_2})$ and $v - \boldsymbol{U_2}*\Delta t \leq v'$, we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*\boldsymbol{\#pumps}*(read - now') - v'^2 /2*\boldsymbol{U_2}$ and this case true.

3. Sub-case $v \geq \boldsymbol{U_2}(read\text{-}now)$:

   Since $now + \Delta t \leq read$ from the precondition, we know $v \geq \boldsymbol{U_2}*\Delta t$ and we get $\boldsymbol{M_2} > q' + \boldsymbol{P}*\boldsymbol{\#pumps}*(read - now') - v*\Delta t - (v*(read\text{-}now') - \boldsymbol{U_2}*\Delta t*(read\text{-}now') - \boldsymbol{U_2}*(read\text{-}now')^2/2) + \boldsymbol{U_2}*\Delta t^2/2 + (v * \Delta t - \boldsymbol{U_2}*\Delta t^2/2)$ *or stop* = ***true***. Since $v*(read\text{-}now') - \boldsymbol{U_2}*\Delta t*(read\text{-}now') = (v -$

U_2*Δt)*(read-now') - U_2*Δt*(read-now') and $v - U_2*\Delta t \leq v'$ from the effect, we get $M_2 > q' + P*\#pumps*(read - now') - (v'*(read-now') - U_2*(read-now')^2/2)$ *or stop* = ***true***.

This case is obviously true.

C) *a* = activate (all but *set* are unchanged):

Since *set = now* from the precondition, *now ≤ read* (Lemma 2) and *set = read +* ***S*** *or set = read -* ***I*** *+* ***S*** (Lemma 5), we know *set = read -* ***I*** *+* ***S*** and from the assumption $M_2 > q + P * \#pumps *(I\text{-}S) - steam$ *or stop* = ***true***. Since ***I*** - ***S*** = *read - now* and the effect sets *set' = read +* ***S*** this lemma is true for the activate action.

■

Lemma 14: *d(u)* is convex:

$d(u) \geq min(0, d(S))$ for $S \geq u \geq 0$, $d(u) = A*u - B*u^2$ with *A* real and *B* positive real

1. Case $u \leq A/(2*B)$:

   Proof (indirect): Suppose $d(u) < 0$. From $A*u\text{-}B*u^2 < 0$, we get $u > A/B$. Since $u \geq 0$ and $A/B > A/(2*B)$, we have a contradiction to the case assumption. We know $d(u) \geq 0 \geq min(0, d(S))$ and this case is true.

2. Case $u > A/(2*B)$:

   Proof (indirect): Suppose $d(u) < d(S)$. Define $S = u + \varepsilon$ with $\varepsilon > 0$. From $A*u\text{-}B*u^2 < A(u + \varepsilon) - B(u + \varepsilon)^2$ follows $u < A/(2*B) - \varepsilon/2$. Since $u \geq 0$ and $\varepsilon \geq 0$ we have a contradiction to the case assumption. We know $d(u) \geq d(S) \geq min(0, d(S))$ and this case is true.

■

6.3 Summarizing Theorems

The following theorems summarize the previous lemmas and translate them into the form in which the required properties were expressed.

Theorem 1: In all reachable states of boiler system,

v < ***W*** *or stop =* ***true***

Proof. Since we know $v + U_1*(read - now) < W$ *or stop* = ***true*** (Lemma 11), $U_1 > 0$ per definition and *read ≥ now* (Lemma 2) this theorem is true.

■

Theorem 2: In all reachable states of boiler system,

$M_1 < q < M_2$ *or stop* = ***true***

Proof. First, we show $M_1 < q$ *or stop* = ***true*** by induction on the steps of the automaton. It is true in the initial state and trivial for the actuator action. The only remaining action is *a* = time passage (*stop* is unchanged):

We know *do_output* = ***false*** from (Lemma 7 *if now < read then do_output* = ***false***, from the precondition (*now + Δt ≤ read*) and $\Delta t > 0$. Since we know *set = read +* ***S*** *or set = read -* ***I*** *+* ***S*** (Lemma 5), we can distinguish two cases:

A) Case $set = read - \boldsymbol{I} + \boldsymbol{S}$:

From Lemma 12, we get $\boldsymbol{M_1} < q + \boldsymbol{P}*pumps*(set\text{-}now) - (v * (read\text{-}now) + \boldsymbol{U_1}*(read\text{-}now)^2/2)$ *or stop* = ***true***. Using $(v * (read\text{-}now) + \boldsymbol{U_1}*(read\text{-}now)^2/2) > (v * (set\text{-}now) + \boldsymbol{U_1}*(set\text{-}now)^2/2)$ (since $set < read$), $pumps = pr$ from Lemma 10: *if set = read +* ***S*** *and do_output =* ***false*** *then pr = pr_new - error else pr = pumps* and $d(u) = A*u - B*u^2$ as defined in Lemma 14 with $A = \boldsymbol{P}*pr - v$ and $B = \boldsymbol{U_1}/2$, we get: $\boldsymbol{M_1} < q + d(set\text{-}now)$ *or stop* = ***true***.

From Lemma 14 follows that $d(\Delta t) \geq min(0, d(set\text{-}now))$ for $\Delta t \leq set\text{-}now$.

a. Sub-case $d(\Delta t) \geq d(set\text{-}now)$:

Here, we know $\boldsymbol{M_1} < q + d(\Delta t)$ *or stop* = ***true***. Since $q + pr * \boldsymbol{P} * \Delta t - \delta_{HIGH}(v, v', \Delta t) \leq q'$ from the effect which is equivalent to $q + d(\Delta t) \leq q'$ because $\delta_{HIGH}(a, b, u) \leq (a*u + \boldsymbol{U_1}*u^2/2)$ from Lemma 1.9, we know $\boldsymbol{M_1} < q'$ *or stop* = ***true*** and this sub-case true.

b. Sub-case $d(\Delta t) \geq 0$:

We assume $\boldsymbol{M_1} < q$ *or stop* = ***true***. Since $d(\Delta t) \geq 0$ and $q + pr * \boldsymbol{P} * \Delta t - \delta_{HIGH}(v, v', \Delta t) \leq q'$ from the effect which is equivalent to $q + d(\Delta t) \leq q'$ because $\delta_{HIGH}(a, b, u) \leq (a*u + \boldsymbol{U_1}*u^2/2)$ from Lemma 1.9, we know $\boldsymbol{M_1} < q'$ *or stop* = ***true*** and this sub-case true.

B) Case $set = read + \boldsymbol{S}$:

From Lemma 12, we get $\boldsymbol{M_1} < q' - (v' * (read\text{-}now') + \boldsymbol{U_1}*(read\text{-}now')^2/2)$ *or stop* = ***true***. Since $v' * (read\text{-}now') + \boldsymbol{U_1}*(read\text{-}now')^2/2 \geq 0$ this lemma is true.

Second, we show $\boldsymbol{M_2} > q$ *or stop* = ***true*** trough induction on the steps of the automaton. It is true in the initial state and trivial for the actuator action. The only remaining action is a = time passage (*stop* is unchanged):
We know *output* = ***false*** from (Lemma 7) *if now < read then do_output =* ***false***, from the precondition ($now + \Delta t \leq read$) and $\Delta t > 0$. Since we know $set = read + \boldsymbol{S}$ *or* $set = read - \boldsymbol{I} + \boldsymbol{S}$ (Lemma 5), we can distinguish following cases:

A) Case $set = read - \boldsymbol{I} + \boldsymbol{S}$:

From Lemma 13, we get $\boldsymbol{M_2} > q + \boldsymbol{P}*(pumps*(read - \boldsymbol{I} + \boldsymbol{S} - now) + \boldsymbol{\#pumps}*(\boldsymbol{I}\text{-}\boldsymbol{S})) - steam$ *or stop* = ***true***.

