Implementation of
Timed Abstract State Machines
with Instantaneous Actions
by Machines with Delays

Joëlle Cohen  Anatol Slissenko

January 2008
TR–LACL–2008–02
Laboratory of Algorithmics, Complexity and Logic (LACL)
University Paris 12 (Paris Est)


J. Cohen, A. Slissenko.

Implementation of Timed Abstract State Machines with Instantaneous Actions
by Machines with Delays

Implementation
of Timed Abstract State Machines
with Instantaneous Actions
by Machines with Delays

Joëlle Cohen1
Anatol Slissenko1♯

Laboratory of Algorithmics, Complexity and Logic (LACL)
University Paris 12 (Paris Est), France

Abstract

We define an implementation of real-time reactive Abstract State Machines (ASMs) with instantaneous actions by ASMs with delayed actions. The time is continuous, and time constraints are metric, more concretely, defined by linear inequalities. It is clear that not every machine of the first type can be implemented by a machine of the second type. Because of the delays the runs of the initial machine can be represented in an implementation only approximately. We define a natural, straightforward syntactical transformation of an ASM with instantaneous actions into a machine with delayed actions, and give sufficient conditions on the ASM with instantaneous actions which guarantee that our transformation gives an implementation with a given approximation of runs. This means that there is an approximate correspondence between runs of both machines. The approximation is measured by 2 parameters, one for approximating time instants of updates and the other to approximate the values of real-valued functions. In the conclusion we discuss open questions, in particular the property preservation when implementing ASMs with instantaneous actions by ASMs with delayed actions. One may see our result as a first step towards solving a more general problem: under what conditions a bounded desynchronization of a real-time program approximately preserves its runs.

1 Introduction

Motivation. The problem we study is motivated by the passage from an abstract specification of a real-time distributed reactive algorithm to a more realistic specification. The real time that we consider is ‘hard’, i.e., it is defined by metric constraints. We limit our constraints to linear inequalities. The time is continuous; it is a very intuitive and useful abstraction. The problem persists also for discrete time, however for this case it is simpler, so we do not discuss it.

We consider two levels of abstraction: one with instantaneous actions and the other with delayed actions. The first one corresponds to the initial ‘high level’ specification, the second one to what we call implementation. Our specifications are formulated within the framework of Abstract State Machines (ASMs) of Y. Gurevich [Gur95, Gur00]. The ‘high level’ ASMs that we
consider are syntactically simple ASMs that have, nevertheless, the general expressibility and are sufficient to support the *Sequential ASM Thesis* [Gur00] (i.e., each state based algorithms without growing number of processes can be isomorphically modeled by such an ASM). These “sequential” ASM can be used to specify distributed algorithms with a fixed number of processes — such a machine acts by transitions of its global state that can be distributed between many processes in the reality that we model. As we study timed ASM and related implementation issues, we refer to [GH96] that pioneered the subject of timed ASM, and [BGR95], where two levels of abstraction are discussed. The lower level specification with delayed actions ‘implements’ the higher level specification. This paper [BGR95] contains many subtle observations related to this kind of ‘implementation’, however, there is no real-time in the setting studied in [BGR95].

Our main result gives sufficient conditions when a simple syntactic transformation provides an implementation of a high level machine with instantaneous actions by a machine with delayed actions. That means that each run of the implementation approximates a run of the initial specification. In fact we prove that this correspondence between runs goes also in other direction, namely, that for each run of the initial machine there is a run of the implementation that approximates this run. This gives some kind of *approximate bisimulation*. The construction of the implementation is quite straightforward, and seems to be the first natural one that comes to mind when one tries to find such a construction. Notice that if we extend this construction for general ASMs (that we do not do in this paper) then purely consecutive programs remain in some way intact, and for this case implementation conditions become much simpler. But our premier goal is to study the case where time and parallelism intertangle.

It is intuitively evident, that not each machine with instantaneous actions can be implemented as a machine with delayed actions. When we speak about delayed actions we mean the following framework (one formalization is given in [CS00]; we follow the ideas used in this paper). The delays are non-deterministic and bounded. Compare instantaneous actions with delayed ones. A machine with instantaneous actions fires the updates (assignments) of its functions immediately after having arrived at a possibility to fire (we tacitly use the semantics developed in [BS02b]). For example, an operator \( \textbf{if } G \textbf{ then } u := \theta \) acts as follows. Suppose that at an instant \( t \) the guard \( G \) becomes true (‘becomes’ means that it was false to the left of \( t \) within some positive duration). Then the nullary function (variable in terms of programming) \( u \) takes the value of \( \theta \) calculated from the values at \( t \) at the instants immediately to the right of \( t \). In the case of delayed actions the same operator acts differently. Take again an instant \( t \) where \( G \) becomes true. Though \( G \) becomes true at \( t \), the machine with delayed actions perceives it at some instant \( t' \) such that \( t < t' \); one may think that it takes some time to read the values of functions that are used in \( \theta \), and moreover, these values are read not simultaneously. Finally, \( u \) acquires a new value at some instant \( t'' > t' \). Bounded delays mean that \( (t'' - t) \) is bounded from above by a constant.

We do not go into details of delays, and summarize all of them into one constant \( \xi \) that guarantees that the updates of one step of the initial machine will be accomplished within a delay not greater than \( \xi \) from the beginning of the processing (this will be made precise in section 2.3). Notice that we assume that actions themselves are instantaneous, so at each instant the values of all functions are defined.

As the time is explicitly involved, and as some functions (variables of programming) have real values, including time, the updated values of \( u \) in two update processes described just above, may be different and may give different results for \( u \). Under some conditions, the results are close. However, in the general case, the error may be accumulated and finally may completely destroy
the desired correspondence between runs. A more evident cause of losing such correspondence is a bad separation of instants of changes of inputs. We consider reactive systems. In this paper the inputs are predicates (that constitutes rather general practical case — see the introduction to section 2). If the duration of a time interval where an input is true (or false), is smaller than the mentioned $\xi$, the machine with delayed actions may miss this change of the input.

Another difficulty comes from the fact that machines model distributed systems. Several updates may happen at a same time instant in our initial ‘high level’ machine. One can say that in the high level specification the processes are ideally synchronized from the viewpoint of its answer to changes of inputs, time included. In the implementation they are synchronized only approximately. But even a minor desynchronization is sufficient to destroy the desired correspondence between the runs.

Notice that machines with instantaneous actions can as well model partially synchronized distributed algorithms like, for example, clock synchronization. In such a model the non-deterministic delays are specified as a special input. Sure, we can specify such an algorithm directly as a machine with delayed action. But the desynchronization that we address is of another nature. It is a desynchronization that is necessarily present in real implementations and that is a nuisance for the developer at the initial stage of design when he/she should resolve more difficult, domain specific problems that do not depend much on details of implementation.

Our Result. We introduce a class of ASM with instantaneous actions ($IA-ASM$) that we call sturdy$^2$, a class of ASM with delayed actions ($DA-ASM$) and a transformation $\mathcal{I}$ of an $IA-ASM$ into $DA-ASM$. The class of sturdy machines has as parameter a precision of approximation of runs. This parameter is a pair, the first component bounds the shift of time instants of updates, and the second one bounds the deviation of values of real-valued functions. This pair of parameters determines a delay that is sufficient to ensure that our transformation bisimilarly implements with the given precision any sturdy $IA-ASM$ $A_0$ by $DA-ASM$ $\mathcal{I}(A_0)$. This notion of sturdy $IA-ASM$ together with the theorem that states the possibility to implement it, constitutes our main result (see Main Theorem in section 5.1). The set of constraints is in some way minimal, namely if we draw out at least one of them, our construction does not always give implementation.

Related Results. We are not aware of results that are really relevant to this of ours. Recall that our result is about sufficient conditions that permit to implement approximately an ideally synchronized algorithm with metric real-time constraints and interacting processes by an algorithms that is synchronized only approximately. There is a vast amount of papers that speak about refinements of timed algorithms or about implementing an algorithm with exact timing by an algorithm with approximate timing. However, no one, to our knowledge, deals with the entire framework as we have just described.

If we consider refinements that preserve the time instants then the time does pose particular problems. We cited such papers in [CS00, CS03]. More references on this subject (that is not relevant to our study), as well as an analysis of the timing of state machines can be found in [GP07].

Robustness of sequential algorithms with respect to the errors in synchronization (and with rather limited time constraints), was a subject of many papers with engineering flavor, though some of them treat such questions in a rigorous manner. We give 2 examples of such papers. Paper [Smi99] describes several refinement operations for sequential programs in terms of speci-

$^2$We avoid the term “robust”, that is linguistically quite adequate, because it is used in real-time systems engineering in some other sense
fication language Z that take into account time errors. The framework is much less general than that of ours, in particular the vocabularies are less rich (e.g., there are no real-valued functions). The paper deals mainly with preservation of simple properties in this situation.

In [DDMR04] the authors consider implementation of a timed automaton with usual semantics by a timed automata with what they call Almost ASAP semantics, which is parameterized by the reaction delay of the controller. Remark that the timed automata has no interacting parallel processes, and are much less expressive than ASMs. The question under study in [DDMR04] is the decidability of the existence of such implementations. As we deal with the general notion of algorithm and our conditions are semantic, we can not count on the decidability of such a property for a more or less general ASMs.

Structure of the paper. In Section 2 we describe the ASMs that we consider. The presentation is not quite formal but sufficiently rigorous. Approximate implementation and approximate bisimulation are defined in Section 3. Then in Section 4 we introduce our notion of sturdy machines. Section 5 contains the Main Theorem and its proof. In Conclusion (section 6) we discuss some open questions. We use the symbol □ to mark the end of definition, notation, assumption, remark, example, and we use the symbol ■ to mark the end of proof.

2 Real-Time Abstract State Machines

In this section we describe the two types of abstract state machines that we exploit. Our description is sufficiently precise though not quite formal.

Any ASM we consider, consists of a vocabulary, initial values and a program. A vocabulary consists of sorts (atomic types) and functions. The sorts and functions may be pre-interpreted or abstract. The pre-interpreted part has a fixed interpretation, and in our case concerns mathematical objects and functions that are used in ASMs. In the ASMs we use, this part is constituted by Boolean values and related functions, by real numbers and functions that permit to manipulate with linear sums and linear inequalities. Remark that if the initial values of functions and the coefficients of sums are rational numbers then in fact, we deal only with rational numbers, though we speak about reals (it is similar to the classical theory of real addition where only rational numbers are expressible).

The abstract part of the vocabulary consists of functions that are related to the proper functioning of the ASM. We deal with implementations of non-parametric ASMs 4 there will be no abstract sorts and consequently no functions with arguments. In other words our ASMs will have only nullary (with zero arguments) functions that are called variables in programming. All abstract functions are dynamic, i.e., they change with time. They are divided into inputs and internal functions. The inputs are given, they are external. The internal functions are calculated by the ASM.

Our ASMs have 2 pre-interpreted sorts: Bool of Boolean values and R that of reals. The continuous time T, that is usually defined as $\mathbb{R}_{\geq 0}$, in our case is defined as $\mathbb{R}$ for technical reasons, just to simplify the proof of the Main Theorem. Thus $-\infty$ is the minimal time instant.

A short text “High Complexity and Diagonal Algorithms are Irrelevant to Computational Practice of Nowadays” at www.univ-paris12.fr/laci/slissenko/Miscellaneous/ argues that negative results like undecidability, are irrelevant to real computations.

4Implementations of parametric ASMs are of considerable interest, but they are more difficult to study. An example of a parametric ASM is in Figure 2 of Example 1. All executable programs in computers are not parametric.
All the abstract functions of any ASM we consider are constant on \((-\infty, 0]\) with the values equal to their initial values.

\textbf{Remark 1.} A puristic framework (e.g., see [BS06]) demands to distinguish a sort and its interpretation. For example, a sort of time \(T\) is a symbol, and its interpretation \(T\) is a set. In our context such a purism is excessive. \(\square\)

The sort of reals is supported by pre-interpreted functions: rational constants, addition, subtraction, unary multiplications by rational constants, order relations over reals. One special pre-interpreted function is dynamic, namely, it is \(CT: \rightarrow \mathbb{R}\) (Current Time). This function is interpreted as identity in runs of ASM: at an instant \(t\) its value is \(t\).

As for dynamic abstract functions, we assume that the inputs are predicates, and the internal functions are either predicates or real-valued functions.

By limiting inputs to predicates we loose the generality; we do it to avoid technical complications that seem not very productive at the present stage of our study. However, inputs as predicates, moreover piecewise constant as we consider (see Assumption 1), cover a good amount of applications. A controller is usually not interested in the value of a real-valued input but only in what of a fixed number of intervals this value lies. The latter can be easily reduced to a finite number of input predicates.

We use internal real-valued functions that, semantically speaking, are linear combinations of the form \(\varphi = c_0 \cdot CT + c_1 \cdot f_1 + \cdots + c_q \cdot f_q\), where \(c_i\) are rational numbers, and \(f_i\) are internal real-valued functions, \((i = 1, \ldots, q)\). During an ASM execution the values of \(CT\) and \(f_i\) change with time; The changes of \(f_i\) are due to updates. So different updates of \(\varphi\) give different values of \(\varphi\). However, these changes are done in discrete, separated instants of time. And \(\varphi\) is represented in ASM as a list of \(c_i\) and the values of \(f_i\). This list remains unchanged between update instants of \(\varphi\). From this viewpoint internal real-valued functions are piecewise constant.

\subsection*{2.1 States of Real-Time Abstract State Machines}

The type of vocabularies of ASM that we consider is summarized in Definition 1.

\textbf{Definition 1 (ASM vocabulary)} An ASM vocabulary is composed of:

- \textbf{Pre-interpreted sorts}: reals (interpreted as \(\mathbb{R}\)), time (sometimes denoted by \(T\) but here interpreted as \(\mathbb{R}\)), boolean values (interpreted as \(\text{Bool} = \{\text{true}, \text{false}\}\)).

- \textbf{Functions}:
  - \textit{Pre-interpreted functions}:
    * \textit{Constants}: Boolean values \(\text{true}, \text{false}\) (each of type \(\rightarrow \text{Bool}\)), and rational numbers \(\mathbb{Q}\) (each of type \(\rightarrow \mathbb{R}\)).
    * \textit{Arithmetical operations}: addition (+), subtraction (−) of reals, and unary multiplications by rational numbers.
    * \textit{Arithmetical relations}: usual order relations over reals (\(=, <, \leq, >, \geq\)).
    * \textit{Current time}: \(CT: \rightarrow \mathbb{R}\)
    * \textit{Boolean operations}: \(\land, \lor, \neg\).
  - \textit{Abstract functions}:
Nullary function symbols of type $\rightarrow \mathbb{R}$.

Nullary predicate symbols of type $\rightarrow \text{Bool}$.

**Definition 2 (Term, external and internal literal, guard)** Within an ASM vocabulary (as well as within any logic vocabulary):

- **Terms** are: variables, constants, and their linear combinations with rational coefficients (the latter are arithmetical terms).
- **Atomic formulas** are: linear inequalities (we exclude equalities for technical reasons) and predicates (recall that all abstract functions and predicates are nullary).
- A **literal** is an atomic formula or its negation, though the latter case will be excluded in the ASM we consider, see Definition 5.
- A literal containing neither CT nor inputs is an **internal literal**, the other ones are **external literals**. In our context, an internal literal is either an internal predicate or an inequality not containing CT; input predicates and inequalities containing CT constitute the set of external literals.
- A **guard** is a propositional formula. (Predicate formulas have no sense in our context, as there are no variables to quantify.)

- $f \in \mathcal{H}$, where $f$ is a function and $\mathcal{H}$ is a formula, term, set of formulas or set of terms, means that $f$ occurs in one of the expressions constituting $\mathcal{H}$. In such a context, an ASM is considered as its program, and the latter is considered as a term.

In our case a negation of an inequality is again inequality. Remark that negative literals $\neg P$ can be avoided in ASM descriptions. For each predicate $P$ of the vocabulary of a given ASM we introduce a new, ‘dual’ predicate $\overline{P}$ such that $\overline{P} \leftrightarrow P$, and replace $\neg P$ by $\overline{P}$. The constraint $\overline{P} \leftrightarrow P$ should be added to the properties we consider if we care about them. After replacing negations by new atoms we have to add to our ASM rules that treat them (remark that we do not exclude boolean constants in updates). We will comment on this below, after Definition 8.

**Assumption 1 (Admissible interpretations).** Inputs are assumed to be predicates that are piecewise constant on left-closed, right-open intervals. The interpretation of pre-interpreted sorts and functions is fixed. These constraints and the definition of semantics of our ASMs given below (described in subsequent subsections, see also [BS02b, CS00] for formal details) define which interpretations are admissible. According to our semantics, the internal functions are also piecewise constant but on left-open, right-closed (or right-infinite) intervals.

Usually, a state of an ASM is an admissible interpretation, though not every admissible interpretation necessarily appears as a state in the executions of the ASM. We need some additional information to be appended to states.

We assume that each real-valued internal dynamic function has a special value that is called its default value, and it is regularly reset to its default value. This is made precise in Assumption 6 and the definitions that follow this assumption. For a given execution and a time instant, the number of updates of a given real-valued dynamic internal function after its last reset to its default value is called update number of this function at this instant in this execution, and is appended in the representation of the state.

**Definition 3 (State of an ASM).** A state of an ASM is any admissible interpretation of its vocabulary together with update numbers attributed to real-valued internal dynamic functions. Formally, we can speak not about update numbers, but about natural numbers (without any comment).
A semantics of an ASM $A$ defines, for a given (interpretation of) inputs an execution of an ASM that is also called run. Here we introduce some notations that use only the type of run as a mapping.

**Definition 4 (Type of run, instant $t^+$).** For a run $\rho$ of an ASM $A$ and a time instant $t$:

- A run $\rho$ of $A$ is *total* if it defined for all time instants.
- By $\rho(t^+)$ we denote the state $\rho(t)$ if there exists $\delta > 0$ such that there are no changes of values of internal functions of $A$ in time interval $[t, t + \delta)$. If for some $\delta > 0$ the values of internal functions at $t$ and in $(t, t + \delta)$ are different, and they do not change in $(t, t + \delta)$, then $\rho(t^+)$ denotes the state with these values of internal functions from $(t, t + \delta)$ and the values of inputs from $t$. (We will consider only runs with the latter property: if some functions are updated at $t$ then there exists the mentioned $\delta$.)
- For a formula $F(t_0, \ldots, t_{n-1})$, $n \geq 0$, with time parameters $t_0, \ldots, t_{n-1}$:

$$F(t_0^+, \ldots, t_{n-1}^+)=\exists \delta > 0 \forall \tau_0, \tau_{n-1} \in (0, \delta) F(t_0 + \tau_0, \ldots, t_{n-1} + \tau_{n-1}).$$

The following notations will be used for both types of ASM that we consider.

**Notations 1 (The value of formula over an interpretation).**

- $F[I]$ is the *value* of a formula $F$ in an interpretation $I$.
- $\varphi[\rho](t)=\varphi[\rho(t)]$ is the *value* of a term or a formula $\varphi$ in a run $\rho$ at an instant $t$.
- A phrase like “$\varphi[\rho](\sigma)$ is true”, where $\sigma$ is a set of time instants, means that “ for all $\tau \in \sigma$ there holds $\varphi[\rho](\tau)$”.

**Terminological convention.** Below by “real-valued” function we mean “internal dynamic real-valued function”. Thus, by default, the current time $CT$ is excluded when we speak about real-valued functions.

### 2.2 Real-Time Abstract State Machines with Instantaneous Actions (IA-ASM)

We start with ASMs with instantaneous actions for which we define the syntax and semantics. To illustrate the types of IA-ASMs we deal with and the whole problem of implementation, consider the following ASM in Figure 2 that represents a controller for the notorious Generalized Railroad Crossing Problem. This controller is a minor modification of the controller from [BS02b], that is in its turn a minor rectification of the controller from seminal paper [GH96]. The syntax is self-explanatory if to say (more comments will be given below) that

- $\theta := \theta'$ denotes an assignment (update). An assignment of a predicate $\Theta := true$ is abbreviated to simply $\Theta$, and assignment of a predicate $\Theta := false$ is abbreviated to $\neg \Theta$;
- all if-then-operators are executed synchronously in parallel;
- the block of if-then-operators turns around in an infinite repeat loop.
Example 1 (Railroad Crossing Controller: traditional IA-ASM approach). The situation to control is illustrated in Figure 1. The specification of the problem is as follows.

There are several tracks that constitute a set \( \text{Tracks} \) that is finite but of abstract cardinality (that is a parameter). We represent this set as \( \{0, 1, \ldots, N-1\} \), where \( N \) is an abstract constant. Without loss of generality, we assume that on all the tracks the trains go from left to right in Figure 1. The only abstract input is a predicate \( \text{cmg} : \text{Tracks} \to \text{Bool} \) (\textit{cmg} comes from \textit{coming}) such that \( \text{cmg}(x) \) is true when a train is detected on track \( x \). Denote the negation of \( \text{cmg} \) by \( \text{emp} \), i.e., \( \text{emp}(x) = \neg \text{cmg}(x) \) (\textit{emp} comes from \textit{empty}). In Figure 1 while a train is in the zone of control on a track \( x \) the signal \( \text{cmg}(x) \) remains true. We assume, as it was specified above, that \( \text{cmg}(x) \) is piecewise constant and preserves its value on intervals close on the left and open on the right.

There can be at most one train on a given track in the zone of control. However, there can be simultaneously several trains in this zone but on different tracks.

The controller does not see the real position of trains in the zone of control. It can detect only the instant of the incoming of a train on a track and the instant of its leaving. The speed of trains is bounded from above, and this bound is given as a positive time constant \( d_{\text{min}} \) that means that a train cannot reach the crossing before the duration \( d_{\text{min}} \) after having been detected. Otherwise, the speed of a train may vary. However, we are sure that if \( \text{cmg}(x) \) becomes true at an instant \( t \) ("become" signifies that \( \text{cmg}(x) \) is false in an interval of the form \([t - \Delta, t]\) with \( \Delta > 0 \) and is true at \( t \)), then it is true at least in time interval \([t, t + d_{\text{min}}]\).

The controller controls the opening and the closing of the gate. The gate is abstract, one for all tracks. In reality it may be a light or a barrier. The signal to control is \( \text{open} : \to \text{Bool} \). This is an output — an internal dynamic function used in the specification. The value \( \text{open} = \text{true} \) means that the signal to open the gate is enabled. We introduce \( \text{close}=\neg \text{open} \). The signal \( \text{open} \) is also piecewise constant but its intervals of constant values are left-open and right-closed (or right-infinite). If a gate is open at an instant \( t \) and a signal \( \text{close} \) is enabled starting from this instant then it may take up to \( d_{\text{close}} > 0 \) time to close the gate. So it is sure that the gate will be closed at some instant inside time interval \([t, t + d_{\text{close}}]\). In order to open the gate the signal \( \text{open} \) should be enabled during a time \( d_{\text{open}} > 0 \). We assume that \( d_{\text{close}}, d_{\text{open}} < d_{\text{min}} \).
The constant $d_{\text{open}}$ is not used in the controller, it is needed to make precise the requirements on functioning, namely Safety and Liveness. All the constants $d_{\text{min}}, d_{\text{close}}, d_{\text{open}}$ are abstract (nullary static functions of type $\rightarrow T$). So our specification is parametric with parameters $N$, $d_{\text{min}}, d_{\text{close}}, d_{\text{open}}$.

Here are the requirements on functioning formulated informally:

**(Safety)** When a train is in the crossing, the gate is closed.

**(Liveness)** The gate is open as long as possible (without violating Safety).

Besides the requirements on functioning, there are requirements on the environment (for details see [BS02b]). They concern the length of intervals wherein $\text{cmg}$ remains true, the form of the intervals where the values are constant etc. They are essential for the verification, but the latter is not our concern.

We do not formalize these requirements (they should be formulated in terms of inputs/outputs and constants) as it is not our goal to prove that our controller satisfies them.

The initial values of the functions are: $\text{cmg}(x) = \text{false}$ and $\text{dl}(x) = 0$ for all $x \in \text{Tracks}$, $\text{open} = \text{true}$ (thus, $\text{emp}(x) = \text{true}$, $\text{close} = \text{false}$), and $CT = 0$ (i.e. the first instant when the controller is launched, is zero).

To construct a controller we introduce one more internal function (by default, when we speak about functions of an ASM, we mean abstract dynamic functions). A function $dl : \text{Tracks} \rightarrow T$ ("$dl$" comes from "deadline") serves to memorize the instant after a detection of a train when the controller should enable the signal to close the gate. To calculate this instant the controller uses a constant $wt = d_{\text{f}}(d_{\text{min}} - d_{\text{close}})$ — the controller counts on the fastest train. The same function $dl$ will be used as a flag to detect the first instant of arrival of a train and the first instant of departure of a train; we will comment on it later.

In order to make a decision on closing or opening the gate the controller takes into account the situation on all tracks. This is done by the condition

$$\text{SafeToOpen} = \forall x \left( \text{emp}(x) \lor (\text{cmg}(x) \land \text{dl}(x) = 0) \lor CT < \text{dl}(x) \right).$$  \hfill (1)

In words, this formula $\text{SafeToOpen}$ describes situations when the signal $\text{open}$ can be and should be enabled ("can" refers to Safety, and "should" refers to Liveness). On any track $x$ either there is no train ($\text{emp}(x)$) or a train has just arrived ($\text{cmg}(x) \land \text{dl}(x) = 0$) or the detected train is still far from the crossing ($CT < \text{dl}(x)$). Remark that as initially $CT = 0$ and $\forall x \neg \text{cmg}(x)$, the condition $CT < \text{dl}(x)$ may become true only when $\text{dl}(x) > 0$. $\text{SafeToOpen}$ becomes false when on some track $x_0$ there holds $\text{cmg}(x_0)$ and $0 < \text{dl}(x_0) \leq CT$.

A controller as an IA-ASM $\mathcal{C}_0$ is in Figure 2.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>if $\text{cmg}(x) \land \text{dl}(x) = 0$ then $\text{dl}(x) := CT + wt$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>if $\text{emp}(x) \land \text{dl}(x) &gt; 0$ then $\text{dl}(x) := 0$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>if $\text{open} \land \neg \text{SafeToOpen}$ then $\text{close}$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>if $\text{close} \land \text{SafeToOpen}$ then $\text{open}$</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2: Railroad Crossing Controller $\mathcal{C}_0$. 
On can prove that the controller $C_0$ verifies Safety and Liveness (it is rather difficult to do — see [BS02a]), though it is not essential in our context. (We may take as an illustration a controller that does not satisfy the properties.) However, this controller is not implementable — neither the program nor the environment are sufficiently sturdy to implement it. We will start to discuss this issue in Example 3.

As we remarked above the controller does not see the real position of a detected or undetected train. It reasons in terms of the fastest train that arrives in the crossing after $d_{\text{min}}$ time after having been detected. All our comments below deal, by default, with these fastest trains.

This IA-ASM $C_0$ executes an infinite repeat external loop where in each passage it verifies simultaneously all the guards (formulas between $\text{if}$ and $\text{then}$), and for those guards that are true it executes the updates (assignments after $\text{then}$). Remark that this is done for all tracks simultaneously. There are no delays, if a guard is true at an instant $t$, it fires, and the corresponding update takes place immediately after $t$, i.e., at $t^+$. For example, if $(\text{cmg}(0) \land \text{dl}(0) = 0)$ becomes true at $t$ (i.e., it is true at $t$ and false just before $t$) then $\text{dl}(x) = t + wt$ on an interval $(t, T]$ with $T \geq (t + d_{\text{min}})$.

So line 1 of $C_0$ describes a way to detect an arrival of a train on a track $x$ and to calculate the instant $\text{dl}(x)$ when the fastest train dangerously approaches the crossing. At this instant taken over all tracks the signal to close the gate should be enabled; this is done in line 3. Line 2 resets $\text{dl}(x)$ to its initial value when the detected train leaves the track $x$. Line 4 executes an opening of the gate whenever it is possible. Remark that the controller $C_0$ tries to open the gate even if the available time is less than $d_{\text{open}}$. This is done for formal optimization of the time of the availability of the crossing in Liveness. But this is not implementable as we will see in Example 9.

**Remark 2 (SafeToOpen in the context of close).** As we will transform our ASM in some canonical form, we need some precisions on the usage of $\text{SafeToOpen}$ in the context of $C_0$. Suppose that line 4 fires at an instant $t_0$. Look at the situation just before this instant. We see $\neg \text{open}$, i.e., the signal to close the gate is running. This means that on some set of tracks $S$ trains are dangerously approaching the crossing, and clearly, $\text{cmg}(x)$ is still true for $x \in S$. How can $\text{SafeToOpen}$ become true in this situation? If it happens then the trains that were dangerously approaching the crossing have left, i.e., there holds $\text{emp}(x)$ for $x \in S$ but on the other tracks either there are no trains or the trains are still far from the crossing. Thus, the guard of line 4 of $C_0$ can fire when on all dangerous tracks $\text{emp}(x)$ become true. It cannot fire only because of $CT < \text{dl}(x)$ (moreover, this formula has no first instant where it becomes true what we do not admit). Thus, there are situations that are implicitly present in $\text{SafeToOpen}$ but that cannot happen in the context of true $\text{close}$. It does not influence the correctness of $C_0$, but in the canonical form there may appear useless guards that may become disturbing parasites in the implementation. We will come back to this issue in Example 2. □

The ASM $C_0$ of Example 1 is parametric, as we mentioned above. An implementation as executable code cannot have parameters. To implement an ASM we have to interpret the abstract sorts and external abstract functions. Thus, there is no $\text{forall}$ constructor in the ASMs we consider. Similar for abstract constants, they should be interpreted. Within these constraints we do not need functions with abstract values — they can be replaced by predicates; for example a function with 4 values can be represented by 2 predicates. This justifies that we consider only real-valued internal functions without arguments, i.e., functions of the type $\rightarrow \mathbb{R}$. In many cases, in particular in the case of Generalized Railroad Crossing, all external dynamic functions
are predicates. We assume this for technical simplicity — more general case is technically more involved, and needs a special study.

**Definition 5 (IA-ASM with canonical syntax).** The general form of an IA-ASM that we consider is in Figure 3. We will sometimes refer to its form as *canonical*.

```plaintext
if \( G_1 \) then \( A_1 \)
if \( G_2 \) then \( A_2 \)
·
if \( G_k \) then \( A_k \)
```

Here \( G_l, 1 \leq l \leq k \), are guards that we assume to be conjunctions of positive literals, and \( A_l \) are assignments (that will be called *updates* according to the ASM terminology). Each update \( A_l \) is assumed to be of the form \( \alpha_l := \beta_l \) with \( \alpha_l \) being an internal function and \( \beta_l \) being a nullary predicate or arithmetical term, i.e., a sum of real-valued functions and \( CT \) with rational coefficients.

Figure 3: IA-ASM \( A \).

All our constraints, namely only positive literals in guards, guards as conjunctions and only one update after *then*, does not diminish the generality. Any IA-ASM \( A \) with arbitrary propositional guards and multiple parallel synchronous updates after *then* can be transformed into an equivalent ASM of the form of \( A \); it will be commented after we explain the semantics of IA-ASMs just below, see also Figure 4 that shows the results of a transformation of \( C_0 \) into this form.

Given an input and initial values at instant 0 (and thus, in \((-\infty, 0]\)), the run (that is unique) of an ASM \( A \) is defined in the following way. In an infinite loop, we check simultaneously all the guards, and for those that are true we execute, again simultaneously, the corresponding updates. If at a certain instant whereat guards are true, the updates are inconsistent then the run is undefined after this instant. As the inputs are piecewise constant on left-closed right-open intervals, then the internal functions are piecewise constant on left-open right-closed (or right-infinite) intervals. In other words, if a guard is true at \( t \), and it fires an update of a function \( f \) that changes its value from \( v \) to \( w \), then \( f \) has value \( v \) on some interval \((t_0, t]\), and has value \( w \) on some \((t, t_1]\) or on \((t, \infty)\).

This explanation of the semantic is sufficient for our purposes, a detailed definition can be found in [BS02b]. We summarize the form of the interval for inputs, interval functions and guards in the following Definition 6.

**Definition 6 (Summary of IA-ASM semantics).** The semantics of IA-ASM is defined in such a way that in any run
- each input predicate preserves its value on intervals closed to the left, open to the right,
- each internal function preserves its value on intervals open to the left, closed to the right,
- a guard can become true at some instant \( t \) only if all its literals are true at \( t \) (recall that we consider guards that are conjunctions of literals), and there is a literal that is false in some \((t - \delta, t)\). In other words, we exclude the situations when a guard is true in an interval open to the left, but not at the left end of this interval. Or formulating this property differently, if after some instant \( T \) there is an instant whereat a guard becomes true then there is the first instant after \( T \) whereat it happens. □

**Definition 7 (Firing guards, fired updates).** For an ASM \( A \), its run \( \rho \), a time instant \( t \) and a guard \( G \) of \( A \):
• A guard of an IA-ASM that is true at \( t \) in \( \rho \), is a guard that fires or is a firing guard at \( t \) (there may be several firing guards at \( t \)).
  • Each if-then-operator of \( \mathcal{A} \) where \( G \) stands in if-part, is \( G \)-operator.
  • For \( G \) that fires at \( t \), the updates of all \( G \)-operators are updates fired by \( G \) at \( t \) in \( \rho \). The functions in then-part of these \( G \)-operators are functions updated by (firing) \( G \) at \( t \) in \( \rho \).
  • For a function \( f \) that is updated at \( t \) in \( \rho \), the instant \( t \) is an instant of update or update instant of \( f \) in \( \rho \).

An ASM of the form \( \mathcal{A} \) in Figure 3 with the just explained semantics is IA-ASM.

Remark that the set of admissible inputs, as well as the set of admissible initial values, is a 'hidden' parameter of the notion of IA-ASM.

Assumption 2 (Total IA-ASM). We will consider only IA-ASMs \( \mathcal{A} \) such that for any input and for any initial values, \( \mathcal{A} \) has a total run without accumulation of update instants.

Remark 3 (Firing is governed by inputs). Each guard that fires (in reasonable programs each guard fires in some run) contains an external literal. Indeed, let \( G \) be a guard that fires at some instant \( t \). If we suppose that it consists of only internal literals, then in order to fire, its literals must be updates by some other guard. But internal literals are true in intervals open to the left. Thus, there is no first instant where \( G \) becomes true with these values, and we consider only IA-ASM where guards become true at a well defined instant of time.

Now within the presented semantics, it is clear that a transformation that replace an if-then-operator

\[
\text{if } G \lor G' \text{ then } [x := \theta, x' := \theta']
\]

where \([\ldots, \ldots]\) is a synchronous parallel composition, by if-then-operators

\[
\begin{align*}
\text{if } G \text{ then } & x := \theta \\
\text{if } G' \text{ then } & x := \theta \\
\text{if } G \text{ then } & x' := \theta' \\
\text{if } G' \text{ then } & x' := \theta'
\end{align*}
\]

preserves the equivalence of IA-ASMs\(^5\). Thus, the number of updates after then can be reduced to one.

Remark 4 (On the role of constraints in canonical form). The role of constraints in canonical form IA-ASM (Definition 5) is different. The form of guards (conjunction of positive literals) simplifies some important notions. On the other hand, one update for each line is not important and plays a minor technical role. When transforming a complicated guard into disjunctive normal form we can arrive at disjuncts (conjunctions of literals) that never fire, see Remark 2. Moreover, such parasite guards may not satisfy our constraint that if a guard becomes true, it becomes true at a well defined instant (not as in \( 5 < CT \)). We will illustrate this in Example 2.

Assumption 3 (No guards that never fire). We assume that in each IA-ASM that we consider there are no guards that never fire. More formally for each guard there is a run wherein this guard fires at least once.

\(^5\)The equivalence, clearly, means that the projection of any run of the transformed ASM onto the vocabulary of the initial ASM is a run of the initial ASM.
Definition 8 (Canonical IA-ASM). An IA-ASM is canonical if it has canonical syntax (see Definition 5), has semantics of instantaneous actions summarized in Definition 6 and satisfies Assumption 3.

For the elimination of negation we make two transformations. For each input predicate we add a predicate that is equivalent to its negation, and redefine the set of inputs in the appropriate way. For each internal dynamic predicate $P$ we also introduce a predicate that we make equivalent to the negation of $P$ and add to the program updates that change it together with $P$ but by the negated Boolean value. More formally, we accompany each operator

\[
\text{if } G \text{ then } P := \text{true}
\]

by

\[
\text{if } G \text{ then } \overline{P} := \text{false},
\]

where $\overline{P}$ is a new predicate representing the negation of $P$.

Example 2 (Railroad Crossing controller $C_1$: canonical form of $C_0$). Figure 4 presents a controller $C_1$ that is the result of an application of the reductions described above to controller $C_0$. It is done for 2 tracks.

Now $emp$ and $close$ are new predicates related to $cmg$ and $open$ by the equivalences $(emp(x) \leftrightarrow \neg cmg(x))$, $x \in \text{Tracks}$, and $(close \leftrightarrow \neg open)$ respectively. Let $\text{Tracks} = \{0, 1\}$.

Look at $\text{SafeToOpen}$ and its negation for these 2 tracks.

\[
\text{SafeToOpen} =\]
\[
( emp(0) \lor (cmg(0) \land dl(0) = 0) \lor CT < dl(0) ) \land
\]
\[
( emp(1) \lor (cmg(1) \land dl(1) = 0) \lor CT < dl(1) ) \]

$\leftrightarrow$

\[
( emp(0) \land emp(1) ) \lor
\]
\[
( emp(0) \land (cmg(1) \land dl(1) = 0) ) \lor
\]
\[
( emp(1) \land (cmg(0) \land dl(0) = 0) ) \lor
\]
\[
( emp(0) \land CT < dl(1) ) \lor
\]
\[
( emp(1) \land CT < dl(0) ) \lor
\]
\[
( (cmg(0) \land dl(0) = 0) \land (cmg(1) \land dl(1) = 0) ) \lor
\]
\[
( (cmg(0) \land dl(0) = 0) \land CT < dl(1) ) \lor
\]
\[
( (cmg(1) \land dl(1) = 0) \land CT < dl(0) ) \lor
\]
\[
( CT < dl(0) ) \land CT < dl(1) ) \]

Recall that $\text{SafeToOpen}$ is used in conjunction with $close$ in $C_0$. Recall Remark 2. The last formula in the disjunctive normal form of $\text{SafeToOpen}$, namely the formula (10), can never fire in conjunction with $close$. Thus, it cannot be used in canonical IA-ASM (see Definition 8).

Consider formula (9) also in conjunction with $close$. Suppose that this guard, namely $(close \land (9))$ becomes true at an instant $t$. It means that the signal to close the gate is enabled, that a train has just arrived on track 1 (as $dl(1) = 0$ is still true), and there is a train on track 0 that has been processed (as $CT < dl(0)$ implies $dl(0) > 0$) but it is yet far from the instant when the gate should be closed. And in this case the controller send a signal to open the
gate. But it is clear that the track 1 was empty before \( t \), and thus the signal to open the gate had to be enabled before \( t \). Hence, this guard cannot fire. Similar for (8), it can fire neither.

Consider the guard \(( \text{close} \land (7))\), and suppose that it becomes true at some \( t \). This means that trains have just simultaneously arrived on both tracks but the signal to close the gate is running. Clearly, the tracks were empty before \( t \), and thus the signal to open the gate is enabled. So this guard cannot fire.

As for the remaining formulas (2)–(6), they represent the events that may happen in the context of enabled close. Formula \(( \text{close} \land (2))\) says that the trains that were on both tracks, leave them simultaneously. Formula \(( \text{close} \land (3))\) says that the train on track 0 has just left, and a train has just arrived on track 1. Formula \(( \text{close} \land (4))\) describes a similar situation where tracks are swapped round. Formula \(( \text{close} \land (5))\) says that the train on track 0 has just left, and the train on track 1 is far from the dangerous zone, and the controller can try to open the gate. Formula \(( \text{close} \land (6))\) describes a similar situation where tracks are swapped round.

Consider the negation of SafeToOpen (recall that it is used in \( C_0 \) in conjunction with enabled open).

\[
\neg \text{SafeToOpen} =
\neg \left[ \left( \text{emp}(0) \lor (\text{cmg}(0) \land \text{dl}(0) = 0) \lor CT < \text{dl}(0) \right) \land \\
\left( \text{emp}(1) \lor (\text{cmg}(1) \land \text{dl}(1) = 0) \lor CT < \text{dl}(1) \right) \right]
\leftrightarrow
\left( \text{cmg}(0) \land (\text{emp}(0) \lor \text{dl}(0) > 0) \land CT \geq \text{dl}(0) \right) \lor \\
\left( \text{cmg}(1) \land (\text{emp}(1) \lor \text{dl}(1) > 0) \land CT \geq \text{dl}(1) \right)
\leftrightarrow
\left( \text{cmg}(0) \land CT \geq \text{dl}(0) > 0 \right) \lor \\
\left( \text{cmg}(1) \land CT \geq \text{dl}(1) > 0 \right)
\text{(11)} \hspace{1cm} \text{(12)}
\]

The both formulas (11)–(12) represent realistic situations in conjunction with enabled open. The first one (11) says that there is a train on track 0, and it has just arrived at an instant when the controller should enable the signal to close the gate. The second formula (12) describes the same situation on the other track.

Now we can transform \( C_0 \) into canonical form. The result is a controller \( C_1 \) in Figure 4. To make this controller \( C_1 \) more readable we do not respect the constraint (by the way, not important and purely technical) that each if-then-operator fires only one update.

We summarize the constraints on the form of IA-ASMs in Assumption 4.

**Assumption 4 (Constraints on the form of IA-ASM)** Without loss of generality we consider IA-ASM such that:

- all literals of guards are positive (no negations);
- each guard is a conjunction of atomic formulas;
- in each if-then-operator there is only one update (in then-part).
2.3 Real-Time Abstract State Machines with Delayed Actions (DA-ASM)

IA-ASM is a convenient abstraction for the specification of real-time reactive controllers that permits to concentrate on the problem specific logic of the controller and its time constraints. However, the exact implementation of such a machine is hardly possible — better to say impossible, as we cannot ensure absolute synchronization within considerable parallelism. Moreover, the genuine parallelism is often non justifiable as it can be too costly, and we should implement it on a single processor.

Thus, it is reasonable to study implementations of IA-ASM by machines that are not absolutely synchronous, that have delays between actions. One would like, for example, to implement the Railroad Crossing controller \( \mathcal{C}_1 \) by a sequential machine. We will consider a more general implementation of IA-ASMs of the form \( \mathcal{A} \) by DA-ASM that takes into account various possibilities.

A **DA-ASM** is defined over a vocabulary of the same type as IA-ASM. The set of constructors of DA-ASM is richer than that of IA-ASM: updates (of the same type as for IA-ASM), sequential composition denoted here by \( \text{seq} \varphi_1, \ldots, \varphi_n \text{endseq} \), parallel composition denoted here by \( \text{par} \varphi_1, \ldots, \varphi_n \text{endpar} \), and loops.

There can be different ways to describe the delays. One can distinguish the delays attributed to different operations (like in \([CS00]\)) or take into account only delays of reading and writing. The latter is more reasonable. The only delay that is important in our context is a global delay of implementing one step of the initial IA-ASM, see Definition 11 below. By one step of the initial IA-ASM we mean here one execution of all \if\then\-operators.

There is one more technical detail concerning executions that should be concretized, namely, what is the form of intervals wherein an internal function remains unchanged. For the homogeneity with IA-ASM we assume that these interval are of the same form as stated in Assumption 9.

**Definition 9 (Constant value interval of DA-ASM runs).** The semantics of DA-ASM is defined in such a way that the intervals wherein an internal dynamic function does not change its value are left-open right-closed (or right-infinite). An internal dynamic function \( f \) changes its value to \( v \) at instant \( t \) means that \( f(\tau) = v \) in an interval \( (t, t') \) with \( t' < \infty \) or in an interval \( (t, \infty) \), and \( f(t) \neq v \).

We will need only one particular case of DA-ASM that describes our implementation of IA-ASM of the form \( \mathcal{A} \) in Figure 3. Denote by \( \mathcal{J} \) the syntactical transformation described below in

\[
\begin{align*}
1: \ & \text{if } \text{cmg}(0) \land \text{dl}(0) = 0 \text{ then } \text{dl}(0) := \text{CT} + \text{wt} \\
2: \ & \text{if } \text{cmg}(1) \land \text{dl}(1) = 0 \text{ then } \text{dl}(1) := \text{CT} + \text{wt} \\
3: \ & \text{if } \text{emp}(0) \land \text{dl}(0) > 0 \text{ then } \text{dl}(0) := 0 \\
4: \ & \text{if } \text{emp}(1) \land \text{dl}(1) = 0 \text{ then } \text{dl}(1) := 0 \\
5: \ & \text{if } \text{close} \land (\text{emp}(0) \land \text{emp}(1)) \text{ then } [\text{open}, \neg \text{close}] \\
6: \ & \text{if } \text{close} \land (\text{emp}(0) \land (\text{cmg}(1) \land \text{dl}(1) = 0)) \text{ then } [\text{open}, \neg \text{close}] \\
7: \ & \text{if } \text{close} \land (\text{emp}(1) \land (\text{cmg}(0) \land \text{dl}(0) = 0)) \text{ then } [\text{open}, \neg \text{close}] \\
8: \ & \text{if } \text{close} \land (\text{emp}(0) \land \text{CT} < \text{dl}(1)) \text{ then } [\text{open}, \neg \text{close}] \\
9: \ & \text{if } \text{close} \land (\text{emp}(1) \land \text{CT} < \text{dl}(0)) \text{ then } [\text{open}, \neg \text{close}] \\
10: \ & \text{if } \text{open} \land (\text{cmg}(0) \land \text{CT} \geq \text{dl}(0) > 0) \text{ then } [\text{close}, \neg \text{open}] \\
11: \ & \text{if } \text{open} \land (\text{cmg}(1) \land \text{CT} \geq \text{dl}(1) > 0) \text{ then } [\text{close}, \neg \text{open}]
\end{align*}
\]

Figure 4: Railroad Crossing Controller \( \mathcal{C}_1 \).
Definition 10 that transforms a given IA-ASM into a DA-ASM. The idea of this transformation is quite straightforward, namely, we save (Backup phase) the current values of functions, and then simulate one cycle of $A$ to update the functions (Update phase), see Figure 6.

**Definition 10 (Description of $I$).** Let $A$ be an IA-ASM of the form shown in Figure 3, and let $V$ be its vocabulary of dynamic functions. Denote by $\tilde{V}$ a vocabulary of new functions such that each $z \in V$ has a respective function $\tilde{z} \in \tilde{V}$ of the same type. Denote by $\tilde{CT}$ a new function that will be used to save $CT$.

The machine $I(A)$ is shown in Figure 5 and illustrated in Fig. 6. In fact, Figure 5 defines $I$.

\[
\begin{align*}
\text{repeat} \\
\quad \text{seq} \\
\quad \quad \text{forall } z \in V \\
\quad \quad \quad \text{par} \\
\quad \quad \quad \quad 1 : \quad [\tilde{z} := z, \ \tilde{CT} := CT] \\
\quad \quad \quad \quad \text{endpar}; \\
\quad \quad \quad \text{forall } l = 1 \cdots k \\
\quad \quad \quad \text{par} \\
\quad \quad \quad \quad 2 : \quad \text{if } \tilde{G}_l \text{ then } \alpha_l := \tilde{\beta}_l \text{ endif} \\
\quad \quad \quad \text{endseq} \\
\quad \text{endrepeat}
\end{align*}
\]

Figure 5: Implementation $I(A)$ of $A$

Figure 6: $I(A)$ control flow

Now we describe the semantics of $I(A)$. This machine has non-deterministic delays between actions as described above. Machine $I(A)$ works in two phases: Backup and Update, executed sequentially. During the first phase the machine $I(A)$ saves the state of $A$ in the new variables marked by tilde. This phase is executed in parallel, however because of delays, the backup may be a mixture of consecutive and simultaneous actions. The second phase simulates, in a similar parallel manner, the work of $A$ using the saved values. The guard $\tilde{G}_l$ and the term $\tilde{\beta}_l$ are defined in Notations 2 just below.

**Notations 2 (Tilde-formula).**

- $F$, where $F$ is a formula (a term), denotes the formula (the term) obtained from $F$ by replacing $z \in V$ by $\tilde{z} \in \tilde{V}$. Formula (term) $\tilde{F}$ is *tilde-image* or *tilde-version* of $F$.
- $\tilde{F}$, where $F$ is a set of formulas or terms, denotes the set of tilde-images of elements of $F$.

The only important delay of the implementation $I(A)$ is the *total time of one loop*. It is denoted by $\xi$. We do not care how this delay is distributed inside the Backup and Update phases.
Definition 11 (\(\xi\)-interval). A \(\xi\)-interval of a run of a DA-ASM \(\mathcal{I}(A)\) is a time interval that corresponds to the execution of one external \texttt{repeat}-loop of this DA-ASM. We mean that the length of this time interval is bounded by \(\xi\). The form of \(\xi\)-intervals is not important, as all the backup and update actions take place inside such interval. However, as all time instants should be covered, we assume for concreteness that \(\xi\)-interval are closed to the left and open to the right.

Clearly, if an input may change its value within a duration less that \(\xi\) then the machine \(\mathcal{I}(A)\) can miss some of input changes, and consequently, cannot do the corresponding update. Or it may execute an update that does not correspond to any update in \(A\). In fact, the situation is more subtle, and \(A\) should be very robust (we call the exact property “sturdy”) to admit an implementation that we have just described.

Definition 7 distinguishes time instants whereat guards of an IA-ASM fire, and the updates fired by this guard. The next Definition 12 draws attention to some particular time instants in executions of DA-ASM that appear in various contexts below, in particular, in lemmas and proofs.

Definition 12 (Evaluation and update instants). For a run \(\rho_1\) of an DA-ASM \(\mathcal{I}(A)\) we distinguish the following time instants that lie in a same \(\xi\)-interval (recall that the form of the intervals of internal functions of \(\mathcal{I}(A)\), wherein they preserve their values, is described in Definition 9):

- A function \(\tilde{f}\), where \(f\) is an internal dynamic function of \(A\), may acquire a new value in line 1 in run \(\rho_1\). The time instant of the update of \(\tilde{f}\) is a backup instant of \(\tilde{f}\) in \(\rho_1\) (recall that \(\tilde{f}\) acquires the new value immediately after this instant).
- A time instant whereat a guard \(\tilde{G}_l\) of \(\mathcal{I}(A)\) is evaluated to true in line 2 in run \(\rho_1\) is an evaluation instant of \(\tilde{G}_l\) in \(\rho_1\).
- A time instant whereat an internal function \(f\) of \(\mathcal{I}(A)\) is updated in run \(\rho_1\) by an assignment in line 2 of \(\mathcal{I}(A)\) with a guard \(\tilde{G}\) is an update instant of \(f\) in \(\rho_1\) fired by \(\tilde{G}\) or an instant whereat \(\tilde{G}\) fires in \(\rho_1\).
- For both types of ASM we say that a guard updates a function at a given instant if the function is updated at this instant by firing this guard.

We do not speak about instant whereat a tilde-guard becomes true in an \(\xi\)-interval, because a priori a tilde-guard may change its truth value several times during a backup phase. However, such behaviors will be excluded in our sturdy ASMs.

Definition 13 (A guard fires with given values of internal functions). Given

- an DA-ASM \(A_1 = \mathcal{I}(A_0)\), where \(A_0\) is an IA-ASM with the vocabulary \(W\) of internal dynamic functions,
- a run \(\rho_1\) of \(A_1\),
- a state \(S\) over the vocabulary of \(A_1\) or \(A_0\),
- a guard \(\tilde{G}\) of \(A_1\) that fires in \(\rho_1\) at an instant \(t\),

we say that \(\tilde{G}\) fires (and thus, is true) in \(\rho_1\) at \(t\) with the values of internal functions from \(S\) if \(\tilde{f}[\rho_1(t)] = f[S]\) for all \(f \in W\) occurring in \(G\).

Here the notation \(\tilde{W}\) has the same meaning for internal dynamic functions of \(A_0\) as \(\tilde{V}\) in Definition 10 for all dynamic functions of \(A_0\).
Remark 5 (Update instant of a guard versus its true instants). Consider a DA-ASM $A_1 = \mathcal{I}(A_0)$, where $A_0$ is an IA-ASM, and any run $\rho_1$ of $A_1$. Let $\zeta_1$ and $\zeta_2$ be two consecutive $\xi$-intervals of $\rho_1$, $\zeta_1^{(r)} \leq \zeta_2^{(l)}$, and suppose that a guard $\tilde{G}$ fires in $\zeta_2$. The truth value of $\tilde{G}$ in $\rho_1$ is intact between the end of the backup phase in $\zeta_1$ and the beginning of the update phase in $\zeta_2$ (recall that tilde-functions are updated during the backup phases). The update phase changes the values of functions but not the values of their tilde-images. The truth value of $\tilde{G}$ may change during the update phase.

If $G$ becomes true in $\rho_1$ at $t$ and remains true until the nearest update phase of $A_1$ in $\rho_1$ then $\tilde{G}$ fires strictly before $t + 2\xi$. Moreover, the $\xi$-interval wherein $\tilde{G}$ fires at this instant, also lies to the left of $t + 2\xi$. The bound $2\xi$ (not better) is justified by the following observation. The update phase of $\zeta_1$ may terminate by an instant ‘very’ close to the left end of $\zeta_1$, and $G$ may become true immediately after. On the other hand, the next backup phase may start at an instant ‘very’ close to the right end of $\zeta_2$.

A priori, $\tilde{G}$ may become, or even remain, true and fire in two consecutive in $\xi$-intervals of $\rho_1$. However, we will exclude this possibility in our sturdy ASMs. If the mentioned situation may not appear, then the instant whereat $\tilde{G}$ fires in $\zeta_2$ and the nearest to the left instant whereat $\tilde{G}$ becomes true, are $2\xi$-close.