Using $\boldsymbol{\#pumps} \geq pumps$ per definition, $pumps = pr$ from Lemma 10: *if set = read +* ***S*** *and do_output =* ***false*** *then pr = pr_new - error else pr = pumps*, we get $\boldsymbol{M_2} > q + \boldsymbol{P}*pr*(read - now) - (v*(read\text{-}now) - \boldsymbol{U_2}*(read\text{-}now)^2/2) + \boldsymbol{P}*(pumps*(\boldsymbol{S}\text{-}\boldsymbol{I}) + pumps*(\boldsymbol{I}\text{-}\boldsymbol{S}))$ *or stop* = ***true***. The rest of the proof for this case is analog to the case $set = read + \boldsymbol{S}$.

B) Case $set = read + \boldsymbol{S}$ and $v \geq \boldsymbol{U_2}(read\text{-}now)$:

From Lemma 13 and using $\boldsymbol{\#pumps} \geq pr$ per definition, we get $\boldsymbol{M_2} > q + \boldsymbol{P}*pr*(read - now) - (v*(read\text{-}now) - \boldsymbol{U_2}*(read\text{-}now)^2/2)$ *or stop* = ***true***.

Since $d(u) = A*u - B*u^2$ as defined in Lemma 14 with $A = v - \boldsymbol{P}*pr$ and $B = \boldsymbol{U_2}/2$, we get: $\boldsymbol{M_2} > q - d(read - now)$ *or stop* = ***true***.

From Lemma 14 follows that $d(\Delta t) \geq min(0, d(read\text{-}now))$ for $\Delta t \leq read\text{-}now$.

a. Sub-case $d(\Delta t) \geq d(read\text{-}now)$:

Here, we know $\boldsymbol{M_2} > q - d(\Delta t)$ *or stop* = ***true***.

Since $q + pr * \boldsymbol{P} * \Delta t - \delta_{LOW}(v, v', \Delta t) \geq q'$ from the effect which is equivalent to $q - d(\Delta t) \geq q'$ because $v \geq \boldsymbol{U_2}(read\text{-}now)$, $read\text{-}now \geq \Delta t$ from the precondition and Lemma 1.2:

$$\delta_{LOW}(a, b, u) \geq \begin{cases} a^2/(2*\boldsymbol{U_2}) & \text{if } a < \boldsymbol{U_2} * u \\ a * u - \boldsymbol{U_2}*u^2/2 & \text{otherwise} \end{cases}$$

we know $\boldsymbol{M_2} > q'$ *or stop* = ***true*** and this sub-case true.

b. Sub-case $d(\Delta t) \geq 0$:

Here, we assume $\boldsymbol{M_2} > q$ *or stop* = ***true***. Since $d(\Delta t) \geq 0$ and $q + pr * \boldsymbol{P} * \Delta t - \delta_{LOW}(v, v', \Delta t) \geq q'$ from the effect which is equivalent to $q - d(\Delta t) \geq q'$ because $v \geq \boldsymbol{U_2}(read\text{-}now)$, $read\text{-}now \geq \Delta t$ from the precondition and Lemma 1.2, we know $\boldsymbol{M_2} > q \geq q - d(\Delta t) \geq q'$ *or stop* = ***true*** and this sub-case true.

C) Case $set = read + \boldsymbol{S}$ and $v < \boldsymbol{U_2}(read\text{-}now)$:

From Lemma 13 and using ***#pumps*** $\geq pr$ per definition, we get $\boldsymbol{M_2} > q + \boldsymbol{P}*pr*(read - now) - v^2/2*\boldsymbol{U_2}$ *or stop* = ***true***. From Lemma 1.2, we get two sub-cases:

a. Sub-case $v < \boldsymbol{U_2} * \Delta t$:

We get $\boldsymbol{M_2} > q + \boldsymbol{P}*pr*(read - now) - \delta_{LOW}(v, v', \Delta t)$ *or stop* = ***true***. Since $read - now \geq \Delta t$ from the precondition, we know $\boldsymbol{M_2} > q + \boldsymbol{P} * pr*\Delta t - \delta_{LOW}(v, v', \Delta t)$ *or stop* = ***true***. Since $q + \boldsymbol{P} * pr*\Delta t - \delta_{LOW}(v, v', \Delta t) \geq q'$, this case is true.

b. Sub-case $v \geq \boldsymbol{U_2} * \Delta t$:

We get $\boldsymbol{M_2} > q + \boldsymbol{P}*pr*(read - now) - v^2/(2*\boldsymbol{U_2})$ *or stop* = ***true***. Since $v^2/(2*\boldsymbol{U_2}) = v*(v/\boldsymbol{U_2}) - \boldsymbol{U_2}*(v/\boldsymbol{U_2})^2/2$, we know $\boldsymbol{M_2} > q + \boldsymbol{P}*pr*(read - now) - (v*(v/\boldsymbol{U_2}) - \boldsymbol{U_2}*(v/\boldsymbol{U_2})^2/2)$ *or stop* = ***true***. Using $d(u) = A*u - B*u^2$ as defined in Lemma 14 with $A = v - \boldsymbol{P}*pr$ and $B = \boldsymbol{U_2}/2$, we get: $\boldsymbol{M_2} > q - d(v/\boldsymbol{U_2}) + \boldsymbol{P}*pr*(read - now - v/\boldsymbol{U_2})$ *or stop* = ***true***. Since $pr \geq 0$ per definition and from the case statement we know $v < \boldsymbol{U_2}(read\text{-}now)$, we get $\boldsymbol{M_2} > q - d(v/\boldsymbol{U_2})$ *or stop* = ***true***.

From Lemma 14 follows that $d(\Delta t) \geq min(0, d(v/\boldsymbol{U_2}))$ for $\Delta t \leq v/\boldsymbol{U_2}$.

1. Sub-sub-case $d(\Delta t) \geq d(v/\boldsymbol{U_2})$:

   Here, we know, using Lemma 14, $\boldsymbol{M_2} > q - d(\Delta t)$ *or stop* = ***true***.

   Since $q + pr * \boldsymbol{P} * \Delta t - \delta_{LOW}(v, v', \Delta t) \geq q'$ from the effect which is equivalent to $q - d(\Delta t) \geq q'$ because $v \geq \boldsymbol{U_2} * \Delta t$ and Lemma 1.2, we know $\boldsymbol{M_2} > q'$ *or stop* = ***true*** and this sub-case true.

2. Sub-sub-case $d(\Delta t) \geq 0$:

   Here, we assume $\boldsymbol{M_2} > q$ *or stop* = ***true***. Since $d(\Delta t) \geq 0$ and $q + pr * \boldsymbol{P} * \Delta t - \delta_{LOW}(v, v', \Delta t) \geq q'$ from the effect which is equivalent to $q - d(\Delta t) \geq q'$ because $v \geq \boldsymbol{U_2} * \Delta t$ and Lemma 1.2, we know $\boldsymbol{M_2} > q \geq q - d(\Delta t) \geq q'$ *or stop* = ***true*** and this sub-case true.

■

With above proofs, we have shown that the steam boiler model together with the controller model meets all the safety requirements. As a further step, we must modify the controller model to allow sensor faults. This is presented in the following section.