Another unpleasant situation that may happen in the general case is that the guard $G$ itself, not its tilde-image, may remain false in both $\zeta_1$ and $\zeta_2$. It can result from the fact that $\tilde{G}$ uses for its evaluation the value of an input that becomes false at the instant of backup of the time but is true at the instant of its backup. For the evaluation of $G$ the values of all its functions are taken at the same instant. Again, we will exclude such situations in sturdy ASMs.

Example 3 (Obstacles to implementing the Railroad Crossing controller $\mathcal{C}_1$). The program $\tilde{\mathcal{C}}_1 = \mathcal{I}(\mathcal{C}_1)$ is too cumbersome to be instructive in itself, so we discuss some of its features without writing it down.

First of all we can notice that because of delays the implementation may not satisfy the requirements.

Remark 6 (Safety is not preserved in implementation). What is dangerous for safety of $\tilde{\mathcal{C}}_1$ is $wt$ taken intact from the IA-ASM specification. The arrival of a train on a track is detected with some lateness, and thus, the signal to close the gate is sent with a delay as compared with $\mathcal{C}_1$. All this implies that the gate may be not closed when a train is in the crossing area. On the other hand, such a dangerous run is close to the corresponding run of $\mathcal{C}_1$. Thus, we do not care about properties. It is a different question that we will touch in Conclusion. We care only about approximate correspondence of runs.

Now we will discuss liveness issues; they are closely related to our main property of approximate correspondence between runs of the abstract machine and its implementation.

Obstacle 1. The instant of arrival of a train after a departure of a previous one on the same track can be unboundedly close to the instant of this departure. Thus, whatever be delay $\xi$ the controller $\tilde{\mathcal{C}}$ may miss this event and continue to work as if there were no departure. It is not dangerous for safety but clearly destroy liveness, as the gate remain closed in a situation where it can be opened. It is particularly evident, if the time to open the gate $d_{\text{open}}$ is much smaller than $wt$.

Notice that in the railroad reality the trains on the same track should be well separated. If we introduce a positive constant $d_{\text{sep}}$ that guarantees that between the departure of a train
and the arrival of the next one on the same track, there is a $d_{\text{sep}}$-length time interval, and if we choose $\xi$ sufficiently small with respect to $d_{\text{sep}}$, then this violation of liveness will be eliminated. In particular, we assume that there are no trains during $d_{\text{sep}}$ interval of time after the instant 0.

Thus, **Obstacle 1** can be overcome by imposing a quite realistic constraint on the environment: the length of intervals where $\text{emp}(x)$ is constant is not less than a positive constant $d_{\text{sep}}$. And the first train cannot appear on any track before the instant $d_{\text{sep}}$.

**Obstacle 2.** There is another, much more serious obstacle to implementing of $\mathcal{C}_1$. The issue we considered above for one track can arise for two different tracks. Indeed, consider the guard of line 8 of $\mathcal{C}_1$ (similar for line 9), and on order to relate is with a general situation illustrated in Figure 7, set $x = 1$, $y = 0$ and write the guard of line 8 as

$$G_8 = (\text{close} \land (\text{emp}(y) \land CT < d(x))).$$

This formula becomes true when the signal to close the gate is enabled, the train detected on track $y$ has just left, and there is a time to try to open the gate. But this time can be very small, smaller than the delay $\xi$. Indeed, take a run $\rho_0$ such that at instant $t_y$, the situation is as follows. On track $y$, the train, that was in the zone of control, leaves it, i.e., $\text{emp}(y)[\rho_0](t_y)$ becomes true at $t_y$, and $\text{cmg}(y)[\rho_0]([T_y, t_y])$ (i.e., $\text{cmg}(y)$ holds on a time interval $[T_y, t_y]$) with $T_y < t_y$. On the other hand, suppose that on the other track $d_l(x)[\rho_0](t_y) = t_y + z$ with some positive $z$ that can be arbitrary small. In this situation the guard $G_8$ remains true during $z$ time, and if $z$ is smaller than $\xi$ the controller with delays may miss the respective update.

More formally, this observation can be formulated in the following way. Let $\rho_0$ be a run, $T_x$ be an instant of arrival of a train on a track $x$, and $t_y$ be an instant of departure of a train from track $y$, $x \neq y$. To avoid the mentioned obstacle to the implementation mentioned above we should suppose (see Figure 7) that

$$T_x < t_y \leq (T_x + wt) \rightarrow ((T_x + wt) - t_y) > d_s, \quad (13)$$

where $d_s$ is a constant that should be sufficiently bigger than the delay $\xi$ of the implementation. The mentioned $z$ is $z = ((T_x + wt) - t_y)$. The value $(T_x + wt)$ is $d_l(x)$ defined by the guard of $\mathcal{C}_1$ in line 1 or in line 2 that becomes true at $T_x$.

![Figure 7: Illustration for formula (13)](image)

It is not at all convincing to impose the condition (13) on the environment, more concretely, on the arrival and departure of trains on different tracks.

Thus, **without a constraint on the environment alike the condition (13)** (that seems non-
realistic) the controllers $C_0$ and $C_1$ are not implementable.

Hence, we have to weaken the requirements that are too exigent. It is necessary to maintain safety, so we have to demand less of the availability of the crossing.

**Weakening Liveness.** We come back to the general case of arbitrary number of tracks. A simple way to solve the problem is the following requirement:

*(Weak Liveness)* If the gate is closed then open the gate only when all the trains leave the zone of control.

**Remark 7.** One can imagine a more subtle weakening of liveness. However, our analysis of $G_8$ above shows that one should be very cautious when playing with inequalities involving $CT$. We can try to analyze relations between $dl(x)$ for different $x$ to get a better Liveness. We do not go on with it. □

With this Weak Liveness the controller becomes much simpler. First replace $dl : Tracks \rightarrow \mathbb{T}$ by $dl : \rightarrow \mathbb{T}$. Below in Figure 8 we describe a controller $C_2$ in terms of general IA-ASMs, like $C_0$; $emp$ and $close$ are notations, not functions of the vocabulary, again like in $C_0$.

\[
\begin{align*}
\text{forall } x \\
1: & \quad \text{if } cmg(x) \land dl = 0 \text{ then } dl := CT + wt \\
2: & \quad \text{if } open \land CT \geq dl > 0 \text{ then } close \\
3: & \quad \text{if } close \land \forall y \ emp(y) \text{ then } [open, dl := 0]
\end{align*}
\]

Here $x$ and $y$ are variables for tracks.

Figure 8: Railroad Crossing Controller $C_2$.

This controller $C_2$ invokes again the separation of trains. The individual track separation does not suffice now as the controller reacts to the global departure of trains. So we impose, this time enough realistic, constraint on the environment:

**Track uniform separation of trains.** After any instant of departure of all trains from the zone of control the next one cannot come before $d_{sep}$ time. More formally, for any run $\rho_0$ and for any instant $t$ such that $\forall x \ emp(x)$ at $t$ in $\rho_0$ and $\neg \forall x \ emp(x)$ everywhere in $(t - \delta, t)$ in $\rho_0$ for some $\delta > 0$ there holds $\forall x \ emp(x)$ in $\rho_0$ for all $\tau \in [t, t + d_{sep})$.

However, we still have no implementable IA-ASM. We come back to this question later, in Example 9. □

Below we give some simple examples of obstacles to implementation. In these examples for better readability we present IA-ASM using the general syntax that we used in $C_1$ in Figure 4.

**Example 4 (IA-ASM, non implementable because of too dependable operators).**

Consider the following IA-ASM $D_0$:

\[
\begin{align*}
1: & \quad \text{if } I_0 \land \neg I_1 \land f_0 \land f_1 \text{ then } [\alpha_0, f_0 := false] \\
2: & \quad \text{if } \neg I_0 \land I_1 \land f_0 \land f_1 \text{ then } [\alpha_1, f_1 := false] \\
3: & \quad \text{if } I_0 = I_1 \land f_0 \neq f_1 \text{ then } [f_0 = f_1 := true]
\end{align*}
\]

Here $I_j$ ($j = 0, 1$) are inputs, and $f_j$ are flags (internal functions), the both of type $\rightarrow \mathbb{Bool}$. Letters $\alpha_j$ symbolize 2 different, incompatible actions, for example $x := 0$ and $x := 1$. Initially
$I_0 = I_1 = f_0 = f_1 = true$. We demand that the inputs remain unchanged on sufficiently long intervals, but the instants of change of two different inputs can be very close or even coincide.

This IA-ASM $D_0$ starts with equal inputs, and reacts to the first instant whereat this equality is broken. Depending on which input is false and what input is true, the ASM execute different incompatible actions.

Denote its implementation by $D_1 = d_I D_0$.

Consider a run, where IA-ASM $D_0$ is in the initial state before instant $t$, and at $t$ the both inputs swap (simultaneously) to false. In this case ASM $D_0$ does nothing at $t$. Contrary to IA-ASM $D_0$ its implementation by DA-ASM $D_1$ may behave differently. In the $\xi$-interval containing $t$ there may happen one of the following sequences of actions:

(a). $D_1$ saves the value of $I_0$ before $t$ and the value of $I_1$ after $t$,  
(b). $D_1$ saves the value of $I_1$ before $t$ and the value of $I_0$ after $t$,  
(c). $D_1$ saves the both values of $I_0$ and $I_1$ before $t$ or after $t$.

In case ((c)) the behavior of $D_1$ around $t$ coincides with the behavior of $D_0$. In case ((a)) the values caught by $D_1$ are $I_0$ and $\neg I_1$; thus, $D_1$ fires line 1. In case ((b)) the values caught by $D_1$ are $\neg I_0$ and $I_1$; thus, $D_1$ fires line 2. The latter two cases give runs that do not correspond to the run of $D_0$.

If the input makes a similar swap as above at two instants closer that $\xi$, and line 1 fires in $D_0$, then $D_1$ may miss it, and do nothing. Again there is no correspondence of runs. □

Example 5 (IA-ASM, non implementable because of accumulation of errors)

Consider the following 1-line IA-ASM $P_0$:

\[
\text{if } dl - CT \geq 1 \text{ then } \begin{cases} \text{out} := 1 - \text{out}, \\
\text{dl} := CT + 1 \end{cases}
\]

with internal functions $out : \rightarrow \{0, 1\}$ and $dl : \rightarrow \mathbb{R}$ whose initial values are $out = 0$ and $dl = 1$. The only input is $CT$. This IA-ASM outputs a periodic function with two values 0 and 1, and period 1. Denote its implementation by $P_1 = d_I P_0$.

Notice that if we take as guard $dl - CT = 1$ then the implementation will usually miss the instant when the guard should fire.

In a ‘typical’ run of $P_1$ we will have the following shift. For some positive $\delta$, smaller than $\xi$, DA-ASM $P_1$ fires the guard for the first time at an instant after $dl = 1 + \delta$. Thus, it computes $dl \geq (2 + \delta)$. With this $dl$ DA-ASM $P_1$ fires the guard for the second time at an instant after $dl = 2 + 2\delta$. Hence the next $dl \geq (3 + 2\delta)$. And so on, giving after some time $dl \geq (N + (N - 1)\delta)$. So the error in timing arrives at $(N - 1)\delta$, where $N$ grow up, and the deviation of a run of the implementation from the run of $P_0$ becomes arbitrary big. (For this reason, in real life we have to update our clocks taking values of a more precise clocks.) To avoid such an accumulation of errors the controllers from time to time should reset all its real-valued functions to some default value. □

Example 6 (IA-ASM, non implementable because of badly interacting shared variables) Consider the following 2-line IA-ASM $B_0$:

1. if $(x + CT) \geq (y + 1)$ then $[\alpha, x := y + 1]$
2: \[ \text{if } (y + 3) \geq (x + 5) \text{ then } [\beta, y := x + 1] \]

Here \( \alpha \) and \( \beta \) are two different compatible actions that may depend on \( x \) and \( y \), and \( x \) and \( y \) are two internal functions with initial values, say \( x = 0 \) and \( y = 2 \).

We are interested in the instant \( CT = 3 \). At this instant IA-ASM \( A_0 \) its guards become true:
\[
(x + CT) \geq (y + 1) \iff (0 + 3) \geq (2 + 1) \iff true, \quad (y + 3) \geq (x + 5) \iff (2 + 3) \geq (0 + 5) \iff true.
\]
Thus, at instant 3 IA-ASM \( A_0 \) fires and updates its internal functions to get \( x = 3 \) and \( y = 1 \), and it executes \( \alpha \) and \( \beta \) with initial values of \( x \) and \( y \).

Denote \( B_1 = \omega J(A_0) \). There is a run of \( B_1 \) where the instant 3 is inside some \( \xi \)-interval, and the current time is saved in \( CT \) at an instant smaller than 3. Thus, line 1 does not fire in this \( \xi \)-interval. On the other hand, line 2 fires. Hence, on leaving this \( \xi \)-interval \( B_1 \) has \( x = 0 \), \( \alpha \) non executed, \( y = 1 \) and \( \beta \) executed with the initial values. In the next \( \xi \)-interval the guard of line 1 has value \((0 + 3 + \delta) \geq (1 + 1) \iff true\), and the guard of line 2 is \((1 + 3) \geq (0 + 5) \iff false\). Thus, line 1 is executed with \( x = 2 \) and \( y = 1 \) that does not correspond to the run of \( A_0 \). \( \square \)

# 3 Approximate Implementations

Now we make precise the type of implementations we study. In this section \( A_0 \) is a IA-ASM and \( A_1 \) is a DA-ASM (not necessarily the implementation of \( A_0 \) as specified in Definition 10). We introduce an approximate implementation of \( A_0 \) by \( A_1 \). Because of delays we cannot have exact correspondence between update instants of two machines, neither between the value of real-valued functions and \( CT \). So we may demand only approximate correspondence between runs of two machines, the approximation for the instants of updates will be denoted by \( \varepsilon \) and the approximation for the values — by \( \eta \). The term “correspondence” can be treated in various manners. We take as our basis an ‘approximate’ inclusion.

Denote by \( V \) the vocabulary of dynamic functions of \( A_0 \). We assume that \( V \) is also a part of the vocabulary of \( A_1 \). These are the observable functions that we are interested in when implementing \( A_0 \) by \( A_1 \).

Below, when we speak about an update or a guard of an ASM, we usually, though not always, mean an occurrence of this update or respectively of this guard in the program of this ASM. What we mean will be clear from the context.

**Definition 14** (Separating sequence).

- Two intervals are \( \delta \)-separated, \( \delta > 0 \), if the distance between the sets of points of these intervals is greater than \( \delta \) (recall that the distance between two sets of points of a metric space is the infimum of pairwise distances between their points).

- \( \eta^{(l)} \) and \( \eta^{(r)} \) are respectively the left and the right ends of an interval \( \eta \).

- A separating sequence of a run over a vocabulary \( W \) is a sequence of open intervals, each one containing only one time instant of updates of functions from \( W \). If \( \sigma \) is such a sequence then it may be finite or infinite, and its domain \( \text{dom}(\sigma) \) is a prefix of \( \mathbb{N} \). We assume that for any \( i, (i + 1) \in \text{dom}(\sigma) \) the right end \( \sigma(i)^{(r)} \) of interval \( \sigma(i) \) is strictly smaller then the left end \( \sigma(i + 1)^{(l)} \) of the next one: \( \sigma(i)^{(r)} < \sigma(i + 1)^{(l)} \), i.e., \( \sigma(i) \) is strictly monotone.

- A pair \( (\rho, \sigma) \), where \( \rho \) is a run and \( \sigma \) is a separating sequence of \( \rho \), is a run with separated updates.
• If the length of each interval of $\sigma$ is less than $\varepsilon > 0$ then we say that $\sigma$ is $\varepsilon$-separating sequence.

Remark that an interval of a separating sequence may have updates of several functions, but all of them take place at the same time instant.

Notations 3 (Instants of updates, updated functions in a separating sequence).
These notations are used only in the current section 3.

- $\Theta[\rho, \sigma](i)$, where $\sigma$ is a separating sequence of a run $\rho$ and $i \in \text{dom}(\sigma)$, is the set of functions whose values are updated in $\sigma(i)$ in a run $\rho$.
- $\tau[\rho, \sigma](i)$ is the instant where the functions from $\Theta[\rho, \sigma](i)$ are updated in $\sigma(i)$. Some arguments in this and previous notation can be omitted if they are clear from the context.
- For two pairs of runs with separated updates $(\rho_0, \sigma_0)$ and $(\rho_1, \sigma_1)$, where $\sigma_0$ and $\sigma_1$ are $\varepsilon$-separating sequences, just below we denote $\Theta[\rho_m, \sigma_m](i)$ and $\tau[\rho_m, \sigma_m](i)$ by respectively $\Theta_m(i)$ and $\tau_m(i)$, $m = 0, 1$.

Definition 15 (Admissible partition). A partition $\Pi_1$ of $\text{dom}(\sigma_1)$ is admissible for a separating sequence $\sigma_0$ if

- $\text{dom}(\Pi_1) = \text{dom}(\sigma_0)$,
- each $\Pi_1(j)$ is a non empty set of consecutive integers from $\text{dom}(\sigma_1)$,
- if $\Pi_1(j) = \{m, m+1, \ldots, m+l_j\}$, $l_j \geq 0$, the intervals $\sigma_1(i)$, $i \in \Pi_1(j)$ are inside $\sigma_0(j)$, but the ends are the same: $\sigma_1(m)^{[l]} = \sigma_0(j)^{[l]}$ and $\sigma_1(m+l_j)^{[r]} = \sigma_0(j)^{[r]}$.

We are in position to define closeness of runs.

Definition 16 (Closeness of runs).

- (Parameters of approximation) From now on we assume that $\varepsilon$, $\eta \in (0, 1)$.
- The closeness of values is characterized by the parameter $\eta$. For predicates $\eta$-closeness means that the values are equal, and for real-valued functions $\eta$-closeness means that absolute value of the difference of values is less than $\eta$.
- A run with separated updates $(\rho_1, \sigma_1)$ over a vocabulary $W$ is $(\varepsilon, \eta)$-close to a run with separated updates $(\rho_0, \sigma_0)$ over the same vocabulary if there exists an admissible partition $\Pi_1$ of $\text{dom}(\sigma_1)$ for $\sigma_0$, such that, using notation $\Pi_1(j) = \{m, m+1, \ldots, m+l_j\}$ ($l_j \geq 0$),

  - for each $j \in \text{dom}(\sigma_0)$ for each $k \in \Pi_1(j)$ there holds $\Theta_1(k) \subseteq \Theta_0(j)$;
  - for each $j \in \text{dom}(\sigma_0)$, the updated functions $\Theta_0(j)$ are distributed in intervals $\sigma_1(i)$, $i \in \Pi_1(j)$, i.e., $\Theta_0(j) = \bigcup_{k \in \Pi_1(j)} \Theta_1(k)$, and $\Theta_1(k) \cap \Theta_1(l) = \emptyset$ for $k \neq l$ (and thus, the instants $\tau_1(k)$ of updates of functions from $\Theta_1(k)$ are distributed in a similar way in respective intervals: $\tau_1(k) \in \sigma_1(k)$);
  - for any function $f \in \Theta_0(j)$ the value of $f[\rho_0][\tau_0(j)^+]$ is $\eta$-close to $f[\rho_1][\tau_1(j)^+]$ (in other words the values assigned by the corresponding updates are $\eta$-close).

Remark 8. Notice that for two $(\varepsilon, \eta)$-close runs (as above), the time instants of the respective updates are $\varepsilon$-close.
Definition 17 (Projection of a run). By $\text{proj}_U(\rho)$ we denote the projection of a run $\rho$ over a vocabulary $W$ onto its sub-vocabulary $U$. Projection, as usually, means that we delete from $\rho$ all the functions that are not in $U$.

Clearly, if $(\rho_1, \sigma_1)$ and $(\rho_0, \sigma_0)$ are $(\varepsilon, \eta)$-close over a vocabulary $W$ then the same takes place over vocabulary $U \subseteq W$ for $(\text{proj}_U(\rho_1), \sigma_1)$ and $(\text{proj}_U(\rho_0), \sigma_0)$.

![Figure 9: Definition of implementation](image)

Definition 18 (Approximate implementation). A DA-ASM $A_1$ with a vocabulary $V_1$ of dynamic functions is an $(\varepsilon, \eta)$-implementation of an IA-ASM $A_0$ with a vocabulary $V$ of dynamic functions if $V \subseteq V_1$ and for any input (i.e., for any timed interpretation of input functions) and initial values, for every run $\rho_1$ of $A_1$ with this input and these initial values, there exist a separating sequence $\sigma_1$ for $\text{proj}_V(\rho_1)$ and a separating sequence $\sigma_0$ for the run $\rho_0$ of $A_0$, such that $(\rho_0, \sigma_0)$ and $(\text{proj}_V(\rho_1), \sigma_1)$ are $(\varepsilon, \eta)$-close (see Figure 9).

Definition 19 (Approximate bisimulation). A DA-ASM $A_1$ is a bisimilar $(\varepsilon, \eta)$-implementation of an IA-ASM $A_0$ if

$\text{(DownUP)}$ it is an $(\varepsilon, \eta)$-implementation of an IA-ASM $A_0$

and

$\text{(UpDown)}$ for any input and initial values, for every run $\rho_0$ of $A_0$ with this input and these initial values, there exist a run $\rho_1$ of $A_1$ with these input and initial values and a separating sequence $\sigma_1$ for $\text{proj}_V(\rho_1)$, a separating sequence $\sigma_0$ for the run $\rho_0$ of algorithm $A_0$ such that $(\rho_0, \sigma_0)$ and $(\text{proj}_V(\rho_1), \sigma_1)$ are $(\varepsilon, \eta)$-close.

4 Sturdy Real Time Machines

As we noticed above in section 2.3, not any IA-ASM can be implemented by a DA-ASM. In this section, we will give a sufficient condition on the behavior of an IA-ASM under which its $\mathcal{I}$-transformation defined in Definition 10 is an $(\varepsilon, \eta)$-implementation. First we introduce parameters used in our sufficient condition. Then we formulate the sufficient condition as a notion of $(\varepsilon, \eta)$-sturdy IA-ASM. This notion concerns only IA-ASM, not DA-ASM. Then, in section 5, we prove that $\mathcal{I}$-transformation of any $(\varepsilon, \eta)$-sturdy machine gives an $(\varepsilon, \eta)$-bisimulation.
4.1 Definition of Sturdy Machines

Let $A_0$ be an IA-ASM, $A_1 = \mathcal{I}(A_0)$ and $\varepsilon, \eta \in (0, 1)$.

**Notations 4 (Notations for external literals of $A_0$).** For an IA-ASM $A_0$, for a formula, term, set of formulas or set of terms $\Phi$:
- $\text{Grds}$ is the set of all guards of $A_0$.
- $\text{RF}$ is the set of all real-valued functions different from $CT$ that occur in (the program of) $A_0$.
- $\text{Inp}\Phi$ is the set of input literals occurring in $\Phi$.
- $\text{Ineq}\Phi$ is the set of external inequalities (i.e., external inequality literals) occurring in $\Phi$.

The updates of $A_0$ are either of the form $\alpha := \beta$, where $\alpha, \beta$ are boolean-valued functions, or of the form $g := c_0 \cdot CT + c_1 \cdot f_1 + \cdots + c_q \cdot f_q$ (14) where $c_i (i = 0, \ldots, q)$, are rational numbers, and $f_i \in \text{RF}$, $(i = 1, \ldots, q)$.

A literal constituting a guard $G_i$ of $A_0$ is either a predicate or an inequality, maybe involving $CT$:

$$b \omega a_0 CT + a_1 h_1 + \cdots + a_r h_r$$

(15) where $\omega \in \{<, \leq, >, \geq\}$, $a_i, b$ $(i = 0, \ldots, r)$ are rational numbers, and $h_i \in \text{RF}$, $(i = 1, \ldots, r)$.

For expressions like those in the right-hand part of (14), (15) we use also vector form and $\mathbb{L}^{\infty}$-norm. For vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$:
- $|x| = \max\{|x_1|, \ldots, |x_n|\}$,
- $x - y = (x_1 - y_1, \ldots, x_n - y_n)$,
- $x \cdot y = xy = x_1 y_1 + \cdots + x_n y_n$.

Sometimes the time component is distinguished in vectors like $(c_0 CT, x)$ or $(c_0 t, x)$; it is always the first one.

**Assumption 5 (Coefficient of $CT$ in inequalities).** Without loss of generality, the coefficient $a_0$ of $CT$ in (15) can be taken either 0 or 1.

A vector $x$ can be a list of function symbols of the vocabulary of an ASM, or a list of real numbers, or a list of terms involving the both.

**Definition 20 ($\delta$-stable inequality).** For an arbitrary vocabulary, a vector $h = (h_1, \ldots, h_n)$ consisting of $CT$ and real-valued functions of this vocabulary, and for a state $S$ over this vocabulary:
- $h[S]$ denotes the vector $(h_1[S], \ldots, h_n[S])$.
- An inequality $b \omega c \cdot h$ (we refer to (15) for notations), where $c$ is a vector of rational numbers, is $\delta$-stable at $S$, where $\delta \geq 0$ is real number, if for any $\alpha$ with $|\alpha| < \delta$ there holds
  $$b \omega c \cdot h[S] \Leftrightarrow b \omega c \cdot (h[S] + \alpha).$$
  This notion is applied below only to inequalities without time, i.e. with $a_0 = 0$ in terms of (15).

The following parameters will be used to define constraints on the delay $\xi$ of the implementation, see (23) below.
Notations 5 (Parameters \( p, K, M \) of IA-ASM).

- \( p \geq 1 \) is an upper bound for the number of functions involved in the literals of the form (15) and in the updates of the form (14) (i.e., the maximum of all \( q \)'s and \( r \)'s plus 1).
- \( K \) is an upper bound for the maximum of absolute values of \( a_i, c_i \) taken over all expressions of the form (14) and (15).
- \( M \geq 1 \) is an upper bound for \(|RF|\), i.e., for the number of (internal dynamic) real-valued function of IA-ASM \( A_0 \). \( \square \)

In terms of these parameters from Notations 5 and of notations from Definition 20, if \( |\alpha| \leq \delta \) then

\[
|c \cdot h[S] - c \cdot (h[S] + \alpha)| < |c \cdot \alpha| \leq Kp\delta.
\]  (16)

Example 7 (Canonical form of controller \( C_2 \): controller \( C_3 \)). In Example 3 we have arrived at a new controller \( C_2 \) that we can try to implement, as the elegant initial controller \( C_0 \) and its canonical version \( C_1 \) are not implementable. First, we transform \( C_2 \) into a controller \( C_3 \) in canonical form. For better readability we group several updates of the same guard in one \textbf{if-then}-operator. The resulting IA-ASM \( C_3 \) is in Figure 10. This time \( emp \) and \( close \) are new functions of the vocabulary as in \( C_1 \) in Figure 4.

\[
\begin{align*}
0: & \text{ if } cmg(0) \wedge dl = 0 \text{ then } dl := CT + wt \\
1: & \text{ if } cmg(1) \wedge dl = 0 \text{ then } dl := CT + wt \\
2: & \text{ if } open \wedge CT \geq dl > 0 \text{ then } [close, \neg open] \\
3: & \text{ if } close \wedge emp(0) \wedge emp(1) \text{ then } [open, dl := 0, \neg close]
\end{align*}
\]

Figure 10: Railroad Crossing Controller \( C_3 \).

One can see that for \( C_3 \): \( p = 2 \) and the other parameters from Notations 5 have their minimal values.

Look at lines 0–1 of \( C_3 \) in Figure 10. They have different guards but the same update. It may happen that these two guards fire simultaneously. However, in an implementation they may fire in very close but different \( \xi \)-intervals. If one of them but not the other one fires in such a way, the other one becomes false. But this other one executes the same update, so no need that it fires itself. There may happen that line 1 fires slightly before line 0 in IA-ASM, but in the implementation their firings are inverted. But they execute updates that give very close results and hence, it is not dangerous.

This situation is not so rare for controllers that have one action to control but the control signal may arrive from different processes (in controller \( C_1 \) in Figure 4 there are more combinations of this kind). In order to be able to treat such situations we introduce appropriate notions in Definition 21. \( \square \)

Definition 21 (Locking of guards). A set of guards \( \mathcal{G} \) is \textit{locked by a state} \( S \) if all guards of \( \mathcal{G} \) are false at this state. An \textit{update locks} a set of guards \( \mathcal{G} \) if the state obtained after this update locks the set \( \mathcal{G} \). (From the viewpoint of a given run the update may be partial — see Notations 8.) \( \square \)
In an update in a run of $A_0$ a real-valued function may be updated by a computed real value involving current time $CT$. In the implementation $A_1$ it is updated with some value $\tilde{CT}$ of the current time caught (backed up) during the backup loop of $A_1$, see Figures 5, 9. This time shift may be multiplied in sums of the form (14) first by $c_0$, and later it may be used via updated $g$ as $f_i$ in another sum like (14) and so on. The result may strongly deviate from the respective values of real-valued functions in $A_0$ as compared with those in $A_1$.

In order to avoid such accumulation of value shifts (in fact, they are errors), we suppose that the real-valued functions are regularly reset to a default value in a ‘synchronized’ way.

**Assumption 6 (Default value).** We suppose that each internal real-valued function of an ASM $A$ (IA-ASM or DA-ASM) that we consider, has some fixed value that will be called its default value. For technical simplicity we assume that the initial value of any internal real-valued function is its default value. □

**Definition 22 (Default values resetting instant).** A time instant $t$ is a default values (resetting) instant for a set of real-valued functions $F \subseteq RF$ in a run $\rho$ if all the functions of $F$ have their default value immediately after an update at $t$ but not at $t$. We consider $-\infty$ as the first default values instant. The word “resetting” will be often omitted for brevity. □

So one can see that a default values resetting instant finalizes the resetting, and does not means that all the functions are reset at the same instant. The resetting to the default values is needed only for ‘dependent’ real-valued functions, defined below.

**Definition 23 (FD-partition).** For a given IA-ASM $A_0$ we define FD-partition (FD stands for Function Dependency) $D(A_0)$ of functions of $RF$ in the following way. For any if-then operator with updates of real-valued functions all real-valued functions that occur in this operator (both in guards and updates) are in the same set of $D(U)$. □

**Remark 9 (FD-partition and the form of guards).** This Definition 23 is reasonable only for ASMs whose guards are conjunctions of literals, see Example 8. □

**Example 8 (FD-partitions for Railroad Crossing Controllers).** For example, in the controller $C_1$ in Figure 4 all internal real-valued functions, namely $dl(0)$ and $dl(1)$, are in different sets of its FD-partition. It is not the case for $C_0$ in Figure 2, but its guards are not conjunctions of literals. The Resetting Assumption 7 formulated below also holds for $C_0$ and $C_1$. Controller $C_3$ in Figure 10 has only one internal real-valued function, and therefore its FD-partition is trivial. The controllers $C_1$ and $C_0$ have another property that can play a role similar to that of the Resetting Assumption, namely, the real-valued functions are updated by expressions depending only on $CT$, not on other real-valued functions. However, we do not explore this way. □

All this leads us to the constraint:

**Assumption 7 (Resetting Assumption).** There is a natural number $\nu$ with the following property: for any set $F \in D(A_0)$, after any time instant $t_0$ that is either the initial instant or a default values instant, there can be not more than $(\nu - 1)$ updates of the functions of $F$ before the next default values instant. Without loss of generality, we assume that $\nu \geq 2$. □
Remark that we take the same $\nu$ for all sets of a FD-partition without loss of generality. We also assume that

$$K \geq \mu + 1,$$

where $\mu = d_M \nu$ — this notation is largely used below. \hfill (17)

**Definition 24 (Last default values reset).** For a run $\rho$ of $A_0$, for a $f \in RF$ and for a time instant $t$, we define the last default values reset at $t$ as the maximal time instant not greater than $t$ where $f$ has been reset to its default value. \hfill $\square$

Here we make precise what update numbers figure in states.

**Definition 25 (State with the numbers of updates after reset).** A state of an ASM $A$ is an interpretation of its vocabulary, where each real-valued $f$ is supplied with a number between 0 and $(\nu - 1)$ that is called the update number of this function. For a state $\rho(t)$ defined by a run $\rho$ this number is always the number of updates of this function after the last default values reset for $f$ and $t$ in $\rho$. \hfill $\square$

**Notations 6 (Function $s$).** For a time instant $t$ and a state $E$:

- $s_f[E]$, where $f \in RF$, denotes the update number of $f$ in state $E$.
- $s_F[E]$, where $F$ is a term, a formula, a set of terms or a set of formulas, denotes the maximal update number of internal real-valued functions occurring in $F$, in state $E$. We set $s_{\emptyset}[E] = 0$.

Clearly, $s_F[E]$ is monotone with respect to sets $F$: if $F \subseteq \mathcal{H}$ then $s_F[E] \leq s_{\mathcal{H}}[E]$. (This monotonicity is not strict, i.e., $F \subseteq \mathcal{H}$ may not imply $s_F[E] < s_{\mathcal{H}}[E]$.)

- In the proof the Main Theorem below an IA-ASM $A_0$ will be fixed as well as its run $\rho_0$.

By default,

$s_F(t)$ denotes $s_F[\rho_0(t)]$, and $s_F(t^+)$ denotes $s_F[\rho_0(t^+)]$. \hfill $\square$

We need a detailed view on the usage of $\tilde{CT}[\rho_1]$ in $A_1$. This value is updated in each $\xi$-interval (Definition 11). If $\tilde{CT}$ is updated at $\tau$, it preserves its values until the next update. And the next update is in the next $\xi$-interval. So

$$\tau - 2\xi < \tilde{CT}[\rho_1](\tau) < \tau.$$ \hfill (18)

Now we introduce notations that will be used to express a resistance of $A_0$ to small perturbations of its run caused by its implementation.

**Definition 26 (Partial run).** A run $\rho$ is partial if it is defined over an interval $(-\infty, T]$ and at $T^+$ for some $T < \infty$. We say that it is defined until $T$. Notice that inputs of a partial run are defined everywhere in $\mathbb{R}$. \hfill $\square$

The next Definition 27 introduces maximal time intervals where literals and guards remain unchanged in a run. These intervals will play a central role in reasoning about instants where the same guards fire in related runs $\rho_0$ and $\rho_1$ of respectively $A_0$ and $A_1$.

**Definition 27 ($\rho$-intervals).** For a literal $\Phi$, a partial run $\rho$ of $A_0$ or $A_1$ defined until a time instant $t$, an instant $\tau \leq t$ and an interpretation $S$ of a sub-vocabulary of $\rho$ containing all internal functions of $A_0$ and their tilde-images:

- For an inequality $\Phi$ containing $CT$ or $\tilde{CT}$ denote by $\Phi[\tau, S, \rho]$ the proposition obtained from $\Phi$ by the following transformation:
– each internal function of $A_0$ and its tilde-image occurring in $\Phi$ are replaced by their values from $S$,
– $CT$ is replaced by $\tau$, and $\overline{CT}$, if it occurs, is replaced by $\overline{CT}[\rho(\tau)]$.

• For an inequality $\Phi$ containing $CT$ or $\overline{CT}$, its $(S, \rho)$-interval is any maximal time interval where the Boolean-valued function $\lambda \tau \Phi[\tau, S, \rho]$ does not change its truth value. Such an interval is a true (respectively, false) if the value of the mentioned function is true (respectively, false) in this interval.

By $(\rho, t)$-interval for such inequality $\Phi$ we mean its $(\rho(t), \rho)$-interval.

If $\Phi$ does not contain $\overline{CT}$ then $\rho$ is not involved into its definition, and we can refer to its $(S, \rho)$-interval as $S$-interval.

• For an inequality $\Phi$ containing $CT$ or $\overline{CT}$, its $\rho$-interval is any maximal time interval where the function $\lambda \tau \Phi[\rho(\tau)]$ does not change its value. Such an interval is a true (respectively, false) if the value of the mentioned function in it is true (respectively, false). A $\rho$-interval of $\Phi$ that contains $t$, is called $(\rho, t)$-interval.

• For a guard $G$ its $(S, \rho, t)$-interval (respectively, $(\rho, t)$-interval) is an intersection of $(S, \rho)$-intervals (respectively, $(\rho, t)$-intervals) of its external inequalities, and of $(\rho, t)$-intervals of inputs and of its tilde-images, and of $S$-intervals (respectively, $\rho(t)$-intervals) of the internal functions of $A_0$ and their tilde-images (recall that our guards are conjunctions of literals).

• $(\rho, t)$-interval or $(S, \rho, t)$-interval is also referred to as $\rho$-interval when other of its parameters are irrelevant to or clear from the context. In order to homogenize the notation, $\rho$-interval of an input that contains $t$ is also called $(\rho, t)$-interval.

• A $\rho$-interval is left-unbounded if its left end is equal to the minimum time, i.e., $-\infty$, and right-unbounded if its right end is $\infty$. A $\rho$-interval is left(right)-bounded if it is not respectively left(right)-unbounded. It is bounded if it is left- and right-bounded.

• Below by default, a $\rho$- or $(\rho, t)$-interval is a true $\rho$- or $(\rho, t)$-interval.

\begin{remark}
When $\tau$ changes, the values of inputs and their tilde-images respectively change in $\Phi[\tau, S, \rho]$ but the the values of internal functions and their tilde-images remain unchanged.

A $(\rho, t)$-interval of inequality may not contain $t$ as compared to non-empty $(\rho, t)$-intervals of predicates.

It is clear from Definition 27 that a $(\rho, t)$-interval of an external inequality is an interval whose one end equals either $-\infty$ or to $\infty$.
\end{remark}

\begin{remark}
Let $G$ be a guard of $A_0$. The corresponding tilde-guard $\tilde{G}$ of $A_1$ does not contain functions of $G$. The functions $\tilde{G}$ are tilde-versions of functions of $G$. The values of such tilde-functions update the values of corresponding functions of $A_0$ within some delay. Thus, generally speaking, the value of $\tilde{G}[\tau, \rho_1(t), \rho_1]$ is not the same as the value of $G[\tau, \rho_1(t), \rho_1]$.
\end{remark}

Each function of an ASM $A$ (of any type) may be considered as variable over an appropriate space: $CT$ or another real-valued function — over $\mathbb{R}$ and a Boolean-valued function — over the space of $\mathbb{B}$ of two Boolean values (it is convenient, though does not matter, to take $\mathbb{B} = \{0, 1\}$, and treat it as a sub-apace of $\mathbb{R}$). Thus, a state of $A$ is a point in a direct product $S$ of an appropriate number of $\mathbb{R}$ and $\mathbb{B}$.
Notations 7 (Space of states and its subspaces). For an ASM \( A \) of any type we define the following spaces.

- Denote by \( K_r \) the number of real-valued functions of the vocabulary of \( A \), by \( K_{ext} \) the number of input predicates and by \( K_{int} \) the number of internal predicates. The space of states is
  \[
  S = T \times S_{ext} \times S_r \times S_{int},
  \]
  where
  - \( S_{ext} = \mathbb{R}^{K_{ext}} \) is the space of values of input predicates,
  - \( S_r = \mathbb{R}^{K_r} \) is the space of values of real-valued functions,
  - \( S_{int} = \mathbb{R}^{K_{int}} \) is the space of values of internal predicates.

- For \( S \in S \) we denote by \( S|_{S_0} \) projection of \( S \) onto a subspace \( S_0 \) of \( S \).

- A state denoted as \((\tau, S_1, S_2, S_3)\) (if possible, it may be denoted \((\tau, S_{ext}, S_r, S_{int})\)) means that \( \tau \in T, S_1 \in S_{ext}, S_2 \in S_r, S_3 \in S_{int} \).

The parallelism intrinsic to our IA-ASM impose particular demands on the sturdiness of algorithms that are implementable. In particular, the updates executed simultaneously should not essentially influence each other because in the implementation they will not normally be executed simultaneously, though close to each other. The notations below permit to describe appropriate properties.

Notations 8 below describe neighborhoods of a given partially updated state.

Notations 8 (Partial updates and neighborhoods). For a set \( G \) (that may be empty) of all guards of IA-ASM \( A_0 \) that fire at an instant \( t \) in a run \( \rho \) of \( A_0 \), and for its subset \( G_0 \subseteq G \) (\( G_0 \) may also be empty or may coincide with \( G \)) we use the following notations (recall that we use \( \ell^\infty \)-norm, see section 4.1):

- \((S \downarrow G_0)\), where \( S \in S \), is the state obtained from \( S \) by replacing the values of its functions by the values of the respective updated functions after firing the guards from \( G_0 \) with values defined by \( S \). Clearly, \((S \downarrow \emptyset) = S \).

- The update numbers of \((S \downarrow G_0)\) are defined as follows: a real-valued function \( f \) that is not updated by guards from \( G_0 \) preserves its update number \( s_f[S] \), a real-valued function \( f \) that is updated by guards from \( G_0 \) changes its update number to \((s_f[S] + 1 \mod \nu)\).

- \((\rho \downarrow t G_0)tG_0 = \rho(t) \downarrow G_0\) (remark that if \( G_0 = \emptyset \) then \((\rho \downarrow t G_0) = \rho(t)\), if \( G_0 = G \) then \((\rho \downarrow t G_0) = \rho(t^+)\)).

- For a set of time instants \( X \) and a set of values of real-valued functions \( Y \) (See Figure 11 and Remark 12 below)
  \[
  \mathfrak{M}(\downarrow G_0, \rho, t, X, Y) = \{ \tau \in X \land S_{ext} = \rho(\tau)|_{S_{ext}} \land S_r \in Y \land S_{int} = (\rho \downarrow t G_0)|_{S_{int}} \}.
  \]

- For a \( S^0_r \in S_r \) and a non-negative real \( \delta \)
  \[
  \mathfrak{R}(S^0_r, \delta) = \{ S_r \in S_r : |S^0_r - S_r| \leq \delta \}.
  \]
  This is a cube centered at \( S^0_r \) with edges of length \( 2\delta \).
• \((\gamma, \delta)\)-neighborhood of a state \(\rho_0(t)\) at \(t\) for \(\gamma, \delta \geq 0\) for a set \(G_0 \subseteq G\) is the set
\[ \mathcal{N}(\lll G_0, \rho, t, \gamma, \delta) = \mathcal{D}(\lll G_0, \rho, t, [t - \gamma, t + \gamma], \mathcal{R}(\lll \rho \lll G_0)) \, . \]

• When we speak about a state \(S\) from a neighborhood of a state \(S_0\) defined by a run of an IA-ASM, we presume that the *update numbers in \(S\) are inherited* from \(S_0\). □

![Diagram](image-url)

**Figure 11:** A neighborhood of a state.

**Remark 12 (Figure 11 comment).** Figure 11 illustrates a neighborhood for a state with two real-valued functions \(f\) and \(g\), and one input. The set \(X\) of time values is an interval \([a, b]\), and the set \(Y\) is a cube with edges of length \(2\delta\). The instant \(t\) is an instant whereat the input changes its value. □

The Assumption 8 is crucial for the sturdiness. The properties imposed on IA-ASM by this assumption are semantical. Property (AR1) says that small perturbations of the state at which several guards fire do not change their interaction. More precisely, if we execute a subset \(G_0\) of a set of guards that fire at a given instant, then the remaining guards that are not locked after that, are still true at perturbed states.

Property (AR2)(a1) says that the updates of the locked guards are done by not locked ones. Property (AR2)(a2) says that the locked guards cannot be unlocked even in perturbed states.

Property (AR4) says that a guard false at \(t\) is robustly false. This robustness concerns its \(\rho_0\)-intervals, namely, all pairs of disjoint \(\rho_0\)-intervals (if there are any) cannot be too close, at least one such pair should have well separated intervals.

Property (AR5) says that there is no ‘bad’ interaction of shared variables in the program.

In Assumption 8(AR4) below we use the *set of \(\rho_0\)-intervals* of an external literal of a guard \(G\) that are closest to an instant \(t\) from the left or from the right. Though “closest to an instant” is an exact notion that gives a rigorous definition, we make here some comments. For an input literal \(P\) its \(\rho_0\)-intervals are given by the interpretation of the inputs and do not depend on \(G\). There can be only one \(\rho_0\)-interval of \(P\) that contains \(t\), and in this case it is the closest one, or
Remark 13. In the item (AR1) of Assumption 8 below we speak about sets of guards and not about sets of occurrences of guards. Moreover, to make this assumption reasonable we tacitly assume that equivalent guards and equivalent terms are equal. The latter (as well as the former) is not important for the Main Theorem but important to make it applicable to concrete ASMs.

Assumption 8 (Separation and Independence Assumption). For any run $\rho_0$ of $A_0$, any time instant $t$, any sets of guards $G$ and $C$ such that

- $G \neq \emptyset$ is the set of all guards of $A_0$ that fire at instant $t$ in $\rho_0$.
- $C \subseteq G$ (the set $C$ may be empty or equal to $G$).
- $C^\perp$ is the set of guards of $(G \setminus C)$ that are locked by state $(\rho_0 \downarrow_t C)$.

there holds:

(AR1) For any $\delta$ such that $0 \leq \delta \leq \eta$, for any $G \in (G \setminus C)$, that is not locked by $(\rho_0 \downarrow_t C)$, i.e., for any $G \in (G \setminus (C \cup C^\perp))$, there exists $t' \in [t - Kp\delta, t + Kp\delta]$ such that $G$ is true in all states from

$$\mathcal{M}(\downarrow_t C, \rho_0, t, [t', t + \varepsilon], \mathcal{R}(\rho_0 \downarrow_t C|_R, \delta)).$$

(This means that a guard that is true after a partial update, remains true starting from an instant close to $t$, up to $(t + \varepsilon)$ even with perturbed values of real-valued functions; the closeness to $t$ is sufficiently small and is related to the perturbation $\delta$ of real-valued functions.)

(AR2) (a1) (Compensation of locking.) $(\rho_0 \downarrow_t (C \cup C^\perp)) = (\rho_0 \downarrow_t C)$. (This means that locking is not dangerous for updates: the locked updates are done by the guards that lock them. Clearly, these locking guards do the same updates as the updates of $\rho_0$ at $t_0$ are consistent.)

(a2) (Monotonicity of locking.) Once locked, a guard cannot be unlocked later in the neighborhood under consideration: for any guard $G \in C^\perp$, for any set of guards $F$ such that $C \subseteq F \subseteq G$, if $G \in (G \setminus F)$ then $G$ is locked by all states from $\mathcal{M}(\downarrow_t F, \rho_0, t, \varepsilon, \eta)$.

(AR3) All guards from $G' =_{df} (Grds \setminus (G \setminus C))$ are false everywhere in $\mathcal{M}(\downarrow C, \rho_0, t, \varepsilon, 2\eta)$. (In other words, no guard that is not in $(G \setminus C)$, can fire anywhere in $[t - 2\varepsilon, t + 2\varepsilon]$ with $\eta$-perturbed, partially updated values of its functions).

(AR4) Let a guard $G$ be false at an instant $T$ in $\rho_0$ with the values of internal functions from the state $(\rho_0 \downarrow_t C)$.

Define a set $E(G, T, (\rho_0 \downarrow_t C))$, that is also denoted by $E(G, T, C)$ if $\rho_0$ and $T$ are clear from the context, as the set of the following (true) $\rho_0$-intervals of external literals of $G$:
Lemma 1. Let $\delta$ be defined in Notations 4). Therefore according to (AR3), any guard of any guard of instances is not less than (i.e., there are no two updates of the form $f := g$ and $h := \theta(f)$, where $\theta(f)$ is a term containing $f$, in this set of updates fired by these guards).

Remark 14. In fact the property (AR1) of Assumption 8 will be used only for sufficiently small $\delta$, roughly such that $Kp\delta < \varepsilon$. It is hard to ensure this property if $Kp$ is big and $\eta$ comparable with $\varepsilon$. If we omit everything that concerns locking, then the Main Theorem remains true, but many machines with non-trivial parallelism will be out of scope of the theorem. Even the Railroad Crossing controllers that we treat in the examples.

Definition 28 (IA-ASM with stable internal inequalities). An IA-ASM $A_0$ has $\delta$-stable internal inequalities (inequality literals) if for any run $\rho_0$ of $A_0$, any its internal inequality is $\delta$-stable at any state $\rho_0(t)$ (recall Definition 20).

Definition 29 (Sturdy IA-ASM). A canonical (see Definition 8) IA-ASM $A_0$ is $(\varepsilon, \eta)$-sturdy if it has $\eta$-stable internal inequalities and satisfies Assumptions 7, 8.

Remark 15. Within the context of Assumption 8

$$G^\perp = \emptyset,$$  \hspace{1cm} (20)

$$\emptyset^\perp = \emptyset,$$  \hspace{1cm} (21)

$$(C \cup C)^\perp \subseteq C^\perp.$$  \hspace{1cm} (22)

Indeed, (20) is trivial as, by definition, $G^\perp$ is a subset of $G \setminus G = \emptyset$.

Consider (21). The set $\emptyset^\perp$ is the set of guards of $G$ that are locked by state $\rho_0(t)$. But all the guards of $G$ fire at $t$ in $\rho_0(t)$, so no one is locked by this state.

Consider (22). Let $G \in (C \cup C^\perp)^\perp$. Then $G \subseteq (G \setminus (C \cup C^\perp)) \subseteq (G \setminus C)$, and $G$ is locked by $\rho_0(\downarrow t)(C \cup C^\perp))$. According to Assumption 8(a2), $(\rho_0(\downarrow t)(C \cup C^\perp)) = (\rho_0(\downarrow t)C)$. Hence, $G$ is locked by $\rho_0(\downarrow t)$. (Remark that $E \setminus G$, $T \setminus E$ can be false at states from $\rho_0(\downarrow t)$.)

Lemma 1. For any run $\rho_0$ of a sturdy IA-ASM $A_0$

(i) the update instants of $\rho_0$ are $2\varepsilon$-separated, i.e., the distance between 2 consecutive update instances is not less than $2\varepsilon$,

(ii) any false $\rho_0$-interval of any guard of $A_0$ is at least $2\varepsilon$-long, and any true $\rho_0$-interval of any guard of $A_0$ is at least $\varepsilon$-long.

Proof. Let $t$ be a time instance of update (the case when the run contains no updates is trivial). Let $F$ be all guards that fire at $t$.

Apply Assumption 8, (AR3) to $t$ with $G = F$ and $C = F$. In this case $G^* = Grds$ ($Grds$ is defined in Notations 4). Therefore according to (AR3), any guard of $A_0$ is false at states from...
\( \mathcal{R}(\emptyset, \rho_0, t, 2\varepsilon, \eta) \), in particular, at states from \( \mathcal{R}(\emptyset, G, \rho_0, t, 2\varepsilon, 0) \). This implies that any guard with values updated at \( t \) is false after \( t \) until at least the instant \( (t + 2\varepsilon) \).

This proves (i).

It follows from (i) that false \( \rho_0 \)-intervals of guards are at least \( 2\varepsilon \)-long.

To evaluate true \( \rho_0 \)-intervals it suffices to consider \( \rho_0 \)-intervals of guards from \( \mathcal{F} \), as the left end of a true \( \rho_0 \)-interval of a guard is an instant where the guard fires. Apply Assumption 8, (AR1) to \( t \) with \( G = \mathcal{F} \) and \( C = \emptyset \) and \( \delta = 0 \). Formula (21) implies that no guards from \( \mathcal{F} \) are locked by \( \rho_0(t) \). According to (AR1), for any \( G \in \mathcal{F} \) there exists \( t' \in [t, t] \) (i.e., \( t' = t \)) such that \( G \) is true in all states from

\[
\mathfrak{U}(\emptyset, \rho_0, t, [t', t + \varepsilon], \mathcal{R}((\rho_0 \downarrow 0)_{|_{S_t}} , 0)) = \mathfrak{U}(\emptyset, \rho_0, t, [t, t + \varepsilon], \rho_0(t)_{|_{S_t}} , 0),
\]

i.e., \( G \) is true with the values of internal functions at \( t \) and with the values of inputs from all instants from \([t, t + \varepsilon]\) when time ranges over \([t, t + \varepsilon]\). Hence, \([t, t + \varepsilon]\) is a part of \((\rho_0, t)\)-interval of \( G \). This proves (ii) of Lemma 1.

**Example 9 (Towards a sturdy Railroad Crossing controller).** We try to analyze \( \mathfrak{C}_3 \) in Figure 10 from the point of view of sturdiness.

Resetting Assumption 7 holds for all controllers \( \mathfrak{C}_j \) that we considered, with \( \nu = 2 \) (we cannot take less if we wish to apply Main Theorem).

Suppose that \( \varepsilon \) and \( \eta \) are sufficiently small, and that they are sufficiently close with respect to the involved constants \( d_{\text{min}}, d_{\text{close}} \) and \( d_{\text{sep}} \). We try to reason qualitatively, in terms of “sufficiently” small in order to be more comprehensive. Given these parameters, we look at Separation and Independence Assumption 8. Let \( \rho_0 \) be a run of \( \mathfrak{C}_3 \).

**Property (AR5).** This property is evident, see the syntax of \( \mathfrak{C}_3 \).

**Property (AR1) does not hold for \( \mathfrak{C}_3 \).** Denote the guard in line \( j \) in \( \mathfrak{C}_3 \) by \( G_j, 0 \leq j \leq 3 \).

The following sets of guards may fire at a same instant: \( \{G_0, G_1\}, \{G_2\}, \{G_3\} \).

Look at (AR1) for \( G = \{G_0\} \) and \( C = \emptyset \). Suppose that in a run \( \rho_0 \) this guard \( G_0 \) is true at an instant \( t \). So it is not locked by the state at \( t \). The property (AR1) demands that even in a \( \delta \)-perturbed state this guard is true (with non updated functions) at some instant that is \( \varepsilon \)-close to \( t \). But if we perturb \( dl \) the condition \( dl = 0 \) becomes false. Thus, (AR1) is not verified for \( \mathfrak{C}_3 \).

There are two ways to fix the problem. The first one is to remark that 0 is the default value of \( dl \), and that the default values are identical in IA-ASM and its implementation. Thus, we can modify the definition of perturbation by not perturbing the functions that have their default value. However, this way complicates the definition of sturdiness that is already not so simple. Besides, this way is not enough robust, as well as the condition \( dl = 0 \) (one should avoid equalities of real values in guards in programs as these equalities may not hold for computer reals).