7 Sensor Fault-tolerant Controller

In this section, we extend the model of the controller to be tolerant to sensor faults. Rather than proving the safety properties all over again, we use a technique called *Simulation Mapping*. This technique is used to show consistency between abstraction levels. In particular, it provides a means to show that properties proved for an abstract model are preserved in a particular implementation. In this case, the previously described boiler system represents the specification and a new controller that tolerates sensor faults represents a possible implementation.

First, we need some additional information about the boiler system with the previous controller. This knowledge will help us prove the *Simulation Mapping*. Both lemmas relate the situation in the boiler with what the controller got in the last sensor reading. The proofs show that the distance between the actual value and its last representation in the controller is bounded.

The following lemma presents an upper and lower boundary on the difference between the steam rate representation in the controller and the real steam rate depending on the time since the last sensor reading.

Lemma 15: In all reachable states of the combined steam boiler system using the simple controller,

$-\mathbf{U_2}*(\mathbf{I} + now - read) \leq v - sr \leq \mathbf{U_1}*(\mathbf{I} + now - read)$

Proof. In the start state this Lemma is true. We distinguish on the cases for the action *a*: For a ∈ {sensor, activate} this lemma is trivially true.

A) *a* = actuator (*now*, *v* and *sr* unchanged):
We know *do_output* = ***true*** from the precondition and since *if do_output then now = read and sr = v* (Lemma 4) and *read' = now +* ***I*** from the effect, we know $\mathbf{I} + now - read' = 0$ and $v - sr = 0$. Thus, this lemma is fulfilled.

B) *a* = time-passage (*read* and *sr* are unchanged):
We know from the precondition that $\Delta t \leq read - now$. From the effect we get:
$v' \geq -\mathbf{U_2}*\Delta t + v$, $v' \leq \mathbf{U_1}*\Delta t + v$ and $now' = now + \Delta t$. The assumption is equivalent to
$-\mathbf{U_2}*(\mathbf{I} + now + \Delta t - \Delta t - read) \leq v - sr$ and $v - sr \leq \mathbf{U_1}*(\mathbf{I} + now + \Delta t - \Delta t - read)$. This implies
$\mathbf{U_2}*\Delta t - \mathbf{U_2}*(\mathbf{I} + now + \Delta t - read) \leq v - sr$ and $v - sr \leq \mathbf{U_1}*(\mathbf{I} + now + \Delta t - read) - \mathbf{U_1}*\Delta t$. This is equivalent to $-\mathbf{U_2}*(\mathbf{I} + now + \Delta t - read) \leq v - \mathbf{U_2}*\Delta t - sr$ and $v + \mathbf{U_1}*\Delta t - sr \leq \mathbf{U_1}*(\mathbf{I} + now + \Delta t - read)$ which leads to the desired result $-\mathbf{U_2}*(\mathbf{I} + now' - read) \leq v' - sr \leq \mathbf{U_1}*(\mathbf{I} + now' - read)$.

■

Lemma 16: In all reachable states *s* of the combined steam boiler system using the simple controller,

if do_output = ***false*** *then* $ps - \delta_{HIGH}(sr, v, t) \leq q - wl \leq ps - \delta_{LOW}(sr, v, t)$

With $t = (\mathbf{I} + now - read)$ and

$$ps = \begin{cases} \mathbf{P} * pumps * t & \text{if } set = read + \mathbf{S} - \mathbf{I} \\ \mathbf{P} * (pumps * \mathbf{S} + pr * (t - \mathbf{S})) & \text{otherwise} \end{cases}$$

Proof. In the start condition this Lemma is true since $\delta_{LOW}(sr, sr',\Delta t) \leq \delta_{HIGH}(sr, sr',\Delta t)$ (Lemma 1.1), $\delta_{LOW}(sr, sr',\Delta t) \geq 0$ (Lemma 1.1) and $ps \geq 0$ since $pumps \geq 0$ and $pr \geq 0$ per definition. We distinguish on the cases for the action *a*:

A) *a* = sensor (*pr, set, , q, v, t, read* and *now* are unchanged):
We know *do_output'* = ***true*** from the effect. Thus, this lemma is trivially true.

B) *a* = actuator (*set, q, wl, now, pumps* and *pr* are unchanged):
We know *do_output* = ***true*** from the precondition *if do_output then now = read and sr= v and wl = q* (Lemma 4). Furthermore, we know $now \leq read - \boldsymbol{I} + \boldsymbol{S}$ *or* $set = read + \boldsymbol{S}$ (Lemma 6). Since *read* = *now*, we get $set = read + \boldsymbol{S}$ and from the effect, we get *do_output'* = ***false*** and $read' = now + \boldsymbol{I}$. Since it follows that $t' = (\boldsymbol{I} + now - read') = 0$ and $set = read' - \boldsymbol{I} + \boldsymbol{S}$, we know $\boldsymbol{P} * pumps * t' - \delta_{HIGH}(sr, v, t') \leq q - wl \leq \boldsymbol{P} * pumps * t' - \delta_{LOW}(sr, v, t)$ and this lemma is fulfilled.

C) *a* = time-passage (*pr, set, pumps, pr, wl* and *read* are unchanged):
We know *do_output* = ***false*** from *if now < read then do_output* = ***false*** (Lemma 7) and the precondition $now + \Delta t \leq read$. Furthermore, we know $set = read + \boldsymbol{S}$ *or* $set = read - \boldsymbol{I} + \boldsymbol{S}$ (Lemma 5) and following we distinguish these two cases:

a. Case $set = read + \boldsymbol{S}$:
We know from the effect: 1) $q' \geq q + pr * \boldsymbol{P} * \Delta t - \delta_{HIGH}(v, v', \Delta t)$ and 2) $q' \leq q + pr * \boldsymbol{P} * \Delta t - \delta_{LOW}(v, v', \Delta t)$. Substituting *q* in the assumption we get:
1) $q' \geq wl + \boldsymbol{P} * (pumps * \boldsymbol{S} + pr * (t - \boldsymbol{S})) - \delta_{HIGH}(sr, v, t) + pr * \boldsymbol{P} * \Delta t - \delta_{HIGH}(v, v', \Delta t)$
2) $q' \leq wl + \boldsymbol{P} * (pumps * \boldsymbol{S} + pr * (t - \boldsymbol{S})) - \delta_{LOW}(sr, v, t) + pr * \boldsymbol{P} * \Delta t - \delta_{LOW}(v, v', \Delta t)$
Since $\delta_{HIGH}(a, b, u) + \delta_{HIGH}(b, c, t) \leq \delta_{HIGH}(a, c, u + t)$ (Lemma 1.7), $\delta_{LOW}(a, b, u) + \delta_{LOW}(b, c, t) \geq \delta_{LOW}(a, c, t + u)$ (Lemma 1.4) and for $t' = (\boldsymbol{I} + now' - read) = (\boldsymbol{I} + now - read) + \Delta t$ this can be rewritten as 1) $q' \geq wl + \boldsymbol{P} * (pumps * \boldsymbol{S} + pr * (t' - \boldsymbol{S})) - \delta_{HIGH}(sr, v', t')$ and
2) $q' \leq wl + \boldsymbol{P} * (pumps * \boldsymbol{S} + pr * (t' - \boldsymbol{S})) - \delta_{LOW}(sr, v', t')$ and this case is true.

b. Case $set = read + \boldsymbol{S} - \boldsymbol{I}$:
In the same way as above, we get 1) $q' \geq wl + \boldsymbol{P} * pumps * t - \delta_{HIGH}(sr, v, t) + pr * \boldsymbol{P} * \Delta t - \delta_{HIGH}(v, v', \Delta t)$ and 2) $q' \leq wl + \boldsymbol{P} * pumps * t - \delta_{LOW}(sr, v, t) + pr * \boldsymbol{P} * \Delta t - \delta_{LOW}(v, v', \Delta t)$.
Since we know *if set = read +* ***S*** *and do_output =* ***false*** *then pr = pr_new - error else pr = pumps* (Lemma 9) and $t' = (\boldsymbol{I} + now' - read) = (\boldsymbol{I} + now - read) + \Delta t$, we get 1) $q' \geq wl + \boldsymbol{P} * pr * t' - \delta_{HIGH}(sr, v', t')$ and 2) $q' \leq wl + \boldsymbol{P} * pr * t' - \delta_{LOW}(sr, v', t')$ and this case is true, too.