The second way is to replace the literal \( dl = 0 \) by a more sturdy one. We know that the first positive value that \( dl \) can acquire is not less than \( (d_{\text{sep}} + wt) \). Thus, we can replace the literal \( dl = 0 \) by, say, \( dl < d_{\text{sep}} \). Now taking into account that the perturbation admitted in (AR1) are much smaller than \( d_{\text{sep}} \) we ensure the property in the situation just considered above. We follow this way and modify the controller \( \mathfrak{C}_3 \). Recall that the initial value of \( dl \) remains 0.

The modified controller is \( \mathfrak{C}_4 \) in Figure 12.
0: \(\text{if } \text{cmg}(0) \land dl < d_{\text{sep}} \text{ then } dl := CT + wt\)
1: \(\text{if } \text{cmg}(1) \land dl < d_{\text{sep}} \text{ then } dl := CT + wt\)
2: \(\text{if } \text{open} \land CT \geq dl \geq d_{\text{sep}} \text{ then } \{ \text{close, } \neg \text{open} \}\)
3: \(\text{if } \text{close} \land \text{emp}(0) \land \text{emp}(1) \text{ then } \{ \text{open, } dl := 0, \neg \text{close} \}\)

Figure 12: Railroad Crossing Controller \(\mathcal{C}_4\).

However, \(\mathcal{C}_4\) is not yet sturdy, the property (AR3) is not verified when the guard of line 3 fires at some instant \(t\). After the update fired by this guard, there holds \(dl = 0\), and hence, at least one of the guards of line 0 or line 1 is true to the left of \(t\). In fact, the property is too strong. However, making it more adequate complicates the framework. So we again sacrifice liveness:

(More Weak Liveness) If the gate is closed then open the gate only after waiting for time \(d_{\text{adj}}\), \(0 < d_{\text{adj}} < d_{\text{sep}}\) after all the trains leave the zone of control.  
(One may think that \(d_{\text{adj}}\) (adj comes from “adjourn”) is much smaller that \(d_{\text{sep}}\), but is, however, much greater than the parameters \(\varepsilon, \eta\) of approximation.)

To treat this additional waiting we introduce a function \(da\) (deadline for adjourning the opening): \(da :\rightarrow \mathbb{T}\).

Our new controller \(\mathcal{C}_5\) is in Figure 13.

0: \(\text{if } \text{cmg}(0) \land dl < d_{\text{sep}} \text{ then } dl := CT + wt\)
1: \(\text{if } \text{cmg}(1) \land dl < d_{\text{sep}} \text{ then } dl := CT + wt\)
2: \(\text{if } \text{open} \land CT \geq dl \geq d_{\text{sep}} \text{ then } \{ \text{close, } \neg \text{open} \}\)
3: \(\text{if } \text{close} \land \text{emp}(0) \land \text{emp}(1) \land da < d_{\text{adj}} \text{ then } da := CT + d_{\text{adj}}\)
4: \(\text{if } \text{close} \land \text{emp}(0) \land \text{emp}(1) \land CT \geq da \geq d_{\text{adj}} \text{ then } \{ \text{open, } \neg \text{close, } dl := 0, \text{da} := 0 \}\)

Figure 13: Railroad Crossing Controller \(\mathcal{C}_5\).

And again it is not sturdy, but this time because of the environment, see Figure 14. In this figure the conjunction \((\text{emp}(0) \land \text{emp}(1))\) is false everywhere in this figure, in particular, in a small interval \(\alpha\) bounded by two vertical dashed lines whose length is smaller than \(\xi\). On the other hand, it may happen in the implementation \(I(\mathcal{C}_5)\) that \(\text{emp}(0)\) and \(\text{emp}(1)\) are backed up in a \(\xi\)-interval \(\beta\), the first to the left of \(\alpha\), and the second to the right of \(\alpha\). Consequently, \(G_3\) fires at some \(T\) in the implementation.

To fix the problem we impose one more constraint.

Overall separation of train departures and arrivals. The instants of departure and of arrival of trains on different tracks are separated by a positive constant \(d_{\text{cmg}}\).

The index \(\text{cmg}\) is chosen because within the track uniform separation of trains, this constraint is equivalent to the following one: for any two different tracks \(x\) and \(y\) and for any run \(\rho\), if a \(\rho\)-interval \(\sigma_x\) of \(\text{cmg}(x)\) intersects or touch a \(\rho\)-interval \(\sigma_y\) of \(\text{cmg}(y)\), i.e., if \(\sigma_x^{(l)} \leq \sigma_y^{(r)}\), then \(|\sigma_x \cap \sigma_y| \geq d_{\text{cmg}}\).

Now we come back to the sturdiness of \(\mathcal{C}_5\). We suppose that an arbitrary run \(\rho_0\) and a time instant \(t\) are given. We use the same notation \(G_j\) for the \(j\)th guard, but this time, of \(\mathcal{C}_5\).

The Resetting Assumption is clearly valid for \(\mathcal{C}_5\). It is inherited from \(\mathcal{C}_3\). The property
We will use below the following observation that is implied by our constraint on the first instant of incoming of a train after time 0: for any instant $\tau$, if $\text{dl}$ becomes positive at $\tau$ then $\text{dl}[\rho_0](\tau) \geq (d_{\text{sep}} + w t) \gg \varepsilon, \eta$, where $\gg$ means “much greater”, and if $\text{da}$ becomes positive at $\tau$ then $\text{da}[\rho_0](\tau) \geq (d_{\text{sep}} + d_{\text{adj}}) \gg \varepsilon, \eta$.

Controller $C_5$ has two internal inequalities $\text{dl} \geq d_{\text{sep}}$ and $\text{da} \geq d_{\adj}$. The value of any of $\text{dl}$ and $\text{da}$ in any run of $C_5$ is either sufficiently big, as we have indicated just above, or is 0. In both cases the internal inequalities preserve their truth values for $\eta$-perturbed $\text{dl}$ and $\text{da}$. Thus, $C_5$ has $\eta$-stable internal inequalities.

**Properties (AR1)–(AR3) for $C_5$.** Consider the possible values of $G$ and $C$ that are used in the formulation of (AR1).

**Notations 9.** In this example we use the following notations:
- $\mathcal{W}((C) = \mathcal{W}(\downarrow_t C, \rho_0, t, \varepsilon, \eta)$,
- $\mathcal{S}(C) = (\rho_0 \downarrow_t C)$.
- For both $\mathcal{W}$ and $\mathcal{S}$ we will write its argument also as a list of elements of the respective set.

Consider a state and its neighborhood for the sets of guards that can fire as a given instant $t$ in $\rho$. For (AR4) we have an instant $T$ and a guard $G$ that is false at $T$ with the values of internal functions from $\mathcal{S}(C)$, where $C$ is a subset of guards that fire at $t$ in $\rho$. According to the convention in the formulation of (AR4), $\mathcal{E}(G, T, C)$ denotes $\mathcal{E}(G, T, \mathcal{S}(C))$.

**Case 1:** $G_0$ or $G_1$ (or both) fire at $t$ (and thus, become true). Then in state $\mathcal{S}(\emptyset) = \rho_0(t)$: $\text{dl} = 0$, open, and respectively $\text{cmg}(0)$ or $\text{cmg}(1)$ (or both). As $\text{dl}$ is updated to 0 after adjoining $d_{\adj}$ time after all the trains leave, then for $d_{\text{sep}}$ time before $t$ all $\text{cmg}(x)$ are false, and hence, any of these guards is false in $[t - d_{\text{sep}}, t]$, whatever be a perturbation of $\text{dl}$ (recall that $d_{\text{sep}}$ is a ‘global’ separation time). On the other hand, to the right of $t$ within up to $\eta$-perturbation of $\text{dl}$, the guards $G_i$, $i = 0, 1$, that is true at $t$, remains true in $[t, t + d_{\min}]$, as $\eta \ll d_{\text{sep}}$.

The guards $G_0$ or $G_1$ are mutually locking. So if the guards of any non empty subset $C$ of $G = \{G_0, G_1\}$ fire at $t$, then $C \cup C^\perp = G$. Hence, in this case (AR1) is trivially verified as there are no guards to consider. The only case to analyze is the case $C = \emptyset$. In this case $C^\perp = \emptyset$, and the argument just above proves (AR1) for $t' = t$.

Property (AR2)(a1) directly follows from the observation that the guards under consideration
are mutually locking, and their updates are identical.

Property (AR2)(a2) is non-trivial when \( C^\perp \neq \emptyset \). This happens if only one of \( G_0 \) and \( G_1 \) fires. Suppose that \( G_0 \) fires at \( t \), the case of \( G_1 \) is similar.

So property (AR2)(a2) is applicable only to \( G_1 \in \{G_0\}^\perp \) and \( \mathcal{F} = \{G_0\} \), as \( \mathcal{G} \setminus \mathcal{F} = \{G_1\} \) so there are no bigger \( \mathcal{F} \). Consider the state \( S(G_0) \) and its neighborhood \( \mathfrak{M}(G_0) \). The guard \( G_1 \) remains false after \( t \) for \( d_{\min} \) time with any small perturbation of \( d \ell \), as \( \eta < \eta \) and the first non-zero value of \( d \ell \) is \( \geq (d_{\sep} + wt) \). The guard \( G_1 \) is false to the left of \( t \) for \( d_{\sep} \) time whatever be the value of \( d \ell \).

Property (AR3). In fact there are 3 subcases to consider: first, the truth value of \( G_1 \) in \( \mathfrak{M}(G_{1-i}) \) \((i = 0, 1)\), second, the truth values of \( G_2, G_3, G_4 \) in \( \mathfrak{M}(\emptyset) \), and third, the truth values of \( G_2, G_3, G_4 \) in \( \mathfrak{M}(G_0), \mathfrak{M}(G_1) \) and \( \mathfrak{M}(G_0, G_1) \). The latter subcase is covered by the subcase of \( \mathfrak{M}(G_0) \), as \( G_0 \) fires the same update as \( G_1 \), or \( G_0 \) and \( G_1 \) together.

The first subcase, namely, the truth value of \( G_1 \) in \( \mathfrak{M}(G_0) \) has been just treated in the analysis of Property (AR2)(a2) just above.

Consider the truth values of \( G_2, G_3, G_4 \) in \( \mathfrak{M}(\emptyset) \), i.e., in the neighborhood of \( \rho_0(t) \) (see above a description of this state). As \( d \ell = 0 \), then \( d \ell \geq d_{\sep} \) remains false even with \( \eta \)-perturbed \( d \ell \). Hence, \( G_2 \) is false in \( \mathfrak{M}(\emptyset) \) because of its \( d \ell \). As for \( G_3 \) and \( G_4 \), they are false in \( \mathfrak{M}(\emptyset) \) because of \( \text{close} \), that is false at \( t \) (the approximation parameters play no role in this situation, as the values of internal functions are taken only at \( t \)).

Consider the truth values of \( G_2, G_3, G_4 \) in \( \mathfrak{M}(G_0) \), i.e., in the neighborhood of \( S(G_0) \). We have \( cmg(x) \) and \( d \ell = t + \eta \) that both remain true for \( d_{\min} \) time after \( t \), and \( \text{open} \) that remains true for \( \eta \) time after \( t \). Guard \( G_2 \) is false to the left of \( t \) because \( CT \leq t < d \ell \), and \( G_2 \) is false to the right of \( t \) for \( t + \eta \) time. Guards \( G_3 \) and \( G_4 \) are false in \( \mathfrak{M}(G_0) \) for the same reason for which they are false in \( \mathfrak{M}(\emptyset) \) in the previous subcase.

Property (AR3) is proven in Case 1.

Case 2: \( G_2 \) fires at \( t \). Then in state \( S(\emptyset) = \rho_0(t) \): \( \text{open} \), \( cmg(x) \) is true in \( [t - \eta, t + d_{\text{close}}) \) for some \( x \), and \( \text{CT}[\rho_0(t)] = t = d \ell \geq d_{\sep} \). In this case \( C^\perp \) is always empty, and hence, (AR2) is trivially verified.

Property (AR1) is non-trivial only for \( C = \emptyset \). In interval \([t, t + d_{\text{close}}]\) function \( d \ell \) remains unchanged and \( \text{CT} \) increases. If \( d \ell \) is \( \delta \) perturbed then \( \text{CT} \geq d \ell \geq d_{\sep} \) is true for \( \text{CT} \in [t + \delta, t + d_{\text{close}}] \), as \( t + \delta \geq d \ell + \delta > d \ell - \delta > d_{\sep} + \eta \). Hence, we can take \( t' = t + \delta \).

Property (AR3). We have 2 subcases to analyze: \( G^* = \{G_0, G_1, G_3, G_4\} \) (if \( C = \emptyset \)) and \( G^* = \{G_0, G_1, G_2, G_3, G_4\} \) (if \( C = \{G_2\} \)).

Case 2.1: \( C = \emptyset \). The state \( S(C) \) for this case is described above. Guards \( G_0, G_1 \) are false in \( \mathfrak{M}(\emptyset) \) because of \( d \ell \). Guards \( G_3, G_4 \) are false in \( \mathfrak{M}(\emptyset) \) because of \( cmg(x) \) that is true in \( [t - \eta, t + d_{\text{close}}) \) for some \( x \), or as well, because of \( \text{close} \) that is false at \( t \).

Case 2.2: \( C = \{G_2\} \). The state \( S(C) \) for this case: \( \text{close}, d \ell = t \) that remains unchanged in \([t - \eta, t + d_{\text{close}} + d_{\adj}] \), \( cmg(x) \) that is true in \([t - \eta, t + d_{\text{close}} + d_{\adj}] \) for some \( x \). In this situation \( G_0, G_1 \) are false in \( \mathfrak{M}(G_2) \) because of \( d \ell \). Guard \( G_2 \) is false because of \( \text{open} \) that is false now, and guards \( G_3, G_4 \) are false because of \( cmg(x) \).

Case 3: \( G_3 \) fires at \( t \). In the state at \( \text{close}, t > d \ell > d_{\sep} + \eta \), \( da = 0 \), and \( \text{emp}(0) \land \text{emp}(1) \) is false to the left of \( t \), but true to the right in \([t, t + d_{\sep}) \). In this case \( C^\perp \) is always empty, and hence, (AR2) is trivially verified.

Property (AR1) is non-trivial only for \( C = \emptyset \), as in Case 2. So we are in the situation
described just above. The guard \(G_3\) is true in \([t, t + d_{sep}]\), as the perturbation of \(da\) is much smaller than \(d_{adj}\). So one can take \(t = t\).

Property (AR3). We have 2 subcases to analyze: \(G^* = \{G_0, G_1, G_2, G_4\}\) (if \(C = \emptyset\)) and \(G^* = \{G_0, G_1, G_2, G_3, G_4\}\) (if \(C = \{G_3\}\)).

Case 3.1: \(C = \emptyset\). The state \(S(C)\) for this case is described above. Guards \(G_0, G_1\) are false in \(\mathcal{W}(\emptyset)\) because of \(dl\). Guard \(G_2\) is false in \(\mathcal{W}(\emptyset)\) because of open that is false at \(t\). Guard \(G_4\) with \(\eta\)-perturbed \(da = 0\) is false because of \(da \geq d_{adj}\), that does not depend on time.

Case 3.2: \(C = \{G_3\}\). The state \(S(G_3)\) in this case: close, \(t > dl > d_{sep} + wt\), \(da = t + d_{adj}\), and \(emp(0) \land emp(1)\) is true to the left of \(t\) up to \((t - d_{adj})\), and true to the right in \([t, t + d_{sep} - d_{adj}]\). Hence, guards \(G_0, G_1\) are false in \(\mathcal{W}(G_3)\) because of \(cmg(x)\) that is false in \([t - d_{adj}, t + d_{sep} - d_{adj}]\).

Guard \(G_3\) is false because of \(da\) that is bigger than \(d_{sep} + wt + d_{adj}\) in \(S(G_3)\). Guard \(G_4\) with \(\eta\)-perturbed \(da = 0\) is false everywhere to the left of \(t\) and false until \(t + d_{adj} - \eta\) to the right of \(t\).

Case 4: \(G_4\) fires at \(t\). In the state at \(t\): close, \(emp(0) \land emp(1)\) is true in \([t - d_{adj}, t + d_{sep} - d_{adj}]\), \(da = t > d_{sep} + wt + d_{adj}\), \(t > dl > d_{sep} + wt\). In this case \(C^\perp\) is always empty, and hence, (AR1) is trivially verified.

Property (AR1) is non-trivial only for \(C = \emptyset\), as in Case 2. So we are in the situation described just above. The guard \(G_4\) with \(\delta\)-perturbed \(da\) is true in \([t + \delta, t + d_{sep} - d_{adj}]\). So we can take \(t' = t + \delta\).

Property (AR3). We have 2 subcases to analyze: \(G^* = \{G_0, G_1, G_2, G_3\}\) (if \(C = \emptyset\)) and \(G^* = \{G_0, G_1, G_2, G_3, G_4\}\) (if \(C = \{G_4\}\)).

Case 4.1: \(C = \emptyset\). The state \(S(C)\) for this case is described above. Guards \(G_0, G_1\) are false in \(\mathcal{W}(\emptyset)\) because of \(dl\). Guard \(G_2\) is false in \(\mathcal{W}(\emptyset)\) because of open that is false at \(t\). Guard \(G_3\) with \(\eta\)-perturbed \(da = t\) is false because of \(da < d_{adj}\), that does not depend on time.

Case 4.2: \(C = \{G_4\}\). The state \(S(G_4)\) in this case: open, \(dl = 0\), \(da = 0\), and \(emp(0) \land emp(1)\) is true as describes above as an update cannot influence the input. Hence, guards \(G_0, G_1\) are false in \(\mathcal{W}(G_1)\) because of \(cmg(x)\) that is false in \([t - d_{adj}, t + d_{sep} - d_{adj}]\). Guards \(G_3\) and \(G_4\) are false because of close that is false now.

Property (AR4) for \(C_s\). Now we consider guards that are false at an instant \(T\) with the values of internal functions from a state of the form \(S(C)\), where \(C\) is a subset of guards that fire at \(t\).

Case A: Guard \(G_0\) or \(G_1\) is false at \(T\). Take \(G_0\), the case of \(G_1\) is similar. Guard \(G_0\) has only one external literal that is \(cmg(0)\) which is an input. So it is not important what guards fire at \(t\) and what is \(C\). If \(E(G_0, T, C)\) contains 2 disjoint \(\rho_0\)-intervals of \(cmg(0)\), they are \(d_{sep}\)-separated.

Case B: Guard \(G_2\) is false at \(T\). This guard contains only one external inequality. So \(E(G_2, T, C)\) cannot have 2 disjoint intervals. Again, as in the previous case, the situation at \(t\) plays no role.

Case C: Guard \(G_3\) is false at \(T\). This guard has 2 external literals: \(emp(0)\) and \(emp(1)\) that are inputs. Again, as in 2 previous cases the situation at \(t\) plays no role. Suppose that \(E(G_3, T, C)\) contains 2 disjoint intervals. If these disjoint intervals are of the same input, they are \(d_{min}\)-separated.

Suppose these disjoint intervals are of different inputs. Denote that of \(emp(0)\) by \(\sigma_0\), and that of \(emp(1)\) by \(\sigma_1\). Suppose, without loss of generality, that \(\sigma_0^{(r)} \leq \sigma_1^{(l)}\). Then \(\sigma_0^{(r)}\) is an instant of arrival of a train on track 0, and \(\sigma_1^{(l)}\) is an instant of departure of a train on track 1. According
to overall separation of train departures and arrivals, these two instants are $d_{cmg}$-separated.

**Case D:** Guard $G_4$ is false at $T$. This guard has 3 external literals: $emp(0)$, $emp(1)$, and $CT \geq da$. Suppose that $E(G_4, T, C)$ contains 2 disjoint intervals. If there are two disjoint input $\rho_0$-intervals then the argument of the previous case is applicable.

Suppose, without loss of generality, that the disjoint interval is a $\rho_0$-interval $\sigma_0$ of $emp(0)$ and a $\rho_0$-interval $\sigma$ of $CT \geq da$. If there are $\rho_0$-intervals of $emp(x)$ disjoint from $\sigma$, then due to our assumption, they are not disjoint, and in this case we consider an interval whose distance from $T$ is minimal; if there are two such $\rho_0$-intervals of $emp(x)$, then we take any one.

Clearly, $\sigma = [da, \infty)$. However, the value of $da$ depends on what guards fire at $t$.

If a non-empty subset of $\{G_0, G_1\}$ fires at $t$, then $da = 0$ whatever be $C$. Indeed, the controller may arrive at true $G_0$ or $G_1$ only after $d_{sep}$ time after a departure of all trains. Hence, $d_{adj} < d_{sep}$ ensures that $G_4$ fires at least $(d_{sep} - d_{adj})$ time before $t$. No other guard can fire between this firing and $t$.

If $G_2$ fires at $t$, then the previous update was done by a non-empty subset of $\{G_0, G_1\}$, hence, $da = 0$ whatever be $C$.

If $G_3$ fires at $t$, then either $da = 0$ ($C = \emptyset$) or $da = t + d_{adj}$ ($C = \{G_3\}$).

If $G_4$ fires at $t$, then either $da = t$ ($C = \emptyset$) or $da = 0$ ($C = \{G_4\}$).

If $da = 0$ then $\sigma = [0, \infty)$, and $E(G_0, T, C)$ cannot have 2 disjoint intervals, as all $\rho_0$-intervals of $emp(0)$ intersect $\sigma$.

So we have to consider two subcases of $da > 0$. Recall that $da \geq d_{sep} + d_{min}$.

**Case D.1:** $G_3$ fires at $t$, $C = \{G_3\}$ and $da = t + d_{adj}$, see Figure 15. As $G_4$ is false at $T$, then either one of $cmg(x)$ is true at $t$ or $T < da$. On the other hand, both $emp(x)$ are true in $\alpha = a(t, t + d_{sep} - d_{adj})$, as $t$ is the first moment when both tracks become empty. Hence, $\rho_0$-interval of any $emp(x)$ that is closest to any instant in $\alpha$ intersects $\sigma$. For this reason $T \notin \alpha$.

So any $\rho_0$-interval of $emp(x)$ is disjoint from $\alpha$.

If $t + d_{sep} \leq T$, then $\sigma_0$, that is disjoint from $\alpha$, must be to the left of $t - d_{min}$, as for any $x = 0, 1$ there is a $\rho_0$-interval of $emp(x)$ that intersects $\alpha$.

If $T < t$, then we have the same separation: $\sigma_0$ should lie to the left of $t - d_{min}$.

![Figure 15: Property (AR4), Case D.1](image)

**Case D.2:** $G_4$ fires at $t$, $C = \emptyset$ and $da = t$. In this case the both $emp(x)$ are true in $[t - d_{adj}, t + d_{sep} - d_{adj})$. The reasoning is similar to the previous case, but now the role of $t$ in Case D.1 is played by $t - d_{adj}$.

5 Correctness of the Implementation of Sturdy ASM

Recall that $A_0$ is a sturdy IA-ASM, $A_1 = \exists(A_0)$ and $\varepsilon$, $\eta$ are approximation parameters.
5.1 Formulation of the Main Theorem

The correctness of implementation \( \mathcal{I} \) of sturdy IA-ASM is stated in Theorem 1.

**Theorem 1 (Main Theorem)** For any \((\varepsilon, \eta)\)-sturdy IA-ASM \( \mathcal{A}_0 \) the DA-ASM \( \mathcal{A}_1 = \mathcal{I}(\mathcal{A}_0) \) gives a bisimilar \((\varepsilon, \eta)\)-implementation if the delay \( \xi \) satisfies

\[
\xi < \begin{cases} 
\frac{\eta}{(\mu - 1) \cdot p(\mu - 2) \cdot K^{2(\mu - 1)}} \\
\frac{\varepsilon}{2Kp(\mu - 1) \cdot p(\mu - 2) \cdot K^{2(\mu - 1)}}
\end{cases}
\]  

(Recall that \( \mu = M\nu \), see (17).)

We summarize the constraints on \( M, \nu, \mu, K, p \) (see Notations 5, Assumption 7 and (17)) for further references

\[
M \geq 1, \quad \nu \geq 2, \quad \mu \geq 2, \quad K \geq (\mu + 1) \geq 3, \quad p \geq 1. 
\]  

(24)

The rest of this section 5 is a proof of Theorem 1.

We assume for technical simplicity that there is only one FD-partition of functions of \( \mathcal{R}_F \), i.e., \(|\mathcal{D}(\mathcal{A}_0)| = 1\). The general case needs some attention when speaking about the update numbers.

Let an interpretation of input predicates and initial values of dynamic functions be given. These inputs and the initialization uniquely define a run \( \rho_0 \) of \( \mathcal{A}_0 \). Let \( \mathcal{A}_1 = \mathcal{I}(\mathcal{A}_0) \).

We have to prove (here Up and Down refer to the positions of \( \rho_0 \) and \( \rho_1 \) in Figure 9 and other figures below):

(DownUp). Any run \( \rho_1 \) of \( \mathcal{A}_1 \) with the same inputs and initial values as for \( \mathcal{A}_0 \) is \((\varepsilon, \eta)\)-close to \( \rho_0 \).

(UpDown). There is a run \( \rho_1 \) with the same inputs and initial values as for \( \mathcal{A}_0 \) that is \((\varepsilon, \eta)\)-close to \( \rho_0 \).

The both properties are proved below in the same induction that goes along the same lines modulo minor details. To prove (DownUp) we take any run \( \rho_1 \) of \( \mathcal{A}_1 \) for these inputs and initial values. To prove (UpDown) we construct a run that is denoted also \( \rho_1 \). We distinguish them by saying respectively “\( \rho_1 \) of (DownUp)” or “DownUp \( \rho_1 \)” and “\( \rho_1 \) of (UpDown)” or “UpDown \( \rho_1 \)”.

At instant 0, and thus in \((-\infty, 0]\) the run \( \rho_1 \) of (DownUp) is equal to \( \rho_0 \), as the initial conditions are the same for both runs. For instants in \((-\infty, 0]\) we set \( \rho_1 \) of (UpDown) equal to \( \rho_0 \).

5.2 Induction Hypothesis (IH)

Now we start to prepare our induction.

The function \( \pi \) introduced below is used to bound from above the difference of updated values of real-valued functions in both runs. Function \( \Lambda \) bounds the time shifts.
Definition 30 (Function $\pi$).

\[ \pi(x) = \xi \cdot x p^{(x-1)} K^{2x}, \text{ for } x \geq 0 \text{ (in particular } \pi(0) = 0). \]
\[ \Lambda(x) = K p \pi(x). \]  

In the inequalities (25)–(28) below we put integer constants $C_i > 0$ whose relations and lower bounds, that suffice for the proof, are made explicit.