D) *a* = activate (*do_output, sr, v, read, now* and *q* are unchanged):
We know from the precondition that *now* = *set*. Since we know $set = read + \boldsymbol{S}$ *or* $set = read - \boldsymbol{I} + \boldsymbol{S}$ (Lemma 5), $now \leq read$ (Lemma 2), we know $now = set = read - \boldsymbol{I} + \boldsymbol{S}$. Moreover, we know *if now < read then do_output* = ***false*** (Lemma 7) and since $\boldsymbol{S} < \boldsymbol{I}$, *do_output* = ***false***. Therefore, we know in the pre-state 1) $q \geq wl + \boldsymbol{P} * pumps * (\boldsymbol{I} + (read - \boldsymbol{I} + \boldsymbol{S}) - read) - \delta_{HIGH}(sr, v, t)$ and
2) $q \leq wl + \boldsymbol{P} * pumps * (\boldsymbol{I} + (read - \boldsymbol{I} + \boldsymbol{S}) - read) - \delta_{LOW}(sr, v, t)$.
Obviously, the following are also true since $t = \boldsymbol{I} + now - read = \boldsymbol{S}$ and $t - \boldsymbol{S} = 0$:
1) $q \geq wl + \boldsymbol{P} * (pumps * \boldsymbol{S} + pr' * (t - \boldsymbol{S})) - \delta_{HIGH}(sr, v, t)$
2) $q \leq wl + \boldsymbol{P} * (pumps * \boldsymbol{S} + pr' * (t - \boldsymbol{S})) - \delta_{LOW}(sr, v, t)$
Since the effect sets $set' = read + \boldsymbol{S}$ this lemma is fulfilled.

■

Following, we will present the Timed Automaton model of the sensor fault-tolerant controller.

7.1 The Controller Model Allowing Sensor Faults

Variables

Name	Initial Value	Type	Value Range	Unit	Description
do_output	***false***	boolean	{***true, false***}		flag that activates the output; This parameter represents a kind of program counter.
stopmode	***true***	boolean	{***true, false***}		flag to activate the emergency stop, initially true, since condition is not checked yet.
wll	*q*	real	[0 ... **C**]	l	lower bound of the estimation of the current water level
srl	0	real	[0 ... **W**]	l/s	lower bound of the estimation of the current steam rate
wlh	*q*	real	[0 ... **C**]	l	upper bound of the estimation of the current water level
srh	0	real	[0 ... **W**]	l/s	upper bound of the estimation of the current steam rate
sr_ok	***true***	boolean	{***true, false***}		flag that tells whether the steam rate sensor has failed
wl_ok	***true***	boolean	{***true, false***}		flag that tells whether the water level sensor has failed
now	0	real	[0 ... ∞)	s	current time
pumps	0	integer	{0 ... ***#pumps***}		number of currently active pumps supplying water to the boiler
px	0	integer	{0 ... ***#pumps***}		number of pumps that shall supply water next

Table 5: The initial state of the fault-tolerant controller including all variable declarations

7.2 The Fault-tolerant Controller Automaton

Input Actions

sensor (s, w, p)

Effect:

pumps' = p
do_output' = ***true***

\# estimate steam rate
if sr_ok then srh' = srl' = s
else srh' = srh + $\boldsymbol{U_1}$ *** ***I***
srl' = srl - $\boldsymbol{U_2}$ *** ***I***

\# estimate water level
if wl_ok then wlh' = wll' = w
else wlh' = wlh + ***P*** ** pumps ** ***S*** *+* ***P*** ** pumps' * (****I*** *-* ***S****)*
- min_steam_water_est(srl')
wll' = wll + ***P*** ** pumps ** ***S*** *+* ***P*** ** pumps' * (****I*** *-* ***S****)*
- (srh' + $\boldsymbol{U_2}$*****I****/2)****I***

\# safety checks
if srh' ≥ ***W*** *-* $\boldsymbol{U_1}$ *** ***I*** *or*
wlh' ≥ $\boldsymbol{M_2}$ *-* ***P*** **(pumps' ** ***S*** *+ (max_pumps_after_set)*
** (****I*** *-* ***S****)) + min_steam_water(srl) or*
wll' ≤ $\boldsymbol{M_1}$ *+* ***P*** **(pumps' ** ***S*** *+ (min_pumps_after_set)*
** (****I*** *-* ***S****)) - max_steam_water(srh)*
then stopmode' = ***true***
*else stopmode' = {****true, false****} arbitrary*

Internal Actions

bad

Precondition:
true
Effect:
*sr_ok' = {****true****,* ***false****} arbitrary*
*wl_ok' = {****true****,* ***false****} arbitrary*

controller

Precondition:
true
Effect:
0 ≤ px' ≤ ***#pumps***

ν(Δt)

Precondition:
true
Effect:
now' = now + Δt

Output Actions

actuator (e_stop, pset)

Precondition:
do_output = ***true***
pset = px
e_stop = stopmode
Effect:
do_output' = ***false***

The controller model that allows sensor faults has the same structure as the simple controller. An additional action ***bad*** tell the controller whether a sensor has failed. The fault model allows arbitrary combinations of sensor break downs and fast or slow repairs. The ***sensor*** action expresses the strategy of the controller to cope with sensor faults. Basically, the strategy is to calculate an upper and lower limit for the missing value of the failed sensor, using its last recent value and the remaining sensor values. Even in the case that both sensors break, the controller still may allow the operation of the boiler and guarantee safety. In this respect, our controller definition is better than the one suggested in [AS96], since he suggests to shut down the boiler system whenever both steam rate and water level sensors fail.

The various operational modes (normal, degraded and rescue) as specified in [AS96] can be inferred from the variables *sr_ok*, *wl_ok* and the difference between *pumps* and *px*. In our model, these modes are not relevant to the safety of the boiler system and have therefore been ignored.

7.3 Proving the Safety Properties by Simulation Mapping

After composing the steam boiler automaton with the new fault-tolerant controller, we have to prove that the safety properties are satisfied in the new model.

We use a **Simulation Mapping** for proving that one Timed Automaton "implements" another. This technique shows that all possible traces[‡] of the new automata are included in the traces of the already proven model. Therefore, all safety properties involving the states of the steam boiler with the simple controller are valid for the system with the fault-tolerant controller, too. A Simulation Mapping is most useful to show that an implementation actually preserves properties of the specification. This method can be applied repeatedly to get from a very abstract model, which is proven to fulfill the required properties, to a detailed implementation (maybe even the final implementation). Like invariants, the Simulation Mappings involve time deadline information, in particular, they include inequalities between time deadlines. Therefore, they are suitable for showing timing properties, too.