From constraints on the involved parameters (24) one gets

\[ C_1 \xi \leq K^2 \xi \leq \pi(x) \text{ for } 1 \leq x, \text{ where } C_1 \geq 4. \]  

The following inequalities, that are easy to prove, will be often used:

\[ \pi(x) < \pi(x+1) \leq \pi(\mu-1) = \xi(\mu-1) \cdot p^{(\mu-2)} \cdot K^{2(\mu-1)} < \eta, \]
\[ 1 \leq x < (\mu - 1), \]  

\[ C_2 \xi \leq 2K^3 \xi < \varepsilon, \text{ where } C_2 \geq 7. \]  

\[ \forall x \leq (\mu - 1) \left( 0 < 2\xi + \Lambda(x) < \varepsilon - C_3 \xi \right), \text{ where } C_3 \geq 4, \text{ and } C_2 > C_3 + 2. \]  

Proof of (26)–(28). Formula (26) follows straightforwardly from the definition of $\pi$ just above, from the first inequality of (23) and from the constraints (24).

Formula (27) follows from the second inequality of (23) and (24).

Formula (28) is harder to prove, more precisely the second inequality as the first one follows from $\pi(x) \geq 0$. So we consider

\[ 2\xi + \Lambda(x) < \varepsilon - C_3 \xi, \text{ that is equivalent to } \]
\[ (C_3 + 2)\xi + K p \pi(x) < \varepsilon. \]  

For $x = 0$ it follows from (27) and non-negativeness of $\pi$.

Let $x \geq 1$. As $\pi$ is non-decreasing (26), in order to get (29), it suffices to prove

\[ (C_3 + 2)\xi + K p \pi(\mu - 1) < \varepsilon. \]  

From the second inequality of (23) we get

\[ \xi \cdot 2K p(\mu - 1) p^{(\mu - 2)} K^{2(\mu - 1)} = \xi \cdot 2(\mu - 1) p^{\mu - 1} K^{2\mu - 1} < \varepsilon. \]  

So it suffices to prove that the left hand side of (30) is bounded from above by the left hand side of (31):

\[ (C_3 + 2)\xi + K p \pi(\mu - 1) < \xi \cdot 2(\mu - 1) p^{\mu - 1} K^{2\mu - 1}. \]

Replacing $\pi(\mu - 1)$ by its definition in this inequality, we reduce it to

\[ (C_3 + 2)\xi + \xi \cdot (\mu - 1) \cdot p^{\mu - 1} K^{2\mu - 1} < \xi \cdot 2(\mu - 1) p^{\mu - 1} K^{2\mu - 1}. \]

Comparing the left and the right side of the latter inequality we reduce it to

\[ (C_3 + 2)\xi < \xi \cdot (\mu - 1) p^{\mu - 1} K^{2\mu - 1}, \]

and then to

\[ (C_3 + 2) < (\mu - 1) p^{\mu - 1} K^{2\mu - 1}. \]
The latter inequality immediately follows from constraints (24) if we take \((C_3 + 2) < K^3\). One can see that one can easily do respecting the other constraints, in particular (27) and \(C_2 > C_3 + 2\) from (28).

**Definition 31.** For a state \(S_C = (\rho_0 \downarrow \rho_1, C)\), where \(C\) is a set of guards, and \(f \in RF\):

- \(f\) is updated in \(S_C\) if there is a guard from \(C\) that updates \(f\) at \(t_0\) in \(\rho_0\). (Remark that if \(C = \emptyset\) then no \(f\) is updated by \(S_C\).)
- \(f\) is reset to its default value in \(S_C\) if it is updated by a guard from \(C\) at \(t_0\) in \(\rho_0\), and acquires its default value as the result of this update.

\[
\tilde{s_f}(S_C) = \begin{cases} 
  s_f[S_C] & \text{if } f \text{ is not updated in } S_C, \\
  s_f[S_C] + 1 & \text{if } f \text{ is updated in } S_C 
\end{cases}
\]

(Notice that \(f\) may have \(s_f[\rho_0(t_0)] = \nu - 1\), i.e., its maximal value before its update at \(t_0\) in \(\rho_0\), and nevertheless \(\tilde{s_f}(S_C) = \nu\) after its update in \(S_C\). But at the next time instant \(t'_0\) of updates in \(\rho_0\) its update number is taken correctly, i.e. \(s_f[\rho_0(t'_0)] = 0\), and thus, if \(f\) is updated at \(t'_0\) then \(\tilde{s_f}(\rho_0(t'_0)) = 1\). The function \(\tilde{s_f}\) is used for partial updates, i.e. for the case when \(C\) is a subset of guards that fire in some \(\xi\)-intervals in \(\rho_1\) close to \(t_0\).)

- \(\Sigma_s(S_C) = \sum_{f \in RF} \tilde{s_f}(S_C)\).
- \(\pi^* =_{df} \Sigma_s(\rho_0(t^*)), \Lambda^* =_{df} Kp\pi^*\)

(The time instant \(t^*\) is used in the Induction Hypothesis just below as a start instant of the induction step.)

It is evident that for all \(S\) and \(C\) from Definition 31

\[
\Sigma_s(S_\emptyset) \leq M(\nu - 1) = \mu - M \leq \mu - 1,
\]  
(32)

\[
\Sigma_s(S_C) \leq \mu = M\nu,
\]  
(33)

\[
C \subseteq C' \Rightarrow (\tilde{s_f}(S_C) \leq \tilde{s_f}(S_{C'}) \land \Sigma_s(S_C) \leq \Sigma_s(S_{C'}))
\]  
(34)

In the induction hypothesis just below we use expressions like “for an update in an ASM there exists an update in another ASM by the same operator”. By these expressions we mean the following correspondence between \(A_0\) et \(A_1\): the update in if-then numbered \(i\) in \(A_0\) of Figure 3 is the same as the update in if-then numbered \(i\) under the label 2 in \(A_1\) of Figure 5. This correspondence is quite evident from the purpose of \(J\).

**Induction Hypothesis (IH).** Suppose that for a time instant \(t^* \geq 0\) the following properties (IH1)–(IH3), illustrated in Figure 16, hold for the given \(\rho_1\) of (DownUp) and for the constructed part of \(\rho_1\) of (UpDown):

- (IH1) For each update \(u_0\) of a function \(f\) at \(\tau_0 \in [0, t^*]\) in \(\rho_0\) there exists an update \(u_1\) of \(f\) by the same operator (here we speak about an occurrence of an update operator into the program) at some instant \(\tau_1 \in [0, t^*]\) in \(\rho_1\) that is sufficiently close to \(u_0\).

More precisely, for each update \(u_0\) of a function \(f\) at \(\tau_0 \in [0, t^*]\) in \(\rho_0\), there exists an update \(u_1\) of \(f\) at some instant \(\tau_1 \in [0, t^*]\) in \(\rho_1\) by the same operator such that

- \(|\tau_1 - \tau_0| < \varepsilon\), and
for real-valued $f$:
- $|f[\rho_0](\tau_0^+) - f[\rho_1](\tau_1^+)| < \pi(\Sigma_s(\rho_0(\tau_0^+)))$, and the update number of $f$ is the same in $\rho_0(\tau_0^+)$ and $\rho_1(\tau_1^+)$. 

for predicate $f$:
- $f[\rho_0](\tau_0^+) = f[\rho_1](\tau_1^+)$. 

(IH2) For each update $u_1$ of a function $f$ at $\tau_1 \in [0, t^*]$ in $\rho_1$ there exists an update $u_0$ of $f$ by the same operator at some instant $\tau_0 \in [0, t^*]$ in $\rho_0$ that is sufficiently close to $u_1$. More precisely, for each update $u_1$ of a function $f$ at $\tau_1 \in [0, t^*]$ in $\rho_1$, there exists an update $u_0$ of $f$ at some instant $\tau_0 \in [0, t^*]$ in $\rho_0$ by the same operator such that
- $|\tau_1 - \tau_0| < \varepsilon$, and

for real-valued $f$:
- $|f[\rho_0](\tau_0^+) - f[\rho_1](\tau_1^+)| < \pi(\Sigma_s(\rho_0(\tau_0^+)))$, and the update number of $f$ is the same in $\rho_0(\tau_0^+)$ and $\rho_1(\tau_1^+)$. 

for predicate $f$:
- $f[\rho_0](\tau_0^+) = f[\rho_1](\tau_1^+)$. 

(IH3) $[t^* - \varepsilon, t^* + \varepsilon]$ does not contain any update in run $\rho_0$, and $[t^* - 2\xi, t^*]$ does not contain any update in run $\rho_1$. 

\[figure 16: Induction hypothesis\]

**Basis of the (IH).** Assume, without loss of generality, that the runs are extended to the left of instant 0 up to $-\infty$ with the values of functions at 0. All the runs we consider coincide on $[-\infty, 0]$. Assume that the instant $-\infty$ is considered as an update. Then for $t^* = -\infty$ the induction hypothesis holds. (If one prefers a finite number we may take $-2\varepsilon$ instead of $\infty$, but it slightly complicates the reasoning.)

**Thus, we will proceed with the induction step of (IH).**

In the proof below we will speak about $(\rho, t)$-intervals without making explicit precisions whether their ends belong to them or not. Recall that for an input function its $(\rho, t)$-interval is closed from the left and open from the right, and for internal functions it is open from the left and closed from the right. The exact form of the intervals under consideration will be clear from the context, and usually will not be important as we mainly deal with estimations of real-valued functions.
Notations 10 (Notation for intervals hiding the type of its ends).
A \((\rho, t)\)-interval with ends \(a, b\) will be denoted by \(\langle a, b \rangle\).

To prove the Theorem we proceed along the following lines:

- for the first time instant \(t_0\) of updates after \(t^*\) in \(\rho_0\) we prove that there is a collection of updates made by \(A_1\) that correspond to updates in \(\rho_0\) such that
  - their time instants are \((\varepsilon - C\xi)\)-close to \(t_0\), where \(C \geq 2\),
  - the values for real-valued functions updated by \(A_1\) are \(\eta\)-close to the corresponding ones in \(\rho_0\);
- we prove that there are no other updates by \(A_1\) besides the aforementioned ones in interval \((t_0 - \varepsilon, t_0 + \varepsilon)\);
- we prove that Induction Hypothesis can be extended until \(t_0 + \varepsilon\).

These updates of \(A_1\) constitute \(\rho_1\) either it is \((\text{DownUp})\) or \((\text{UpDown})\).

From now on \(t_0\) will denote the first time instant of updates after \(t^*\) in \(\rho_0\) if such an instant exists.

Recall that the notation of functions and formulas with tilde, used below in the proof, was introduced in Definition 10 of \(I\) in section 2.3.

In lemmas below, when we reason about an infinite interval, the case of its infinite end is much simpler than the case of its finite end. The case of infinite end is either straightforward or can be done if we treat \(\infty\) as a formal element extending the additive group of reals, in particular, by setting \(\infty - \infty = 0\).

5.3 Lemmas for the Induction Step of (IH)

We start with two simple technical lemmas. Lemma 2 does not use Induction Hypothesis, moreover it does not refer to any run.

Lemma 2. Let

- \(\Gamma\) be a positive real,
- \(\lambda_i\) be reals whose absolute values are bounded by \(\Gamma\), i.e., \(|\lambda_i| \leq \Gamma\),
- \(h_i\) be reals and \(H_i = h_i + \lambda_i\) (perturbed values of \(h_i\)),
- \(b, a_i\) be rational numbers and \(\omega\) is an inequality relation (like in (15)), \(|b|, |a_i| \leq K\), \(1 \leq i \leq r \leq p\) (we refer to the context of (15) and Notations 5).

- \(\langle \tau, \tau' \rangle = \{ t : b \omega t + a_1 h_1 + \cdots + a_r h_r \}\),
- \(\langle \theta, \theta' \rangle = \{ t : b \omega t + a_1 H_1 + \cdots + a_r H_r \}\).

Then

\[|\tau - \theta| < Kp\Gamma, \ |\tau' - \theta'| < Kp\Gamma.\]

Proof. Rewrite \(\langle \tau, \tau' \rangle\) in terms of \(H_i\):

\[\langle \tau, \tau' \rangle = \{ t : b \omega t + a_1 H_1 + \cdots + a_r H_r - (a_1 \lambda_1 + \cdots + a_r \lambda_r)\}.\]  \[(35)\]

From (35) we see that

\[\theta = \tau + a_1 \lambda_1 + \cdots + a_r \lambda_r, \ \theta' = \tau' + a_1 \lambda_1 + \cdots + a_r \lambda_r.\]  \[(36)\]
These equalities (36) give

\[
|\theta - \tau| < |a_1 \lambda_1| + \cdots + |a_r \lambda_r| \quad (37)
\]

\[
|\theta' - \tau'| < |a_1 \lambda_1| + \cdots + |a_r \lambda_r| \quad (38)
\]

According to the definition of parameters \(K, p\) (see notations 5) and the bound on \(|\lambda_i|\) formulas (37)-(38) imply

\[
|\theta - \tau| < Kp\Gamma
\]

\[
|\theta' - \tau'| < Kp\Gamma
\]

\[\blacksquare\]

The next Lemma 3 uses all statements of Induction Hypothesis (recall that \(\pi^*\) is introduced in Definition 31).

**Lemma 3.** For any real-valued function \(h\) there holds

\[\tilde{h}[\rho_1](t^*) = h[\rho_1](t^*), \quad |h[\rho_0](t^*) - h[\rho_1](t^*)| < \pi^*,\]

for any internal literal \(f\) there holds

\[f[\rho_0](t^*) = f[\rho_1](t^*) = \tilde{f}[\rho_1](t^*).\]

**Proof.** Let \(g\) be any internal function. Let \(u_0\) be its last update in \(\rho_0\) before \(t^*\), and denote by \(\tau_0\) the instant of this update. (IH3) implies that

\[\tau_0 < t^* - \varepsilon.\]  

(II1) says that there is an update \(u_1\) of \(g\) in \(\rho_1\) by the same operator at some instant \(\tau_1\) that is \(\varepsilon\)-close to \(\tau_0\):

\[|\tau_0 - \tau_1| < \varepsilon.\]  

(40)

Inequalities (40) and (39) imply that \(\tau_1 < t^*\), even more, taking into account (IH3), that \(\tau_1 < t^* - 2\xi\).

Suppose that \(u_1\) is not the last update of \(g\) before \(t^*\) in \(\rho_1\). Thus, there is an update \(u'_1\) of \(g\) in \(\rho_1\) at some \(\tau'_1\) such that

\[\tau_1 < \tau'_1 < t^* - 2\xi.\]  

Then according to (IH2) there is an update \(u'_0\) of \(g\) in \(\rho_0\) at some instant \(\tau'_0\) such that

\[|\tau'_0 - \tau'_1| < \varepsilon.\]  

Due to Lemma 1, (i) this instant \(\tau'_0\) is not in the \(2\varepsilon\)-neighborhood of \(\tau_0\)

\[|\tau_0 - \tau'_0| > 2\varepsilon.\]  

(43)

Two cases are possible: either \(\tau'_0\) lies to left of \(\tau_0\) or to right of \(\tau_0\).

Suppose \(\tau'_0 < \tau_0\). Then from (43) we have \(\tau'_0 < \tau_0 - 2\varepsilon\) (\(\tau'_0\) is ‘far’ to the left from \(\tau_0\)), and this contradicts to (42) (\(\tau'_0\) is ‘close’ to \(\tau'_1\)), (41) (\(\tau'_1\) lies to the right of \(\tau_1\)) and (40) (\(\tau_1\) is ‘close’ to \(\tau_0\)). Hence, this case is impossible.

Suppose \(\tau_0 < \tau'_0\). From (IH3) we know that there is no updates in \(\rho_0\) in \([t^* - \varepsilon, t^* + \varepsilon]\). Together with (42), (41) (whose conjunction says that \(\tau'_0\) is \(\varepsilon\)-close to some instant that lies to
the left of $t^*$) this implies that the instant $\tau'_0$ cannot be greater than $t^*$. But in this case $\tau_0$ is not the instant of the last update of $g$ in $\rho_0$ before $t^*$ — again a contradiction.

Hence, $\tau_1$ is the instant of last update of $g$ in $\rho_1$ before $t^*$, and this update is $u_1$ that was defined due to (IH1). As there are no updates at $t^*$ in both runs then the values of internal functions at $t^*$ in $\rho_j, j = 0, 1$, are the same as at $\tau^+_j$. Taking into account that $\bar{g}[\rho_1](t^*_i) = g[\rho_1](t^*_i)$ for some $t^*_i$ and $t^{*'}$ from the same $\xi$-interval, we get the conclusion of the lemma for internal real-valued functions and Boolean-valued function from (IH1).

For internal inequalities the conclusion follows from the obtained bounds for internal real-valued functions and (32), (26) and $\eta$-stability of internal inequalities of $A_0$. □

The next Lemma 4 does not use Induction Hypothesis, though it refers to the runs under consideration.

**Lemma 4. Let**

- $H$ be an external inequality formula of a guard of $A_0$, and $T_0$ and $T_1$ be two time instants,
- $\rho_1$ be either the DownUp run or the UpDown partial run, and the UpDown partial run $\rho_1$ be defined up to $\max\{T_0, T_1\}$,
- $\langle \tau_0, \tau'_0 \rangle$ be the $(\rho_0, T_0)$-interval of $H$ (true or false),
- $\langle \tau_1, \tau'_1 \rangle$ be the $(\rho_1, T_1)$-interval of $H$ (true or false),
- for any real-valued function $h$ of $H$ there hold
  \[
  \bar{h}[\rho_1](T_1) = h[\rho_1](T_1), \quad |h(\rho_0)(T_0) - h[\rho_1](T_1)| < \Gamma
  \]

**Then**

(a) $|\tau_0 - \tau_1| < 2\xi + Kp\Gamma$, $|\tau'_0 - \tau'_1| < 2\xi + Kp\Gamma$ (i.e., the intervals $\langle \tau_0, \tau'_0 \rangle$ and $\langle \tau_1, \tau'_1 \rangle$ are in some way close),

(b) the truth value of $H$ in $\langle \tau_0, \tau'_0 \rangle$ and the truth value of $H$ in $\langle \tau_1, \tau'_1 \rangle$ are the same.

**Proof.** Suppose that $H$ has the form $b \omega CT + a_1 h_1 + \cdots + a_r h_r$, and is true in $\langle \tau_0, \tau'_0 \rangle$. Then $\langle \tau_0, \tau'_0 \rangle = \{ t : b \omega t + a_1 h_1[\rho_0](T_0) + \cdots + a_r h_r[\rho_0](T_0) \}$. If $H$ is false in this interval then its negation has a similar form as $H$, and the reasoning below holds for its negation.

Apply Lemma 2 with $h_i = h_i[\rho_0](T_0)$ and $H_i = h_i[\rho_1](T_1)$. Then for $\langle \tau_0, \tau'_0 \rangle$ and $\langle \theta_0, \theta'_0 \rangle = \{ t : b \omega t + a_1 h_1[\rho_1](T_1) + \cdots + a_r h_r[\rho_1](T_1) \}$ we have from the premise of the lemma and the definition of $K, p$

$$|\tau_0 - \theta_0| < Kp\Gamma, \quad |\tau'_0 - \theta'_0| < Kp\Gamma.$$  \hspace{1cm} (44)

The equalities $\bar{h}[\rho_1](T_1) = h[\rho_1](T_1)$ (the premise of the lemma) permit to rewrite $\langle \theta_0, \theta'_0 \rangle$ as $\langle \theta_0, \theta'_0 \rangle = \{ t : b \omega t + a_1 h_1[\rho_1](T_1) + \cdots + a_r h_r[\rho_1](T_1) \}$.

The $(\rho_1, T_1)$-interval $\langle \tau_1, \tau'_1 \rangle$ of $H$ is $\langle \tau_1, \tau'_1 \rangle = \{ t : b \omega CT[\rho_1](t) + a_1 h_1[\rho_1](T_1) + \cdots + a_r h_r[\rho_1](T_1) \}$.

From this and previous formulas for intervals, from (18) and (44) we get the conclusion of the lemma. □

The next Lemma 5 relies upon Induction Hypothesis as it uses Lemma 3.
Lemma 5. Let
- \( \rho_0 \) be either the DownUp run or the UpDown partial run satisfying Induction Hypothesis,
- \( H \) be an external inequality of a guard of \( A_0 \),
- \( \langle \tau_0, \tau'_0 \rangle \) be the \((\rho_0, t^*)\)-interval of \( H \),
- \( \langle \tau_1, \tau'_1 \rangle \) be the \((\rho_1, t^*)\)-interval of \( \tilde{H} \).

Then
1. \(|\tau_0 - \tau_1| < 2\xi + \Lambda^*\), \(|\tau'_0 - \tau'_1| < 2\xi + \Lambda^*\) (i.e., the intervals \(\langle \tau_0, \tau'_0 \rangle\) and \(\langle \tau_1, \tau'_1 \rangle\) are close),
2. the truth value of \( H \) in \(\langle \tau_0, \tau'_0 \rangle\) and the truth value of \( \tilde{H} \) in \(\langle \tau_1, \tau'_1 \rangle\) are the same.

Proof. Straightforwardly follows from Lemmas 4, 3.

The previous lemmas treat literals. Our goal is to compare the behavior of firing guards and their respective updates in \( \rho_0 \) and the corresponding run of \( A_1 \). The next two Lemmas 6, 7 are technical, and do not rely upon Induction Hypothesis (IH).

Lemma 6. Let
- \( g \in RF \) is updated in \( \rho_0 \) and \( \rho_1 \) by the same term \( g = c_0CT + c_1\varphi_1 + \cdots + c_q\varphi_q \) (where \( \varphi_i \in RF \)) in the same \textbf{if-then}-operator at respectively \( t_0 \) and \( t_1 \),
- the values of functions \( \varphi_i \) in these updates are respectively \( f_i \) in \( \rho_0 \) and \( \tilde{f}_i \) in \( \rho_1 \), and \(|f_i - \tilde{f}_i| < \Gamma\),
- \(|t_0 - t_1| + 2\xi \leq \Delta\).

Then for the values \( g_0 \) and \( g_1 \) of \( g \) after these updates in respectively \( \rho_0 \) and \( \rho_1 \) there holds

\[
|g_1 - g_0| < K\Delta + pK\Gamma
\]  

(45)

Proof. If \( g \) is reset to the default value by the \textbf{if-then}-operator then its value in both runs is the same. Otherwise the update gives the equality

\[
g_0 = c_0t_0 + c_1f_1 + \cdots + c_qf_q
\]

in run \( \rho_0 \) and the equality

\[
g_1 = c_0\tilde{CT}[\rho_1](t_1) + c_1\tilde{f}_1 + \cdots + c_q\tilde{f}_q,
\]

where \( 0 < t_1 - \tilde{CT}[\rho_1](t_1) < 2\xi \) (see(18)), in run \( \rho_1 \).

Therefore we have

\[
|g_1 - g_0| = |c_0(\tilde{CT}[\rho_1](t_1) - t_0) + c_1(\tilde{f}_1 - f_1) + \cdots + c_q(\tilde{f}_q - f_q)| < K\Delta + pK\Gamma.
\]

Lemma 7. If \( 0 \leq x \leq (\mu - 1) \), \( \Gamma = \pi(x) \) and \( \Delta \leq 2\xi + Kp\pi(x) \) then

\[
K\Delta + pK\Gamma \leq \pi(x + 1)
\]  

(46)

Proof. From the premise of Lemma 7 and Definition 30 of \( \pi \) we deduce

\[
K\Delta + pK\Gamma \leq K(2\xi + Kp\pi(x)) + pK\pi(x) = 2K\xi + pK\pi(x)(K + 1) = 2K\xi + pK(K + 1)(\xi \cdot \pi \cdot p^{(x - 1)}K^{2x}) = \xi(2K + (K + 1)x \cdot p^x K^{2x+1}) =
\]
\[ \xi \left( 2K + x \cdot p^x K^{(2x+2)} + x \cdot p^x K^{(2x+1)} \right) \leq \pi(x + 1) = \xi \left( x \cdot p^x K^{(2x+2)} + p^x K^{(2x+2)} \right) \]

— to prove the latter inequality we notice that it is equivalent to

\[ 2K + x \cdot p^x K^{(2x+1)} \leq p^x K^{(2x+2)}, \] (47)

and the inequality (47) is equivalent to \( 2 \leq p^x K^{2x}(K - x) \) that follows from (17), \( x \leq \mu \) and constraints (24); we recall that the case \( x = 0 \) is not excluded. \( \blacksquare \)

**Lemma 8.** Let \( t > t^* + \varepsilon \). If no guard fires in \([t^*, t]\) in \( \rho_0 \) then no guard \( \tilde{G} \) fires in \([t^*, t - \Lambda^* - 2\varepsilon]\) in \( \rho_1 \), where \( \rho_1 \) is either the DownUp run or the UpDown partial run such that for any guard \( G \) the functions used in \( G \) are defined up to instant \( (t - \Lambda^* - 2\varepsilon) \).

**Proof.** Suppose that no guard fires in \([t^*, t]\) in \( \rho_0 \), and thus due to (IH3), no guard is true in \([t^* - \varepsilon, t]\) in \( \rho_0 \).

Suppose that there is a guard \( \tilde{G} \) that fires in \( \rho_1 \) in \([t^*, t - \Lambda^* - 2\varepsilon]\).

Let \( t_1 \) be the first instant after \( t^* \) where some guard \( \tilde{G} \) fires in \( \rho_1 \). Clearly, \( \tilde{G} \) is true in \( \rho_1 \) at \( t_1 \). Let \( \zeta_1 \) be the \( \xi \)-interval of \( \rho_1 \) containing \( t_1 \).

From our assumption and (IH3) we have

\[ t^* < t_1 \leq t - \Lambda^* - 2\xi. \] (48)

The values of internal functions with which \( \tilde{G} \) is evaluated and with which it fires are determined during the backup phase that starts and terminates in \( \zeta_1 \). Thus, it follows from (IH3) and (48) that these are values from \( \rho_1(t^* - 2\xi) \) that are the same as in \( \rho_1(\tau) \) for all \( \tau \in [t^* - 2\xi, t_1] \), in particular, they are the same as in \( \rho_1(t^*) \).

From the premise of the lemma and (IH3) we see that \( G \) is false in \( \rho_0 \) at all instants from \([t^* - \varepsilon, t]\). In particular, it follows from (48) that \( G \) is false in \( \rho_0 \) at all instants from \( \zeta_1 \) with the values of internal functions from \( \rho_0(t^*) \), as the latter are the same everywhere in \([t^* - \varepsilon, t]\).

Thus, the values of internal functions used for the evaluation of \( \tilde{G} \) in \( \rho_1 \) at \( t_1 \) and for the evaluation of \( G \) in \( \rho_0 \) at \( t_1 \) are taken at \( t^* \) in the respective runs. The internal literals of \( G \) are true at \( t^* \) in both runs due to Lemma 3 and to the hypothesis that \( \tilde{G} \) is true at \( t_1 \). Hence, for any internal literal its \((\rho_1, \tau)\)-interval coincides with its \((\rho_1, t^*)\)-interval for all \( \tau \in [t^* - 2\xi, t_1] \) and contains \([t^* - 2\xi, \infty)\), and its \((\rho_0, \tau)\)-interval coincides with its \((\rho_0, t^*)\)-interval for all \( \tau \in [t^* - \varepsilon, t]\) and contains \([t^* - \varepsilon, \infty)\).

Therefore, the guard \( G \) cannot consist only of internal literals, as in this case it is true at \( t_1 \), as we indicated above.