We apply a Simulation Mapping from states of the steam boiler system with the fault-tolerant controller (in short "fault-tolerant controller system") to the system with the simple controller ("simple controller system"). Appendix B contains a formal definitions of the Simulation Mapping technique and the correctness properties it guarantees.

7.3.1 Simulation Relation

Theorem 3: The relation *f* as defined below is a Simulation Mapping from the states of the fault-tolerant controller system to the states of the simple controller system.

Let *s* denote a state of the simple controller system and *i* denote a state of the fault-tolerant controller system. We define *s* and *i* to be related by the relation f provided that:

1) *i.Boiler = s.Boiler*[§]
2) *i.do_output = s.do_output, s.px = i.px, s.pumps = i.pumps, s.now = i.now*
3) $i.srl \le s.sr \le i.srh$
4) $i.wll \le s.wl \le i.wlh$
5) *s.stopmode = i.stopmode*

‡ The exact meaning of "traces" is defined in Appendix A in the full version.
§ This relation expresses that the entire boiler state is preserved.

Proof. Let *i* lead to *i'* via action *a* in the fault-tolerant controller. We must find an *s'* such that *s' f i'* and there exists an execution fragment from *s* to *s'* with the same trace as *a*. Usually, we break by cases on the type of *a*. In the initial state *f* is fulfilled. For this proof it remains to show the case for the sensor action because all other actions are identical in the specification and implementation. It remains to show that there is an equivalent sensor step enabled in *s*, and *s'* relates to *i'* following the definition of *f*. In particular, we must show the three conditions in the definition of a Simulation Mapping in Appendix C. The first condition, preservation of the *now* value, is immediate from the definition of *f*. The second condition is also immediate, because *f* is fulfilled between the start states. The interesting condition is the step condition. For *a* = sensor action we get:

The simulation relation is satisfied for the initial states. The precondition is the same, thus the sensor action is enabled for both systems.

A) Statements *1)* and *2)* of the relation are trivially true for all actions but the sensor action since clearly, *i.pumps' = s.pumps' = p and i.do_output' = s.do_output'* = ***true*** and for any choice of *i.px* we can get the same value for *s.px* from the controller action.

B) Statement *3)*:
We analyze this statement based on the fault situation for the steam rate sensor:
In case *i.sr_ok* = ***true***, we get from the implementation *if i.sr_ok then i.srh' = i.srl' = s*. Clearly, this case is true. Otherwise, we know $-\mathbf{U_2}*(\mathbf{I} + s.now - s.read) \le s.v - s.sr \le \mathbf{U_1}*(\mathbf{I} + s.now - s.read)$ (Lemma 15) and since *s.now = s.read = i.now = i.read* and *s.sr' = s.v = i.v* from the preconditions, we get $s.sr \le s.sr' + \mathbf{U_2}*\mathbf{I}$ and $s.sr' - \mathbf{U_1}*\mathbf{I} \le s.sr$. We know from the assumption $i.srl \le s.sr \le i.srh$ and this is equivalent to $i.srl \le s.sr' + \mathbf{U_2}*\mathbf{I}$ and $s.sr' - \mathbf{U_1}*\mathbf{I} \le i.srh$ and further equivalent to $i.srl - \mathbf{U_2} * \mathbf{I} \le s.sr' \le i.srh + \mathbf{U_1} * \mathbf{I}$. Since we assume here that the steam sensor failed, we know from the effect $i.srh' = i.srh + \mathbf{U_1} * \mathbf{I}$ and $i.srl' = i.srl - \mathbf{U_2} * \mathbf{I}$. Thus, we get $i.srl' \le s.sr' \le i.srh'$ and this statement is true.

C) Statement *4)*:
We analyze this statement based on the fault situation for the water level sensor:
In case *i.wl_ok* = ***true***, we get from the implementation *if i.wl_ok then i.wlh' = i.wll' = w*. Clearly, this case is true. Otherwise, we know from Lemma 16 *if s.do_output* = ***false*** *then* $ps - \delta_{HIGH}(s.sr, s.v, t) \le s.q - s.wl \le ps - \delta_{LOW}(s.sr, s.v, t)$. With $ps = $ *if* $s.set = s.read + \mathbf{S} - \mathbf{I}$ *then* $\mathbf{P} * s.pumps * t$ *else* $\mathbf{P} * (s.pumps * \mathbf{S} + s.pr * (t - \mathbf{S}))$ and $t = (\mathbf{I} + s.now - s.read)$.
We know $s.now \le s.read - \mathbf{I} + \mathbf{S}$ *or* $s.set = s.read + \mathbf{S}$ (Lemma 6), *s.do_output* = ***false*** and *s.now = s.read* from the precondition. Thus, we know $s.set = s.read + \mathbf{S}$ and since *s.now = s.read = i.now = i.read*, *s.v = i.v* and *s.wl' = s.q = i.q* from the preconditions, we get
1. $\mathbf{P} * (i.pumps * \mathbf{S} + i.pr * (\mathbf{I} - \mathbf{S})) - \delta_{HIGH}(s.sr, i.v, \mathbf{I}) \le s.wl' - s.wl$ and
2. $s.wl' - s.wl \le \mathbf{P} * (i.pumps * \mathbf{S} + i.pr * (\mathbf{I} - \mathbf{S})) - \delta_{LOW}(s.sr, i.v, \mathbf{I})$.

We know

$$\delta_{LOW}(a, b, u) \ge \begin{cases} b^2/(2*\mathbf{U_1}) & \text{if } b < \mathbf{U_2} * u \\ b * u - \mathbf{U_1}*u^2/2 & \text{otherwise} \end{cases}$$

and $\delta_{HIGH}(a, b, u) \le (b + \mathbf{U_2}*u/2)*u$ (Lemma 1.3&6) and from this we get
1. $\mathbf{P} * (i.pumps * \mathbf{S} + i.pr * (\mathbf{I} - \mathbf{S})) - (i.v + \mathbf{U_2}*\mathbf{I}/2)*\mathbf{I} \le s.wl' - s.wl$ and
2. $s.wl' - s.wl \le \mathbf{P}*(i.pumps * \mathbf{S} + i.pr * (\mathbf{I} - \mathbf{S})) - steam$ with

$$steam = \begin{cases} i.v^2/(2*\mathbf{U_1}) & \text{if } i.v < \mathbf{U_2} * \mathbf{I} \\ (i.v * \mathbf{I} - \mathbf{U_1}*\mathbf{I}^2/2) & \text{otherwise} \end{cases}$$

Since *i.pumps' = s.pumps' = p = i.pr* from the effect and precondition, we get

1. $\boldsymbol{P} * (i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - (i.v + \boldsymbol{U_2}*\boldsymbol{I}/2)*\boldsymbol{I} \leq s.wl' - s.wl$ and
2. $s.wl' - s.wl \leq \boldsymbol{P}*(i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - steam$

Since we know from the assumption $i.wll \leq s.wl \leq i.wlh$

1. $\boldsymbol{P} * (i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - (i.v + \boldsymbol{U_2}*\boldsymbol{I}/2)*\boldsymbol{I} \leq s.wl' - i.wll$
2. $s.wl' - i.wlh \leq \boldsymbol{P}*(i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - steam$

We already know $i.srl \leq s.sr \leq i.srh$. Thus, it must also be $i.srl' \leq s.sr' \leq i.srh'$. Furthermore, we know $i.v = s.v = s.sr'$ from the 1. statement and the effect. From this, we get

1. $\boldsymbol{P} * (i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - (i.srh' + \boldsymbol{U_2}*\boldsymbol{I}/2)*\boldsymbol{I} \leq s.wl' - i.wll$
2. $s.wl' - i.wlh \leq \boldsymbol{P}*(i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - steam'$

with $steam' = \begin{cases} i.srl'^2/(2*\boldsymbol{U_1}) & \text{if } i.srl' < \boldsymbol{U_2} * \boldsymbol{I} \\ (i.srl' * \boldsymbol{I} - \boldsymbol{U_1}*\boldsymbol{I}^2/2) & \text{otherwise} \end{cases}$

This is equivalent to $i.wll + \boldsymbol{P} * (i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - (i.srh' + \boldsymbol{U_2}*\boldsymbol{I}/2)*\boldsymbol{I} \leq s.wl' \leq i.wlh + \boldsymbol{P}*(i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - steam'$.