Hence, we consider the case when \( G \) contains inputs and/or external inequalities.

Denote by \( Z \) the finite set of time instants whereat the values of functions of \( G \) are updated in \( \zeta_1 \), i.e., backed up during the backup phase of \( A_1 \) in \( \zeta_1 \). Clearly, the distance between \( t_1 \) and any element of \( Z \) is less than \( \xi \). Denote by \( \hat{Z} \) the minimal interval containing \( Z \). Clearly, \( \hat{Z} \) lies strictly inside \([\zeta_1^{(l)}, t_1]\) due to the construction of \( A_1 \).

Whatever be \( \tau \in [t^* - \varepsilon, t]\), the (true) \((\rho_0, \tau)\)-interval of any literal of \( IneqG \) coincides with its \((\rho_0, t^*)\)-interval. Recall that the \((\rho_0, \tau)\)-interval of \( G \) is the intersection of all \((\rho_0, \tau)\)-intervals
of its literals. Notice that for a given inequality its $\rho_0$-interval and $\rho_1$-interval either are both left unbounded or are both right unbounded as they are defined by inequalities of the same form. For example, a left-unbounded inequality in both runs has the form like $0 \leq \tau + \text{const}$, where $\tau$ is the time variable, with different $\text{const}$.

**Claim 1.** If $G$ contains an inequality whose $(\rho_0, t^*)$-interval is disjoint from $(t^* - \varepsilon, t)$ then $\tilde{G}$ cannot be true (and thus, cannot fire) at $t_1$ in $\rho_1$.

**Proof.** Let $H$ be an inequality that satisfies the premise of the claim, and $\alpha$ and $\alpha'$ be respectively its $(\rho_0, t^*)$-interval and $(\rho_1, t^*)$-interval. They both are left or right unbounded, as we mentioned above. Let $X$ and $X'$ be respectively their finite ends.

Lemmas 3, 5 imply that $|X - X'| < 2\xi + \Lambda^*$. Suppose $t \leq X$ (as $H$ is false in $(t^* - \varepsilon, t)$, $X$ is the left end of $\alpha$). Then $(t - \Lambda^* - 2\xi) < X'$. This bound, (48) and the position of $\tilde{Z}$ with respect to $t_1$ imply that the value of $H$ backed up in $\zeta_1$ in $\rho_1$ is false. Hence, in the case $t \leq X$ the guard $\tilde{G}$ cannot be true at $t_1$ in $\rho_1$.

Suppose $X \leq (t^* - \varepsilon)$ (this time $X$ is the right end of $\alpha$ because $H$ is false in $(t^* - \varepsilon, t)$). Then from (28), (48) and the position of $\tilde{Z}$ with respect to $t_1$ we get

$$X' \leq (t^* - \varepsilon + 2\xi + \Lambda^*) < t^* - 2\xi < \tilde{Z}^{(1)}.$$ 

Thus, again $\tilde{G}$ cannot be true at $t_1$ in $\rho_1$.

**Claim 2.** Let $T \in (\zeta_1^0, t_1]$, and let $E(G, T, \emptyset)$ be the set of $\rho_0$-intervals of literals of $G$ defined in Assumption 8(AR4) with $t = t_0$. If $E(G, T, \emptyset)$ contains two disjoint intervals then $\tilde{G}$ cannot be true (and thus, cannot fire) at $t_1$ in $\rho_1$.

**Proof.** Suppose that the set $E(G, T, \emptyset)$ contains two disjoint $\rho_0$-intervals. Remind that $E(G, T, \emptyset)$ contains (true) $(\rho_0 \cup_{t_0} \emptyset)$-intervals that are equal to their respective $(\rho_0, t^*)$-intervals by definition of $t_0$. Then according to Assumption 8(AR4) it contains two $\rho_0$-intervals $\alpha$ and $\alpha'$ one of which is $\varepsilon$-separated from $T$. Indeed they are the closest to $T$ and they are $2\varepsilon$-separated. Thus, one of these intervals is $(\varepsilon - 2\xi)$-separated from any instant from $Z$. Denote by $\alpha$ this $\rho_0$-interval and by $H$ the respective literal of $G$. The literal $H$ is either input or inequality.

**Case I:** $H \in \text{Inp}G$. Recall that the inputs in both runs are the same. The interval $\alpha$ is a $\rho_0$-interval of $H$ closest to $T$. A $\rho_0$-interval of $H$ containing the instant of backup of $H$ (this instant is in $Z$) is $\xi$-close to $\alpha$. Denote the latter interval by $\beta$ (a priori, $\alpha$ may be closer to $T$ than $\beta$).

From (27) and the mentioned $(\varepsilon - 2\xi)$-separation of $\alpha$ we see that $\beta$ is disjoint from $\tilde{Z}$. Hence, the value of $H$ backed up in $\rho_1$ in $Z$ is false, and therefore $\tilde{G}$ cannot be true at $t_1$ in $\rho_1$.

**Case II:** $H \in \text{Ineq}G$. In this case $\alpha$ is a unique $\rho_0$-interval, and is unbounded from one side. From (48), the choice of $\zeta_1$ as the first $\xi$-interval wherein some tilde-guard fires after $t^*$ and our remarks above we see that $\alpha$ is also $(\rho_0, t^*)$-interval. Let $X_0$ be the finite end of $\alpha$, and let $X_1$ the finite end of $(\rho_1, t^*)$-interval of $H$; recall that the latter interval is used in the evaluation of $H$ at $t_1$. Denote the latter interval by $\alpha'$.

Lemmas 3, 5 imply that $|X_0 - X_1| < 2\xi + \Lambda^*$. From this inequality, $(\varepsilon - 2\xi)$-separation of $X_0$ from $Z$ and (28) we see that $\alpha'$ is disjoint from $\zeta_1$, and hence, $\tilde{G}$ cannot be true at $t_1$. Contradiction.

We go on with the proof of the lemma. Let $E(G, t_1, \emptyset)$ be the set defined in Assumption 8(AR4). According to Claim 2 this set cannot have two disjoint intervals. Hence the intersection of its intervals is not empty.
Indeed, take an interval $\sigma_1$ with the biggest left end and an interval $\sigma_2$ the smallest right end. Then $\langle \sigma_1, \sigma_2 \rangle$ is not empty and is included in any other interval.

Denote the intersection of all intervals from $E(G, t_1, \emptyset)$ by $\sigma$. As $G$ is false everywhere in $[t^* - \varepsilon, t]$ in $\rho_0$, this interval $\sigma$ does not intersect $[t^* - \varepsilon, t]$. Let $X$ be the end of $\sigma$ closest to $[t^* - \varepsilon, t]$. It is an end of a $\rho_0$-interval of a literal $H$ of $G$. Literal $H$ cannot be an input, as all input $\rho_0$-intervals of literals of $G$ intersect $Z$, and thus $\zeta_1$. And $E(G, t_1, \emptyset)$ consists of intervals closest to $t_1$.

Therefore $X$ is the finite end of an inequality. According to the choice of $t_1$ the $(\rho_1, t_0)$-interval of this inequality coincides with its $(\rho_1, t^*)$-interval. Claim 1 gives a contradiction.

Hence, our assumption that $G$ fires in $\rho_1$ in $[t^*, t - \Lambda^* - 2\varepsilon]$ is false. ■

**Lemma 9.** Let $G$ be a guard, $\text{Inp}G \neq \emptyset$, $\zeta$ be a time interval, and each input is true somewhere in $\zeta$. Suppose that there is no instant in $\zeta$ where all literals of $\text{Inp}G$ are true (in $\rho_1$ or in $\rho_0$ — the inputs are the same). Then there exist $T \in \zeta$, $I, I' \in \text{Inp}G$, $I \neq I'$, such that the $\rho_0$-intervals (that are also $\rho_1$-intervals) of $I$ and $I'$ closest to $T$ are disjoint.

**Proof.** The set $\text{Inp}G$ contains at least two inputs.

Construct the following sequence of $\rho_0$-intervals (that are also $\rho_1$-intervals) of literals of $\text{Inp}G$. Among all $\rho_0$-intervals of literals from $\text{Inp}G$ that intersect $\zeta$ choose a $\rho_0$-interval with the smallest right end. Denote it by $\alpha_1$, and denote by $I_1$ its literal from $\text{Inp}G$. From $\rho_0$-intervals of literals from $\text{Inp}G \setminus \{I_1\}$ that intersect $\zeta$ again choose a $\rho_0$-interval with the smallest right end. Denote it by $\alpha_2$, and denote by $I_2$ its literal from $\text{Inp}G$. Continue this process until all literals of $\text{Inp}G$ are treated. In this way we get $\alpha_1, \alpha_2, \ldots, \alpha_m$ and $I_1, I_2, \ldots, I_m$, $m \geq 2$.

Consider the left ends of $\alpha_i$. If all of them are smaller than the right end of $\alpha_1$ then there is an instant in the intersection of all $\alpha_i$. But this is excluded by the premise of the lemma. Thus, there is $\alpha_{i_0}$, $1 \leq i_0 \leq m$, such that $\alpha_1(\alpha_{i_0}) \leq \alpha_1(\alpha_{i})$.

Let $T = \alpha_{i_0}(\alpha_1)$ (recall that $\alpha_1$ are left open and hence, $T \notin \alpha_1$). Clearly, $\alpha_1$ and $\alpha_{i_0}$ are $\rho_0$-intervals of respectively $I_1$ and $I_{i_0}$ closest to $T$ (recall that $\rho_0$-interval of the same input are separated), and they are disjoint. ■

## 5.4 Induction Step for (IH)

If no guard fires in $\rho_0$ after $t^*$ then no guard fires in $\rho_1$ after $t^*$ due to Lemma 8.

Suppose that some guard fires in $\rho_0$ after $t^*$. Let $t_0$ be the first instant after $t^*$ whereat some guards fire in $\rho_0$, and let $G$ be all guards that fire at $t_0$.

From (IH3) we have

$$t^* + \varepsilon < t_0.$$  \hspace{1cm} (49)

Now we start to construct an increasing sequence of $\xi$-intervals $\zeta_1, \ldots, \zeta_m$, $1 \leq m \leq L$, (maybe adjacent or not) and a sequence of non empty subsets $G_1, \ldots, G_m$ of $G$ such that the updates fired by the guards from these sets ‘cover’ the updates fired by guards from $G$ as stated in properties (A)–(E) below, see Figure 17 (there can be $t^* < t_0 - \varepsilon < t^* + \varepsilon$).

These properties (A)–(E) either state properties of $\text{DownUp} \rho_1$ or state properties of an extension of $\text{UpDown} \rho_1$ beyond $t^*$; the extension is denoted by the same symbol $\rho_1$. The latter is defined with the help of updates that are done by $A_1$ in intervals $\zeta_1, \ldots, \zeta_m$. Below, when we speak about an update executed by $A_1$, we mean an update either in $\text{DownUp} \rho_1$ or in this extension of $\text{UpDown} \rho_1$.

We use notations from Assumption 8, in particular $C^\perp$. 
Figure 17: Step of induction (H1)–(H3).

Notations 11.

- \( \hat{G}_k = \bigcup_{i=1}^{k} G_i, \) \( \hat{G}_0 = \emptyset, \) \( 0 \leq k \leq m. \)

- The following two sequences of sets of guards are defined by a simultaneous recursion assuming that \( \hat{G}_k \) is defined:
  \( G_0^{\Delta} = \emptyset, \) \( G_k^{\Delta} = (\hat{G}_k \setminus G_{k-1}^{\Delta}), \)
  \( G_0^{\Lambda} = \emptyset, \) \( G_k^{\Lambda} = \bigcup_{i=1}^{k} G_i^{\Delta}, \) \( 1 \leq k \leq m. \)

  \( (G_k^{\Delta} \) is the set of guards \( G \notin \hat{G}_k \) such that \( G \) is locked by firing guards from \( \hat{G}_k, \) but not locked by previously fired guards, i.e., by guards from \( \hat{G}_i \) for some \( i, i < k. \))

- \( F_k = (\hat{G}_k \cup G_k^{\Lambda}). \) (Notice that \( F_0 = \emptyset. \) The set \( F_k \) consists of all guards that fire in \([\xi^{(l)}, \xi^{(r)}_k]\) or are locked by those that fire.)

- \( S_i = (\rho_0 \downarrow t_0 F_i) \)

- \( S_i \)-interval of a conjunction of literals is the \( (S_i, \rho_0, t_0) \)-interval of this conjunction defined in the same way as \( (S_i, \rho_0, t_0) \)-interval of a guard in Definition 27. This defines, in particular, \( S_i \)-interval of a guard or of a literal.

- \( \pi_i = (\Sigma_s(S_i)), \) \( \Lambda_i = Kp\pi_i. \) (Notice that \( \pi_0 = \pi^* \) and \( \Lambda_0 = \Lambda^* \). The monotonicity \( \pi_i \leq \pi_{i+1} \) and \( \Lambda_i \leq \Lambda_{i+1} \) follow from (34) and from evident inclusion \( F_i \subseteq F_{i+1} \).)

□

Remark 16 (Evolution of upper bound on \( \Sigma_s \)). Clearly,

\[
\Sigma_s(S_0) = \Sigma_s(\rho_0(t_0)) \leq M(\nu - 1), \quad \Sigma_s(\rho_0 \downarrow t_0 G) = \Sigma_s(\rho_0(t_0^+)) \leq \mu.
\]

(50)

Indeed, \( \Sigma_s(\rho_0(t_0)) \) is calculated before updates at \( t_0 \) in \( \rho_0 \) with \( s_f[\rho_0(t_0)], f \in RF, \) that are the update numbers of all \( f \) and, thus less than \( \nu - 1. \) Each function \( f \) is updated at most once at \( t_0, \) so the value of \( \Sigma_s(\rho_0(t_0)) \) can be augmented by adding at most \( M. \) In any case, if we
consider consecutive partial updates (that correspond to respective updates in \( \rho_1 \)), the value of \( \Sigma_\pi \) is \( \leq (\mu - 1) \) before the last update. Hence, we can apply Lemma 7 in this situation.

Properties (A)–(E).

(A1) \( \zeta_1^{(r)} \leq \zeta_2^{(l)} < \zeta_2^{(r)} \leq \cdots \leq \zeta_m^{(l)} < \zeta_m^{(r)} \).

(A2) The sets \( G_i \) and the other ones derived from them have the properties:

\[
G_i \cap G_j = \emptyset, \quad (\hat{G}_i \cap G_i^\alpha) = \emptyset \quad \text{for } i \neq j, \ 0 \leq i, j \leq m \tag{51}
\]

\[
\hat{G}_{i-1} \subseteq \hat{G}_i, \ F_{i-1} \subseteq F_i, \ F_i \subseteq F_m = G \quad \text{for } 1 \leq i \leq m \tag{52}
\]

(A3) Each guard of \( G_i \) is a guard of \( G \) whose tilde-image fire in \( \rho_1 \) at some instant \( \tau \in \zeta_i \).

The respective updates are done with the values of internal functions at \( t^* \) (however, the guards that fire may use values updated by \( G_j, j < i \)).

(B) For any guard \( G \in G_i \) for the instant \( t_G \) where it fires in \( \zeta_i \) in \( \rho_1 \)

\[
t_0 - \Lambda_{i-1} - 2\xi < t_G < \frac{\zeta_i^{(r)}}{1} < t_0 + \Lambda_{i-1} + 2\xi.
\]

(Consequently, \( |t_0 - \zeta_i^{(r)}| < 2\xi + \Lambda_{i-1} \) for \( 1 \leq i \leq m \).)

(C) For any real-valued function \( f \), if \( f \) is updated in \( \rho_0 \) by firing a guard \( G \in G_i \) then \( f \) is updated in \( \rho_1 \) by firing \( \hat{G} \), and

\[
|f[\rho_0](t_0^\ast) - f[\rho_1](\zeta_i^{(r)})| \leq \pi_i.
\]

In any case, whether \( f \) is updated or not by a guard \( G \in G_i \), there holds

\[
|f[(\rho_0 \downarrow t_0, F_i)](t_0^\ast) - f[\rho_1](\zeta_i^{(r)})| \leq \pi_i.
\]

(D) After firing guards from \( \hat{G}_i \) the value of each Boolean-valued function that is updated by a guard from \( F_i \) in \( \rho_0 \) at \( t_0 \) is equal to the value of this function in state \( \rho_1(\zeta_i^{(r)}) \), \( 1 \leq i \leq m \). The other Boolean-valued functions preserves their values.

(E) The constructed updates of \( A_1: \) (E0) extend UpDown \( \rho_1 \) up to \( \zeta_m^{(r)} \); (E1) are exactly updates of DownUp \( \rho_1 \) in \( [t^*, \zeta_m^{(r)}] \).

We construct sequences \( \zeta_1, \ldots, \zeta_m \) and \( G_1, \ldots, G_m \) by induction that will be referred below as (A)–(E) induction.

Recall our way of reasoning concerning UpDown and DownUp cases. In any case a run \( \rho_0 \) of \( A_0 \) for given inputs is fixed. For a time instant, whereat some guards of \( A_0 \) fire, we show that there are instants whereat some tilde-image of some of these guards should fire in \( A_1 \). The updates done by these tilde-image provide a state close to the state produced by the guards in \( A_0 \), and there are no other tilde-guards that fire. Thus, these tilde-guards should fire in \( \rho_1 \) in the case DownUp when \( \rho_1 \) is given, and we put these update by tilde-guards into \( \rho_1 \) in the case UpDown when no \( \rho_1 \) is given and we are constructing \( \rho_1 \). We concretize this in more detail in appropriate places below.

Basis of induction hypothesis (A)–(E): construction of \( \zeta_1 \) and \( G_1 \).

Take any \( G \in G \). It is true at \( t_0 \) in \( \rho_0 \), and for its evaluation the values of its functions at \( t^* \) are used. Compare this values with the values of the same functions in \( \rho_1 \) at \( t^* \). Such a
comparison is provided by Lemma 3. This lemma says that the values of internal real-valued functions from \( \rho_1(t^*) \) are \( \pi^* \)-close to the values of these functions in \( \rho_0(t^*) \), and the values of internal literals of \( \mathcal{A}_0 \) are equal in.

Apply Assumption 8(AR1) to guard \( G, C = \emptyset, t = t_0 \), state \((\rho_0 \downarrow \downarrow G) \) and \( \delta = \pi^* \). Then \( K\rho G_0 = \Lambda^* \) and \((\rho_0 \downarrow \downarrow G) = \rho_0(t_0) \). No guard from \( G \), in particular \( G \), is locked by \( \rho_0(t_0) \), as all of them fire at \( t_0 \) (see (21)).

From this assumption we conclude that there exists \( T_G \in [t_0 - \Lambda^*, t_0 + \Lambda^*] \) such that \( G \) is true with the values of internal functions from \( \rho_1(t^*) \) and the values of inputs at any \( \tau \in [T_G, t_0 + \varepsilon] \). In particular, \( G \) is true at some instant from \([t_0 - \Lambda^*, t_0 + \Lambda^*] \) with the values of internal functions from \( \rho_1(t^*) \). Remark that here we refer only to a state, not to any run. Thus, according to Remark 5 the \( \xi \)-interval \( \zeta \) wherein \( \tilde{G} \) fires lies to the left of \( t_0 + \Lambda^* + 2\xi \):

\[
\tilde{t}_G < \zeta^{(r)} < t_0 + \Lambda^* + 2\xi, \tag{56}
\]

where \( \tilde{t}_G \) denotes the instant whereat \( \tilde{G} \) fires in \( \zeta \) in \( \rho_1 \).

Let \( \zeta_1 \) be the first \( \xi \)-interval after \( t^* \) wherein tilde-guards of \( \mathcal{A}_1 \) fire after \( t^* \); notice that they fire in \( \zeta_1 \) with the values of internal functions from \( \rho_1(t^*) \). More precisely, this \( \xi \)-interval is defined as follows depending on the case: DownUp or UpDown.

In the case DownUp this \( \xi \)-interval is determined by \( \rho_1 \), and the just given description is complete.

In the case UpDown we partition the time after \( t^* \) into \( \xi \)-intervals in some way, for example, into equal length interval of size \( \xi \). Within the run \( \rho_1 \) that is defined up to \( t^* \), the first moment whereat some guard becomes true with the values of its functions at \( t^* \) is defined uniquely, and depends only on these values at \( t^* \) and the values of inputs. As for the instant whereat this or that guard is evaluated or whereat it fires, we choose it in some fixed way in an appropriate \( \xi \)-interval, taking into account that the first instant whereat the guard become true, is defined as we have just mentioned, and that instants whereat a guard is evaluated to true or whereat it fires, should be inside the \( \xi \)-interval.

Let \( G_1 \) be the set of all guards of \( \mathcal{A}_0 \) whose tilde-image fires in \( \zeta_1 \) (with the values of internal functions from \( \rho_1(t^*) \)).

As we have just mentioned above, for DownUp \( \rho_1 \) the guards \( \tilde{G} \), where \( G \in G_1 \), fire in \( \rho_1 \). For UpDown case we extend UpDown \( \rho_1 \) by firing the appropriate guards of \( \mathcal{A}_1 \) up to \( \zeta_1^{(r)} \).

**Claim 3.** For any \( G \in G_1 \)

\[
t_0 - \Lambda^* - 2\xi < \tilde{t}_G < \zeta_1^{(r)} \leq t_0 + \Lambda^* + 2\xi, \tag{57}
\]

where \( \tilde{t}_G \) denotes the instant whereat \( \tilde{G} \) fires in \( \zeta_1 \) in \( \rho_1 \).

**Proof.** Let \( G \in G_1 \). The upper bounds on \( \tilde{t}_G \) and \( \zeta_1^{(r)} \) in (57) follow from (56) and from the choice of \( \zeta_1 \) as the first \( \xi \)-interval after \( t^* \) whereat tilde-guards fire in \( \rho_1 \) (any tilde-image of a guard \( G \in G_1 \) fires not later than \( (t_0 + \Lambda^* + 2\xi) \)). So it remains to prove the lower bound from (57).

We wish to apply Lemma 8. The premise of this lemma demands the run \( \rho_1 \) to be defined up to \( (t_0 - \Lambda^*_G - 2\xi) \). For the case DownUp \( \rho_1 \) is defined everywhere. For the case UpDown, if \( \rho_1 \) is not yet defined till this instant, we take an arbitrary extension that corresponds to an execution of \( \mathcal{A}_1 \). Lemma 8 says that \( \tilde{G} \) cannot be true in \([t^*, t_0 - \Lambda^*_G - 2\xi]\) in \( \rho_1 \), whatever be the extension of \( \rho_1 \) that we consider.

Therefore in both cases DownUp and UpDown, \( \tilde{G} \) cannot fire before or at \((t_0 - \Lambda^*_G - 2\xi) \).
Claim 4. $G_1 \subseteq G$.

Proof. Suppose that $G_1 \not\subseteq G$, and thus, $(G_1 \setminus G) \neq \emptyset$.

Take any $G \in (G_1 \setminus G)$. All the guards from $G_1$ are evaluated in $\rho_1$ with the values of internal functions from $\rho_1(t^*)$.

Apply Assumption 8(AR3) to $G$ with $C = \emptyset$ and $t = t_0$. In this case $G \in G^* = (Grds \setminus G)$. According to this Assumption 8(AR3) this guard $G$ is false everywhere in $\Re(\emptyset, \rho_0, t_0, \varepsilon, \eta)$.

(Here and below we do not need the neighborhood $\Re(\emptyset, \rho_0, t_0, 2\varepsilon, \eta)$ that was, however, used to prove Lemma 1.) In particular, due to Lemma 3 and (28), it is false with the values of internal functions from $\rho_1(t^*)$.

Hence,

(*) $G$ is false everywhere in $[t_0 - \varepsilon, t_0 + \varepsilon]$, in particular in $\zeta_1$, with the values of internal functions from $\rho_1(t^*)$, as well as the values of internal functions from $\rho_0(t^*)$.

We cannot deduce directly from this conclusion that $\tilde{G}$ is evaluated to false at the instant $t_{\tilde{G}}$ whereat it, as we supposed, fires in $\zeta_1$ in $\rho_1$ and, thus get a contradiction. The tilde-image of state $\rho_0(t_{\tilde{G}})$ and the tilde-part of state $\rho_1(t_{\tilde{G}})$ are sufficiently close in what concern the internal functions, but inputs may be different as, in $\Re(\emptyset, \rho_0, t_0, \varepsilon, \eta)$, all inputs are taken at the same instant as the time, and in $\rho_1(t_{\tilde{G}})$ the inputs are taken at different instants, and the time is also shifted. In order to overcome this difference, we apply Assumption 8(AR4).

Definition 32 (Common instant of inputs in $\zeta_1$). An instant $t$ is a common instant of inputs of $G$ in $\zeta_1$ if for each literal $I \in \text{InpG}$ there is a $\rho_0$-interval of $I$ containing $t$. $\Box$

In the proof of Claim 4 we use the following notations:

Notations 12.

- $T_{\tilde{G}}$ is the time instant backed up as the value of $\tilde{CT}$ in $\zeta_1$ with which $\tilde{G}$ is evaluated at $t_{\tilde{G}}$,
- $t_{\tilde{G}}$ is the instant whereat $\tilde{G}$ fires in $\zeta_1$ in $\rho_1$,
- $\sigma_{\text{inp}}$ is the intersection of $\zeta_1$ with the $\rho_0$-intervals containing the first common instant of inputs in $\zeta_1$, if such an instant exists, and $\emptyset$ otherwise.
- $\sigma^k_{\text{ineq}}$ is the intersection of all $(\rho_k, t^*)$-intervals of literals from $\text{IneqG}$, $k = 0, 1$. $\Box$

To continue the proof of Claim 4, consider several cases.

Case 1: $\exists t \in \zeta_1 (E(G, t, \emptyset) \text{ contains disjoint intervals})$. Let $T \in \zeta_1$ be such that $E(G, T, \emptyset)$ contains disjoint intervals.

Assumption 8(AR4) says that there are two $\rho_0$-intervals of literals of $G$ from $E(G, T, \emptyset)$ that are $2\varepsilon$-separated. They cannot be both input $\rho_0$-intervals as the input $\rho_0$-intervals are at most $\xi$-separated. Thus, one of them is a $(\rho_0, t^*)$-interval of an inequality. Whatever be the other one, input or inequality, one $(\rho_0, t^*)$-interval of an external inequality is $\varepsilon$-separated from $T$, and thus $(\varepsilon - \xi)$-separated from $\zeta_1$. Denote this $(\rho_0, t^*)$-interval by $\beta$ and the respective inequality from $\text{IneqG}$ by $H$. Denote by $X$ the finite end of $\beta$, and $X'$ the finite end of the $(\rho_1, t^*)$-interval of $H$.

Lemmas 3, 5 imply that $|X - X'| \leq 2\xi + \Lambda^*$. From this inequality, (28), (24) and $(\varepsilon - \xi)$-separation of $X$ from $\zeta_1$ (and thus, from the interval of backup in $\zeta_1$) we see that $H$ cannot be evaluated to true in $G$ in $\zeta_1$. Contradiction. Hence, Case 1 is impossible.