Since we assume for this case that the water level sensor failed, we know

1. $i.wlh' = i.wlh + \boldsymbol{P} * (i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - min\_steam\_water\_est(i.srl')$
2. $i.wll' = i.wll + \boldsymbol{P} * (i.pumps * \boldsymbol{S} + i.pumps' * (\boldsymbol{I} - \boldsymbol{S})) - (i.srh' + \boldsymbol{U_2}*\boldsymbol{I}/2)*\boldsymbol{I}$

Thus, we get $i.wll' \leq s.wl' \leq i.wlh'$ and this statement is true.

D) Statement *5)*:

We distinguish two cases:

1. **Case** $i.srh' \geq \boldsymbol{W} - \boldsymbol{U_1} * \boldsymbol{I}$ *or*
   $i.wlh' \geq \boldsymbol{M_2} - \boldsymbol{P} *(i.pumps' * \boldsymbol{S} + \boldsymbol{\#pumps} * (\boldsymbol{I} - \boldsymbol{S})) + min\_steam\_water\_est(i.srl')$ *or*
   $i.wll' \leq \boldsymbol{M_1} + \boldsymbol{P} * i.pumps' * \boldsymbol{S} - (i.srh' * \boldsymbol{I} + \boldsymbol{U_1} * \boldsymbol{I}^2/2)$:
   In this case, we know from the effect: *if* $i.srh' \geq \boldsymbol{W} - \boldsymbol{U_1} * \boldsymbol{I}$ *or*
   $wlh' \geq \boldsymbol{M_2} - \boldsymbol{P} *(i.pumps' * \boldsymbol{S} + \boldsymbol{\#pumps} * (\boldsymbol{I} - \boldsymbol{S})) + min\_steam\_water\_est(i.srl')$ *or*
   $wll' \leq \boldsymbol{M_1} + \boldsymbol{P} * i.pumps' * \boldsymbol{S} - (i.srh' * \boldsymbol{I} + \boldsymbol{U_1} * \boldsymbol{I}^2/2)$ *then* $i.stopmode' = \boldsymbol{true}$
   Let us define $\boldsymbol{A_I}$ to be $\boldsymbol{M_2} - \boldsymbol{P} * (i.pumps' * \boldsymbol{S} - \boldsymbol{\#pumps} * (\boldsymbol{I} - \boldsymbol{S})) + min\_steam\_water\_est(i.srl')$ and $\boldsymbol{B_I}$ to be $\boldsymbol{M_1} - \boldsymbol{P} * i.pumps' * \boldsymbol{S} + i.srh' * \boldsymbol{I} + \boldsymbol{U_1} * \boldsymbol{I}^2/2$.
   Symmetrically, we know for specification *if* $s.sr' \geq \boldsymbol{W} - \boldsymbol{U_1} * \boldsymbol{I}$ *or*
   $s.wl' \geq \boldsymbol{M_2} - \boldsymbol{P} *(s.pumps' * \boldsymbol{S} + \boldsymbol{\#pumps} * (\boldsymbol{I} - \boldsymbol{S})) + min\_steam\_water\_est(s.sr')$ *or*
   $s.wl' \leq \boldsymbol{M_1} + \boldsymbol{P} * s.pumps' * \boldsymbol{S} - (s.sr' * \boldsymbol{I} + \boldsymbol{U_1} * \boldsymbol{I}^2/2)$ *then* $s.stopmode' = \boldsymbol{true}$
   In the same way as before, we define $\boldsymbol{A_S}$ to be $\boldsymbol{M_2} - \boldsymbol{P} * (s.pumps' * \boldsymbol{S} - \boldsymbol{\#pumps} * (\boldsymbol{I} - \boldsymbol{S})) + min\_steam\_water\_est(i.sr')$ and $\boldsymbol{B_S}$ to be $\boldsymbol{M_1} - \boldsymbol{P} * s.pumps' * \boldsymbol{S} + s.sr' * \boldsymbol{I} + \boldsymbol{U_1} * \boldsymbol{I}^2/2$.
   Since we know statements 2, 3 and 4 are also valid for the post-state, we get $i.srh' \leq s.sr' \leq i.srh'$, $i.wll' \leq s.wl' \leq i.wlh'$ and $s.pumps' = i.pumps'$. Therefore, $\boldsymbol{A_I} \leq \boldsymbol{A_S}$ and $\boldsymbol{B_I} \geq \boldsymbol{B_S}$ and from the effect $i.stopmode' = \boldsymbol{true}$. From this we get following cases:
   a) Case $s.sr' \geq \boldsymbol{W} - \boldsymbol{U_1} * \boldsymbol{I}$ *or* $s.wl' \geq \boldsymbol{A_S}$ *or* $s.wl' \leq \boldsymbol{B_S}$:
      Clearly, in this case $i.stopmode' = s.stopmode' = \boldsymbol{true}$ from the effect.
   b) Otherwise we can get $i.stopmode' = s.stopmode' = \boldsymbol{true}$ from the non-deterministic choice in the specification.
2. **Otherwise**: We know $i.srh' < \boldsymbol{W} - \boldsymbol{U_1} * \boldsymbol{I}$ and $i.wlh' < \boldsymbol{A_I}$ and $i.wll' > \boldsymbol{B_I}$ (using the same definitions as in the other case) and since $s.sr' \leq i.srh'$ and $i.wll' \leq s.wl' \leq i.wlh'$, we know $s.sr' < \boldsymbol{W} - \boldsymbol{U_1} * \boldsymbol{I}$, $s.wl' < \boldsymbol{A_I}$ and $s.wl' > \boldsymbol{B_I}$. Since $\boldsymbol{A_I} \leq \boldsymbol{A_S}$, $\boldsymbol{B_I} \geq \boldsymbol{B_S}$, we know from the effect that *i.stopmode'* and *s.stopmode'* can be true or false arbitrarily. Thus, this lemma is true.

■

This Simulation Mapping maps every reachable state of the boiler system with the fault-tolerant controller to a corresponding reachable state in the system with the simple controller by the relation f. Therefore, the safety properties involving the states of the specification (simple controller) are valid for the implementation (fault-tolerant controller), too. Thus, we have shown that the steam boiler system with the fault-tolerant controller satisfies the required safety properties.

8 Conclusion

We have applied a formal method based on Timed Automata, invariant assertions and Simulation Mappings to the steam boiler model and verified that our controller fulfills the required safety properties. In doing so we have made it possible to compare our techniques to other approaches.

Summarizing, the *Timed Automata*, *composition* and *Simulation Mapping* techniques present an excellent combination for system analysis. The main advantage of Timed Automata is their flexibility in modeling a hybrid system. Timed Automata allow us to combine a continuous environment that is fairly unpredictable over time with a discrete control system such as a computer. The composition and Simulation Mapping techniques supplement this specification tool for formal verification, for more flexibility in how to search for a solution and for the reuse of already gained knowledge. The composition technique lets you combine different automata and scale incrementally solutions from smaller problems to more complex ones. The Simulation Mapping technique provides a consistent transition between different abstraction layers.

This method seems to scale better than other formal verification techniques because of the possibility of applying this method to different abstraction layers, and applying various decomposition techniques [Wei96]. A Simulation Mapping can be used to prove that two abstraction layers preserve certain properties. Decomposition techniques provide modular and incremental verification. For instance, suppose that you have proved that a certain implementation of a shared register provides mutual exclusion. The automaton model together with already proved properties may then be composed into a bigger application without having to prove the mutual exclusion property again.

Constructing the proofs, though not difficult, requires significant work. The hardest parts were getting the details of the models right and finding the right invariants. Unfortunately, this seems to be an art rather than an automatic procedure. Nevertheless, our experience in this paper and others (e.g., [Hei94]) shows that this art is easily learnable even for application engineers. The techniques are very systematic and understandable. The description allows for much flexibility and is very powerful in describing the possible progression of a system.

The actual proofs of the invariants were tedious but routine work. Much work can be avoided by proving the required properties on a general model and using *Simulation Mappings* for more specialized models. Moreover, the characteristics of these techniques make them amenable for mechanical generation and verification of proofs. Related to this, we are currently considering the use of automatic provers such as Larch [Soe93] or PVS [Sha93] with the described techniques.

The only major disadvantage we encountered while working with Timed Automata and the Simulation Mapping technique is that we could not gain any information or any measurement towards the optimality of parameters of a solution. Although our controllers preserve provable safety, there are obviously better implementations. For example, on a steam rate sensor failure, the steam rate estimation could take into account the amount of water which has evaporated since the last sensor reading.

Moreover, we like to note that more of the reality could be modeled formally with a more relaxed pump failure model and diverse pump controller algorithms. The latter might lead to interesting performance comparisons and tighter parameters such as the distance between M_1 and M_2.

Future work includes applying this method to larger and more complex examples, and developing the appropriate computer assistance for carrying out and checking the proofs. On-going research in our group shows that the timed-automata method provides high potential for automating the generation of the proofs [Sha93], [Arc96].

Acknowledgments

We thank Anya Pogosyants and Roberto Segala for several useful comments as well as Angelika Leeb and Dave Evans for comments and proofreading.

References

[AS96] Abrial, J.-R.: A B-solution for the steam-boiler problem. Contains: Steam-boiler control specification problem for the meeting Methods for Semantics and Specification, Dagstuhl; See chapter AS in this LNCS volume.

[Arc96] Archer, M.; Heitmeyer, C.: Mechanical Verification of Timed Automata: A Case Study, To appear in the proceedings of RTAS ’96, 1996

[Cle93] Cleaveland, R.; Parrow, J.; Steffen, B.: The concurrency workbench: A semantics-based tool for verification of concurrent systems. ACM Trans. on Prog. Lang. and Sys., 15(1):36-72, Jan. 1993

[Hei93] Heitmeyer, C.; Jeffords, R.; Labaw, B.: A benchmark for comparing different approaches for specifying and verifying real-time systems. In Proc., 10th Intern Workshop on Real-Time Operating Systems and Software, May, 1993

[Hei94] Heitmeyer, C.; Lynch, N.: The Generalized Railroad Crossing: A Case Study in Formal Verification of Real-Time Systems. In Proceedings of the 15th IEEE Real-Time Systems Symposium, San Juan, Puerto Rico, IEEE Computer Society Press, pages 120 -131, December 1994

[Hoa93] Hoare, C.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs, NJ, 1985

[Jah86] Jahanian, F.; Mok, A.: Safety analysis of timing properties in real-time systems. IEEE Trans. Software Engineering, SE-12(9), Sep. 1986

[Lyn91] Lynch, N.; Vaandrager, F.: Forward and backward simulations for timing-based systems. In Proceedings for REX Workshop: Real-Time: Theory in Practice, vol. 600 of Lecture Notes in Computer Science, p. 397-446, Mook, Netherlands, Springer-Verlag, June 1991

[Lyn94] Lynch, N.: Simulation Techniques for Proving Properties of Real-time Systems, In REX Workshop ‘93, Lecture Notes in Computer Science, Mook, the Netherlands, Springer Verlag, 1994

[Soe93] Soegaard-Anderson, J.; Garland, S.; Guttag, J.; Lynch, N.; Pogosyants, A.: Computer-assisted simulation proofs, In Costas Courcoubetis, Computer-Aided Verification: 5th International

Conference, (CAV'93 Elounda, Greece, June/July 1993, Lecture Notes in Computer Science 697, p. 305-319, Springer-Verlag, 1993

[Seg94] Segala, R.; Lynch, N.: Probabilistic Simulations for Probabilistic Processes. In J. Parrow, Editor, Proceedings of CONCUR 94, Lecture Notes in Computer Science, volume 836, pages 481-496, Uppsala, Sweden, August 1994.

[Sha93] Shankar, N.: Verification of real-time systems using PVS. in Proc. Computer Aided Verification (CAV'93), pages 280-291. Springer-Verlag 1993

[Wei96] Weinberg, H.: Correctness of a Vehicle Control System: A Case Study, Master's Thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, 1996

APPENDIX - A: The Timed Automaton Model

This section contains the formal definitions for the Timed Automaton model, taken from [Lyn94].

Timed Automata

A *Timed Automaton A* consists of a set *states(A)* of states, a non-empty set *start(A)* $\subseteq$ *states(A)* of start states, a set *acts(A)* of actions, including a special *time-passage* action *ν*, a set *steps(A)* of steps (transitions), and a mapping now_A*: states* $\rightarrow \mathbb{R}_0$ ($\mathbb{R}_0$ denotes the nonnegative real numbers). Here, $now_A(s)$ represents the point in time of state *s*. The actions are partitioned into *external* and *internal* actions, where *ν* is considered external; the *visible* actions are the non-*ν* external actions; the visible actions are partitioned into *input* and *output* actions. The set *steps(A)* is a subset of *states(A)* $\times$ *acts(A)* $\times$ *states(A)*. We write $s \xrightarrow{\pi}_A s'$ as shorthand for $(s,\pi,s') \in$ steps(A). Usually, we write $s.now_A$ in place of $now_A(s)$.

A Timed Automaton must satisfy five axioms: [A1] If $s \in$ *start* then *s.now* = 0. [A2] If $s \xrightarrow{\pi}_A s'$ and $\pi \neq \nu$ then *s.now* = *s'.now*. [A3] If $s \xrightarrow{\nu}_A s'$ then *s.now* < *s'.now*. [A4] If $s \xrightarrow{\nu}_A s'$ and $s' \xrightarrow{\nu}_A s''$, then $s \xrightarrow{\nu}_A s''$. Axiom [A1] says that the current time is always 0 in a start state. Axiom [A2] says that non-time-passage steps do not change the time; that is, they occur “instantaneously”, at a single point in time. Axiom [A3] says that time-passage steps must cause the time to increase; this is a convenient technical restriction. Axiom [A4] (*transitivity of time-passage steps*) allows repeated time-passage steps to be combined into one step.

The statement of [A5] (*trajectory consistency*) requires a preliminary definition of a *trajectory*, which describes restrictions on the state changes that can occur during time-passage. Namely, if *I* is any interval of $\mathbb{R}_0$, then a *I-trajectory* is a function *w:I* $\rightarrow$ *states*, such that *w(t).now* = *t* for all $t \in I$, and $w(t_1) \xrightarrow{\nu}_A w(t_2)$ for all $t_1, t_2 \in I$ with $t_1 < t_2$. That is, *w* assigns, to each time *t* in interval *I*, a state having the given time *t* as its *now* component. This assignment is done in such a way that time-passage steps can span between any pair of states in the range of *w*. If *w* is an *I*-trajectory and *I* is left-closed, then define *w.ftime* = *min(I)* and *w.fstate* = *w(w.ftime)*, while if *I* is right-closed, then define *w.ltime* = *max(I)* and *w.lstate* = *w(w.ltime)*. If *I* is a closed interval, then an *I*-trajectory *w* is said to span from state *s* to state *s'* if *w.fstate* = *s* and *w.lstate* = *s'*. The final axiom is: [A5] If $s \xrightarrow{\nu}_A s'$ then there exists a trajectory that spans from *s* to *s'*. Axiom [A5] is a kind of converse to [A4]; it says that any time-passage step can be “filled in” with states for each intervening time, in a “consistent” way.

Timed Executions and Timed Traces

A timed *execution fragment* is a finite or infinite alternating sequence $\alpha = w_0 \pi_1 w_1 \pi_2 w_2 ....$, where:

1. Each w_j is a trajectory and each π_j is a non-time-passage action.
2. If α is a finite sequence, then it ends with a trajectory.
3. If w_j is not the last trajectory in α then its domain is a closed interval. If w_j is the last trajectory then its domain is left-closed (and either right-open or right-closed).

4. If w_j is not the last trajectory then $w_j.\,lstate \xrightarrow{\pi_{j+1}} w_{j+1}.\,fstate$.

The trajectories describe the changes of state during the time-passage steps. The last item says that the actions in α span between successive trajectories. A *timed execution* is a timed execution fragment for which the first state of the first trajectory, w_0 , is a start state. In this paper, we restrict attention to the *admissible* timed executions, i.e. those in which the *now* values occurring in the states approach ∞. We use the notation *atexecs(A)* for the set of admissible timed executions of Timed Automaton *A*. A state of a Timed Automaton is defined to be *reachable* if it is the final state of the final trajectory in some finite timed execution of the automaton.

In order to describe the problems to be solved by Timed Automata, we require a definition for their visible behavior. We use the notion of *timed traces* , where the *timed traces* of any timed execution is just the sequence of visible events that occur in the timed execution, paired with their times of occurrence. The *admissible timed traces* of the Timed Automaton are just the timed traces that arise from all the admissible timed executions. We use the notation *attraces(A)* for the admissible timed traces of Timed Automaton *A*. Often, we express requirements to be satisfied by a Timed Automaton *A* as the set of admissible timed traces of another Timed Automaton *B*. Then we say that *A implements B* if *attraces(A)* ⊆ *attraces(B)*. If α is any timed execution, we use the notation *ttrace(α)* to denote the timed trace of α.

We define a function *time* that maps any non-time-passage event in an execution to the real time at which it occurs. Namely, let π be any non-time-passage event. If π occurs in state *s*, then define *time(π) = s.now*.

Composition

We define a simple binary parallel composition operator for Timed Automata. Let *A* and *B* be Timed Automata satisfying the following *compatibility* conditions: *A* and *B* have no output actions in common, and no internal action of *A* is an action of *B*, and vice versa. Then the *composition* of *A* and *B*, written as $A \times B$, is the Timed Automaton defined as follows.

- $states(A \times B) = \{(s_A, s_B) \in states(A) \times states(B) : s_A.now_A = s_B.now_B\}$;
- $start(A \times B) = start(A) \times start(B)$;
- $acts(A \times B) = acts(A) \cup acts(B)$; an action is *external* in $A \times B$ exactly if it is external in either *A* or *B*, and likewise for *internal* actions; a visible action of $A \times B$ is an *output* in $A \times B$ exactly if it is an output in either *A* or *B*, and is an *input* otherwise;
- $(s_A , s_B) \xrightarrow{\pi}_{A\times B} (s'_A , s'_B)$ exactly if
  1. $s_A \xrightarrow{\pi}_A s'_A$ if $\pi \in acts(A)$, else $s_A = s'_A$, and
  2. $s_B \xrightarrow{\pi}_B s'_B$ if $\pi \in acts(B)$, else $s_A = s'_A$;
- $(s_A , s_B).now_{A\times B} = s_A.now_A$.

Then $A \times B$ is a Timed Automaton. If α is a timed execution of $A \times B$, we write $\alpha|A$ and $\alpha|B$ for the projection of α on *A* and *B*, respectively. For instance, $\alpha|A$ is defined by projecting all states in α on the state of *A*, removing actions that do not belong to *A*, and collapsing consecutive trajectories. We also use the projection notation for sequences of actions, writing, e.g., $\beta \lceil A$ for the subsequence of β consisting of actions of A.

Lemma A.1 (Substitutivity) Let A and B be Timed Automata with the same input and output actions and let C be a Timed Automaton compatible to both. If $attraces(A) \subseteq attraces(B)$ then $attraces(A \times C) \subseteq attraces(B \times C)$.

Lemma A.2 If $\alpha \in atexecs(A \times B)$ then $\alpha \lceil A \in atexecs(A)$ and $\alpha \lceil B \in atexecs(B)$.

Lemma A.3 Suppose that $\alpha_A \in atexecs(A)$ and $\alpha_B \in atexecs(B)$. Suppose β is a sequence of timed visible actions of $A \times B$ such that $\beta \lceil A = ttrace(\alpha_A)$ and $\beta \lceil B = ttrace(\alpha_B)$. Then there exists $\alpha \in atexecs(A \times B)$ such that $\alpha \lceil A = \alpha_A$ and $\alpha \lceil B = \alpha_B$.

Since the composition operation is associative, up to isomorphism, we may extend it to an arbitrary finite number of argument Timed Automata.

APPENDIX - B: Invariants and Simulation Mappings

We define an *invariant* of a Timed Automaton to be any property that is true of all reachable states.

The definition of a Simulation Mapping is paraphrased from [Lyn91, Lyn94]. We use the notation $f[s]$, where f is a binary relation, to denote $\{u : (s,u) \in f\}$. Suppose A and B are Timed Automata and I_A and I_B are invariants of A and B, respectively. Then a *Simulation Mapping* from A to B with respect to I_A and I_B is a relation f over $states(A)$ and $states(B)$ that satisfies:

1. If $u \in f[s]$ then $u.now = s.now$.
2. If $s \in start(A)$ then $f[s] \cap start(B) \neq \{\}$.
3. If $s \xrightarrow{\pi}_A s'$, $s, s' \in I_A$, and $u \in f[s] \cap I_B$, then there exists $u' \in f[s']$ such that there is a timed execution fragment from u to u' having the same timed visible actions as the given step.

Note that π is allowed to be the time-passage action in the third item of this definition. The most important fact about these simulations is that they imply admissible timed trace inclusion:

Theorem B.1 If there is a Simulation Mapping from Timed Automaton A to Timed Automaton B, with respect to any invariants, then $attraces(A) \subseteq attraces(B)$.