Case 2: $\forall t \in \zeta_1 (E(G, t, \emptyset) \text{ does not contain disjoint intervals})$. 
In this case either \( \text{Inp}G = \emptyset \), or there is an instant in \( \zeta_1 \) that belongs to the intersection of all \( \rho_0 \)-intervals of literals of \( \text{Inp}G \), i.e., \( \sigma_{\text{inp}} \neq \emptyset \). The latter follows from Lemma 9.

**Case 2.1: \( \text{Inp}G = \emptyset \).** The guard \( G \) cannot contain only internal literals because they have equal values in \( \rho_0(t^*) \) and \( \rho_1(t^*) \) due to Lemma 3. Thus, \( \tilde{G} \) should fire before \( t^* \) that contradicts to (57).

Thus, \( G \) contains an external inequality, and \( T_{\tilde{G}} \in \sigma_{\text{ineq}} \) as \( \tilde{G} \) is true at \( t_{\tilde{G}} \) in \( \rho_1 \). Indeed, \( \bar{CT}[\rho_1](t_{\tilde{G}}) = T_{\tilde{G}} \). Let \( H \in \text{Ineq}G \). The values of internal real-valued functions in the evaluation of \( \bar{H}[\rho_1](t_{\tilde{G}}) \) and \( H[\rho_1](T_{\tilde{G}}) \) are the same as they are taken from \( \rho_1(t^*) \). Hence, \( \bar{H}[\rho_1](t_{\tilde{G}}) = H[\rho_1](T_{\tilde{G}}) \). This means that the \( (\rho_1, t^*) \)-intervals of \( G \) and \( \tilde{G} \) both contain \( T_{\tilde{G}} \), and thus, \( G \) is true at \( T_{\tilde{G}} \) in \( \rho_1 \) that contradicts to (\*)

**Case 2.2: \( \text{Inp}G \neq \emptyset \).** In this case \( \sigma_{\text{inp}} \neq \emptyset \), as it was mentioned above.

**Case 2.2a: \( \text{Ineq}G = \emptyset \).** In this case \( G \) is true in \( \sigma_{\text{inp}} \) with the values of its interval functions at \( t^* \) whereat the latter are true in both runs. This contradicts to (\*).

**Case 2.2b: \( \text{Ineq}G \neq \emptyset \).** If \( \sigma_{\text{ineq}}^0 \) intersects \( \sigma_{\text{inp}} \) then \( G \) is true in \( \zeta_1 \) with the values of its internal functions from \( \rho_0(t^*) \). This contradicts to (\*).

Hence, \( (\sigma_{\text{ineq}}^0 \cap \sigma_{\text{inp}}) = \emptyset \) (this does not exclude the case when \( \sigma_{\text{ineq}} = \emptyset \)). Then there exists \( T \in (\zeta_1 \setminus \sigma_{\text{ineq}}^0) \). In this case the set \( E(G, T, \emptyset) \) contains disjoint intervals that contradicts to Case 2.

**Claim 5.** \( \zeta_1 \) and \( G_1 \) satisfy properties (A)–(E).

**Proof.** Property (A1) is trivial.

Property (A2) is evident. Indeed, \( G_1 \neq \emptyset \) is proven above when we find that there is a guard that can be included in \( G_1 \). So the first two strict inclusions of (52) are proven. The last strict inclusion of (52) is irrelevant for the moment as we do not know yet whether \( m = 1 \) or \( m > 1 \) — this will be treated in the induction step of the induction (A)–(E) below.

As for the first intersection of (51), it is trivial as there are yet no 2 sets \( G_i \), \( i \geq 1 \) to intersect. Consider the next intersection formula from (51). It follows straightforwardly from the definitions:

\[ \tilde{G}_1 \cap G_i^1 = G_1 \cap G_i^1 = \emptyset \] as \( G_i^1 \subseteq (G \setminus G_i) \).

and from Assumption 8(AR2)(a1) that says that all updates fired by guards from \( G_i^1 \) are done by firing guards from \( G_i \).

Property (A3) follows from Claim 4 and from our construction.

Formula (53) of property (B) is proved in Claim 3.

Consider property (C). Let a real-valued function \( f \) be updated by firing a tilde-image of a guard \( G \in \tilde{G}_1 = G_i \) in \( \zeta_1 \) in \( \rho_1 \). We wish to apply Lemma 6 and Lemma 7.

First, to apply Lemma 6, we take the bound on time instants used in the two updates under consideration from the proven (B): \( \Delta = \Lambda^* + 2\xi = Kp\pi^* + 2\xi \). Next, from Lemma 3 we take \( \Gamma = \pi^* \). This gives the upper bound \( (K\Delta + pK\Gamma) \) of the difference of values of \( f \) after the updates in \( \rho_0 \) and \( \rho_1 \).

Now Lemma 7 gives \( (K\Delta + pK\Gamma) \leq \pi(\Sigma\pi(S_0) + 1) \). The property (C) demands \( \pi_1 \) as the upper bound. But \( \pi_1 = \pi(\Sigma\pi(S_1)) \), and \( \Sigma\pi(S_1) = \Sigma\pi(S_0) + |G_1| \). Thus, from the monotonicity of \( \pi \) on \( i \) we get (C) for \( f \in RF \) that are updated in \( \zeta_1 \).

Property (D) for a Boolean-valued function that is updated in \( \zeta_1 \) follows from the observation
that it is updated by a Boolean-valued function from \( \rho_1(t^*) \) wherein it has the same value as in \( \rho_0(t^*) \).

Now let a real-valued or Boolean-valued \( f \) be updated in \( \rho_0 \) at \( t_0 \) by a guard from \( G_k^\downarrow \). In this case it is locked in \( \zeta_1 \) due to the monotonicity of locking stated in Assumption 8(AR2)(a2).

Assumption 8(AR2)(a1) applied to \( t = t_0 \) and \( \mathcal{C} = G_1 \) says that
\[
(\rho_0 \downarrow \mathcal{F}_1) = (\rho_0 \downarrow \mathcal{F}(G_1 \cup G^\downarrow_1)) = (\rho_0 \downarrow \mathcal{F}_1).
\]

In other words, \( f \) is properly updated by firing guards of \( G_1 \), and from (C) and (D) for such updates proven just above, we get (C) and (D) for \( f \).

Property (E) follows from Claim 4 and the construction of \( \zeta_1 \) and \( G_1 \).

**Induction step for properties (A)–(E).**

Suppose that there is constructed a sequence of \( \xi \)-intervals \( \zeta_1, \ldots, \zeta_k \) and a sequence of non empty subsets \( G_1, \ldots, G_k \) of \( G \) that satisfy (A)–(E), where \( m \) is replaced by \( k \), and \( \mathcal{F}_m = G \) is replaced by \( \mathcal{F}_k \subseteq G, k \geq 1 \).

If \( \mathcal{F}_k = G \) then we set \( m = k \), and we are done.

Suppose that \( \mathcal{F}_k \not\subseteq G \).

**Claim 6.** \( \mathcal{F}_k^\downarrow = \emptyset \).

**Proof.** Suppose \( G \in \mathcal{F}_k^\downarrow \). Let \( \Psi = G \) is locked by \((\rho_0 \downarrow t_0 \mathcal{F}_k)\). Using the definitions of the involved sets, we have
\[
G \in \mathcal{F}_k^\downarrow \iff (G \in (G \setminus (G \setminus G_k^\downarrow)) \wedge \Psi) \iff
(G \in G \wedge G \notin G_k^\downarrow \wedge G \notin G_k^\downarrow \wedge \Psi) \iff \quad (58)
\]

\[
(G \in G_k \setminus \Psi) \wedge (G \notin G_k^\downarrow \wedge \Psi) \iff (G \in G \setminus G_k \wedge (G \notin G_k^\downarrow \wedge G \notin G_k^\downarrow \wedge \Psi)) \iff
(G \in G \setminus G_k \wedge \bigwedge_{i=1}^k (G \notin G_k^\downarrow \wedge G \notin G_k^\downarrow \wedge \Psi)) \iff \quad (59)
\]

From (59) for \( i = k \) we get
\[
(G \in G_k \setminus \Psi) \wedge (G \notin G_k^\downarrow \wedge (G \in G_k^\downarrow \wedge G \in G_k^\downarrow \wedge \Psi)) \quad (60)
\]

Consider two cases determined by the disjunction from (60).

Case 1: \( G \notin G_k^\downarrow \). This implies that \( G \) is not locked by \((\rho_0 \downarrow t_0 \mathcal{F}_k)\). Formula \( \Psi \) says that \( G \) is locked by \((\rho_0 \downarrow t_0 \mathcal{F}_k)\). From the definition of \( G_k^\downarrow \), we see that \( G_k^\downarrow \subseteq G_k^\downarrow \). Taking into account that (60) implies \( G \in (G \setminus G_k \mathcal{F}_k) \), and that locking is monotone (see Assumption 8(AR2)(a2)), we get that \( G \) is locked by \((\rho_0 \downarrow t_0 \mathcal{F}_k) \). Hence, from Assumption 8(AR2)(a1) we conclude that \( G \) is locked by \((\rho_0 \downarrow t_0 \mathcal{F}_k) \) that contradicts to our case.

Case 2: \( G \notin G_k^\downarrow \). By definition, \( G_{k-1}^\downarrow \subseteq G_k^\downarrow \), and thus, \( G \notin G_k^\downarrow \) that contradicts to (58).

**Now we continue the proof of the induction step for properties (A)–(E).**

Claim 6 says that no guard from \( (G \setminus \mathcal{F}_k) \neq \emptyset \) is locked by \((\rho_0 \downarrow t_0 \mathcal{F}_k)\), and hence we can apply Assumption 8(AR1) to all guards from \( (G \setminus \mathcal{F}_k) \).
Take any $G \in (\mathcal{G} \setminus \mathcal{F}_k)$. Apply Assumption 8(AR1) to $G$, $t = t_0$, $\mathcal{C} = \mathcal{F}_k$ and $\delta = \pi_k$. There exists $T_G$ such that

$$t_0 - \Lambda_k \leq T_G \leq t_0 + \Lambda_k$$

and $G$ is true in all states from

$$\mathfrak{D}(\downarrow \mathcal{F}_k, \rho_0, t_0, [T_G, t_0 + \varepsilon], \mathfrak{R}((\rho_0 \downarrow t_0, \mathcal{F}_k)|_{[\varepsilon, \pi_k]}))$$.

Let $\sigma_G = \max\{\zeta_k^{(r)}, t_0 - \Lambda_k, t_0 + \varepsilon\}$. From (B) of the induction hypothesis (A)–(E) and (28) we see that the length of $\sigma_G$ is greater than $2\xi$, and the inputs of $G$ are true in this interval.

The values of internal functions in $\rho_1(\zeta_k^{(r)})$ for $\tau \in \sigma_G$ are sufficiently close to those of state $(\rho_0 \downarrow t_0, \mathcal{F}_k)$ to be in (62): for the real-valued functions it follows from (C) of the induction hypothesis (A)–(E) and the upper bound (26) on $\pi(s)$, for Boolean-valued functions it follows from (D) of the induction hypothesis (A)–(E).

Thus, $\hat{G}$ should fire somewhere in $(\sigma_G^{(l)}(\zeta_k^{(r)}) + 2\xi)$ in $\rho_1$, if there are no other tilde-guards that fire before it, but after $\zeta_k^{(r)}$. Hence, there are guards that fire after $\zeta_k^{(r)}$ and before $(\varepsilon - \xi)$.

Let $\zeta_{k+1}$ be the first $\xi$-interval after $\zeta_i^{(r)}$ whereat tilde-guards fire in $\rho_1$, and let $\mathcal{G}_{k+1}$ be the set of all the guards whose tilde-image fires in $\zeta_{k+1}$. We do not repeat the distinction between DownUp and UpDown cases, as they are treated in the same way as in the proof of the basis of the induction hypothesis (A)–(E) above.

Notice (this remark is not formally relevant to the present reasoning) that if the intervals $\zeta_i$ with $1 \leq i \leq k$ are ‘very close’ to $t_0$ (‘much closer’ than given by (B) of the induction hypothesis (A)–(E)) then $T_G$ can be ‘far’ to the right from $\zeta_k^{(r)}$, and the interval $\zeta_{k-1}$ is not adjacent to $\zeta_k$. On the other hand, if $\zeta_k$ is at the limit of its bound (B), then $\zeta_{k+1}$ may be adjacent to $\zeta_k$.

From the upper bound of (61) we see that for any $G \in \mathcal{G}_{k+1}$ the instant $t_G$ whereat it fires in $\zeta_{k+1}$ is bounded by the same bound with added $2\xi$ (see Remark 5): \hspace{1cm}

$$t_G < \zeta_{k+1}^{(r)} < t_0 + \Lambda_k + 2\xi.$$

As $\zeta_{k+1}$ lies to the right of $\zeta_k$, from (B) of the induction hypothesis (A)–(E) and monotonicity of $\Lambda_i$ on $i$ (see Definition 31) we have

$$t_0 - \Lambda_k - 2\xi \leq t_0 - \Lambda_{k-1} - 2\xi < \zeta_k^{(r)} \leq \zeta_{k+1}^{(r)} < t_G < \zeta_{k+1}^{(r)}.$$

**Claim 7.** $\mathcal{G}_{k+1} \subseteq \mathcal{G}$.

**Proof.** Suppose that $\mathcal{G}_{k+1} \not\subseteq \mathcal{G}$, and thus, $(\mathcal{G}_{k+1} \setminus \mathcal{G}) \neq \emptyset$. Notice that we wish to exclude two cases: either a tilde-image of a guard from $\hat{G}$ fires again or a tilde-image of a guard not from $\mathcal{G}$ fires in $\zeta_{k+1}$. We follow very closely the proof of Claim 4.

Take any $G \in (\mathcal{G}_{k+1} \setminus \mathcal{G})$. Then $G \in (\mathfrak{Grds} \setminus \mathcal{G})$. All the tilde-images of guards from $\mathcal{G}_{k+1}$ are evaluated in $\rho_1$ in $\zeta_{k+1}$ with the values of internal functions from $\rho_1(\zeta_k^{(r)})$. The functions updated in $\rho_1(\zeta_k^{(r)})$ after $t^*$ are the same as the functions in state $(\rho_0 \downarrow t_0, \mathcal{F}_k)$ updated in $\rho_0$ as compared with their values at $t^*$.

Apply Assumption 8(AR3) to $G$ with $\mathcal{C} = \mathcal{F}_k$ and $t = t_0$. In this case $G \in \mathcal{G}^* = (\mathfrak{Grds} \setminus (\mathcal{G} \setminus \mathcal{F}_k))$. According to this Assumption 8(AR3) this guard $G$ is false everywhere.
of all $(T, \rho_0, t_0, \varepsilon, \eta)$. The values of internal functions of $G$ from $\rho_1(\zeta_k^{(r)})$, as well as the values of internal functions from $(\rho_0 \downarrow t_0, F_k)$, are in $\mathfrak{N}(\rho, F_k, \rho_0, t_0, \varepsilon, \eta)$, as follows from (C) and (D) of the induction hypothesis (A)–(E).

From (63), (64) and (28) we see that $\zeta_{k+1}$ is inside $[t_0 - \varepsilon, t_0 + \varepsilon]$.

Hence, 

(\ast\ast) $G$ is false everywhere in $[t_0 - \varepsilon, t_0 + \varepsilon]$, in particular in $\zeta_{k+1}$, with the values of internal functions from $\rho_1(\zeta_k^{(r)})$, as well as from $(\rho_0 \downarrow t_0, F_k)$.

**Definition 33 (Common instant of inputs in $\zeta_{k+1}$).** An instant $t$ is a common instant of inputs of $G$ in $\zeta_{k+1}$ if for each literal $I \in \text{InpG}$ there is a $\rho_0$-interval of $I$ containing $t$. □

In the proof of Claim 7 we use the following notations:

**Notations 13.**

- $T_G$ is the time instant backed up as the value of $\tilde{C}T$ in $\zeta_{k+1}$ with which $\tilde{G}$ is evaluated at $t_{G}$.
- $t_{G}$ is the instant whereat $\tilde{G}$ fires in $\zeta_{k+1}$ in $\rho_1$.
- $\sigma_{\text{inp}}$ is the intersection of $\zeta_{k+1}$ with the $\rho_0$-intervals containing the first common instant of inputs in $\zeta_{k+1}$, if such an instant exists, and $\emptyset$ otherwise.
- $\sigma_{\text{ineq}}^1$ is the intersection of all $(\rho_1, \zeta_k^{(r)})$-intervals of literals from $\text{IneqG}$, and $\sigma_{\text{ineq}}^0$ is the intersection of all $S_k$-intervals of literals from $\text{IneqG}$ (it may be worth to recall Definition 27). □

**Case 1:** $\exists t \in \zeta_{k+1}$ ($\mathcal{E}(G, t, F_k)$ contains disjoint intervals). Let $T \in \zeta_{k+1}$ be such that the set $\mathcal{E}(G, T, F_k)$ contains disjoint intervals.

Assumption 8(AR4) says that there are two $\rho_0$-intervals of literals of $G$ from $\mathcal{E}(G, T, F_k)$ that are $2\varepsilon$-separated. They cannot be both input $\rho_0$-intervals as the input $\rho_0$-intervals are at most $\xi$-separated in the situation we consider. Thus, one of them is a $S_k$-interval of an inequality. Whatever be the other one, input or inequality, one $S_k$-interval of an inequality is $\varepsilon$-separated from $T$, and thus $(\varepsilon - \xi)$-separated from $\zeta_{k+1}$. Denote this $S_k$-interval by $\beta$ and the respective inequality from $\text{IneqG}$ by $H$. Denote by $X$ the finite end of $\beta$, and $X'$ the finite end of the $(\rho_1, \zeta_k^{(r)})$-interval of the inequality $H$, that we denote by $\beta'$.

According to (C) of the induction hypothesis (A)–(E) the real-valued functions used in the calculation of $\beta$ and $\beta'$ are $\pi_k$-close. Lemma 2 imply that $|X - X'| \leq 2\xi + \Lambda_k$. From this inequality, (28), (24) and $(\varepsilon - \xi)$-separation of $X$ from $\zeta_{k+1}$ (and thus, from the interval of backup in $\zeta_{k+1}$) we see that $\tilde{H}$ cannot be evaluated to true in $\tilde{G}$ in $\zeta_{k+1}$. Contradiction. Hence, Case 1 is impossible.

**Case 2:** $\forall t \in \zeta_{k+1}$ ($\mathcal{E}(G, t, F_k)$ does not contain disjoint intervals).

In this case either $\text{InpG} = \emptyset$, or there is an instant in $\zeta_{k+1}$ that belongs to the intersection of all $\rho_0$-intervals of literals of $\text{InpG}$, i.e., $\sigma_{\text{inp}} \neq \emptyset$. The latter follows from Lemma 9.

**Case 2.1:** $\text{InpG} = \emptyset$.

The guard $G$ cannot contain only internal literals. Indeed, suppose that $G$ contains only internal literals. According to (D) of the induction hypothesis (A)–(E), the values of internal Boolean predicates in $\rho_1(\zeta_k^{(r)})$ and in $S_k$ are the same, i.e., true, as they are backed up in $\zeta_{k+1}$ where $\tilde{G}$ is true with these values at some instant. As for internal inequalities, they are true due to (C) of the induction hypothesis (A)–(E), (32), (26) and $\eta$-stability of internal inequalities of $A_0$. 


But in this case, $G$ is true at $S_k$ that contradicts to (**) \\
Hence, $G$ contains an inequality, i.e., $\text{Ineq}_G \neq \emptyset$. Then $T_{\tilde{G}} \in \sigma_{\text{Ineq}}^1$ as $G$ is true at $t_{\tilde{G}}$ in $\rho_1$. Indeed, $\tilde{C}T_{\rho_1}(t_{\tilde{G}}) = T_{\tilde{G}}$. Let $H \in \text{Ineq}_G$. The values of internal real-valued functions in the evaluation of $\tilde{H}_{\rho_1}(t_{\tilde{G}})$ and $H_{\rho_1}(T_{\tilde{G}})$ are the same as they are taken from $\rho_1(\zeta_k^{(r)})$. Hence, $\tilde{H}_{\rho_1}(t_{\tilde{G}}) = H_{\rho_1}(T_{\tilde{G}})$. This means that the $(\rho_0, \zeta_k^{(r)})$-intervals of $G$ and $\tilde{G}$ both contain $T_{\tilde{G}}$; and thus $G$ is true at $T_{\tilde{G}}$ in $\rho_1$ that contradicts to (**) \\
**Case 2.2:** $\text{Inp}_G \neq \emptyset$. In this case $\sigma_{\text{Inp}} \neq \emptyset$, as it was mentioned above, after the statement of Case 2.

**Case 2.2a:** $\text{Ineq}_G = \emptyset$. In this case $G$ is true in $\sigma_{\text{Inp}}$ with the values of its internal functions at $\zeta_k^{(r)}$ whereat the latter are true in both runs. This contradicts to (**) \\
**Case 2.2b:** $\text{Ineq}_G \neq \emptyset$. If $\sigma_{\text{Ineq}}^0$ intersects $\sigma_{\text{Inp}}$ then $G$ is true in $\zeta_{k+1}$ with the values of its internal functions from $S_k$. This contradicts to (**) \\
Hence, $(\sigma_{\text{Ineq}}^0 \cap \sigma_{\text{Inp}}) = \emptyset$ (this does not exclude the case when $\sigma_{\text{Ineq}} = \emptyset$). Then there exists $T \in (\zeta_{k+1} \setminus \sigma_{\text{Ineq}}^0)$. The set $E(G, T, F_k)$ contains disjoint intervals that contradicts to Case 2. ■

**Claim 8.** $\zeta_{k+1}$ and $G_{k+1}$ satisfy (A)–(E). 

**Proof.** Properties (A1) and (A2) are direct consequences of our construction and the definitions of the involved sets.

Property (A3). The first sentence of this property is a direct consequence of our construction and Claim 7. As for the values of internal functions used in updates, we refer to Assumption 8(AR5). This property says that if a functions $f$ is updated by a guard from $G$ then it is not used in other updates fired by guards from $G$. Thus, only non updated functions are used in these updates. And the latter are taken with their values at $t^*$. 

Property (B) is ensured by (63) and (64).

Property (C). Let $f$ be a real-valued function updated in $\zeta_{k+1}$. We use Lemma 6. We know from (A3) proven above that $f$ is updated by a term that uses the values of internal functions from $\rho_1(t^*)$. Hence, in terms of Lemma 6, we can take $\Gamma = \pi^* = \pi_0$ due to Lemma 3 and $\Delta = \Lambda_k + 2\xi$ due to proven (B). As $\pi_0 \leq \Lambda_k$, we can apply Lemma 7. The latter gives a bound $\pi(\Sigma_e(S_k) + 1)$ that is not greater than $\pi_{k+1}$.

Property (D). Due to Assumption 8(AR5) each Boolean-valued function is updated by a Boolean-valued function from $\rho_1(t^*)$, and the latter are the same in both runs.

For locked functions we use Assumption 8(AR2), as the case $k = 1$.

Property (E) follows from our construction and Claim 7.

**Finalizing the Proof of the Main Theorem.** Let $T^* = t_0 + \varepsilon$. This instant has the properties (IH1)–(IH3). Due to Lemma 1 there are no updates in run $\rho_0$ in $[T^* - \varepsilon, T^* + \varepsilon]$. Property (B) and (28) imply that there are no updates in $[T^* - 2\xi, T^*]$. Thus, we have (IH3) for $T^*$. The properties (IH1) and (IH2) follows from (A)–(E). This gives Theorem 1. ■

6 Conclusion

We have described some problems that arise when implementing an ‘absolutely’ synchronous ASM with real-time constraints by a slightly desynchronized ASM. We give sufficient conditions
that ensure that such an implementation is possible. The ASMs that we consider are not general. So one further step to accomplish is to generalize such sufficient conditions to more general ASMs. Another generalization that looks useful, is to formulate sufficient conditions under which a \((\varepsilon, \eta)-desynchronized\) general ASM permits a \((\varepsilon', \eta')-desynchronization\) with \((\varepsilon', \eta') > (\varepsilon, \eta)\) that gives an approximately bisimilar set of runs.

However, it seems more important to simplify our sufficient conditions, maybe weakening them, and to find some sufficient syntactic conditions that give similar result. At the beginning of study of this kind of problems it is too optimistic to hope to find interesting sufficient syntactic conditions.

It is clear that an \((\varepsilon, \eta)\)-implementation \(A_1\) of an IA-ASM \(A_0\) (here we speak about arbitrary implementations), in the general case, does not verify the requirements for the initial machine \(A_0\). To ensure given requirements for \((\varepsilon, \eta)\)-implementations, we should have more strong ones for abstract specifications. It is another topic. And here syntactic transformations that bring us to such stronger requirements are more important that for the previous problem.

The problem looks feasible. We give here some observations confirming this point.

Denote by \(V^\circ\) a timed version of \(V\), i.e. the vocabulary constituted by functions \(f^\circ : T \rightarrow \mathbb{Z}\), where \(f : \rightarrow \mathbb{Z}\) is from \(V\).

Consider a property \(\Phi\) of the form \(\forall T \bigwedge \bigvee_i K_i(T)\), where \(T\) is a list of time variables, and \(K_i\) is a conjunction of literals of the timed version \(V^\circ\) of the vocabulary \(V\) and of inequalities involving real-valued functions from \(V^\circ\). A typical safety property is of this form. For a given run \(\rho_0\) of \(A_0\) and a given instant \(T_0\), a connected component of \(\{T : K_i(T)\}\) that contains \(T_0\) is a polyhedron. These polyhedra cover the space of \(T\)'s. If we take a run \(\rho_1\) of \(A_1\) the respective polyhedra can deflate by some value defined by \((\varepsilon, \eta)\) and other parameters, e.g., like in Notations 5. Thus, to preserve the property for any \((\varepsilon, \eta)\)-implementation such a deflation should give polyhedrons that again cover the whole space of \(T\)'s. So to preserve a property verified by \(A_0\) we can inflate the polyhedra. Sometimes this can be done by a syntactical transformation of the initial conjunctions \(K_i(T)\). The main difficulty comes from the real-valued functions. However, their influence is traceable.

Rewrite the property \(\Phi\) in more detail

\[
\Phi = \forall T \bigvee_i \left( \Lambda I_i(T) \land \Lambda O_i(T) \land \Pi_i(T) \right),
\]

(65)

where \(\Lambda I_i(T)\) is a conjunction consisting of input predicates, \(\Lambda O_i(T)\) is a conjunction consisting of output predicate, and \(\Pi_i(T)\) is a conjunction of all inequalities. Each \(\Pi_i(T)\) can be viewed as a polyhedron in a real space.

For each polyhedron \(\Pi_i(T)\) we introduce an inflated polyhedron in the following way \(\Pi_i^\eta(T)\).

Suppose that polyhedron \(\Pi_i(T)\) is a conjunction of inequalities of the form

\[
\sum_{i \in I} \alpha_i \varphi_i(t_i) \leq \beta,
\]

(66)

where each \(\varphi_i\) is either an abstract real-valued function (suppose for indices \(i \in I_R\)) or identity function (for \(i \in I_{id}\)). Denote this inequality by \(Eq(T)\).

We replace this inequality by inequality \(Eq^\eta(T)\)

\[
\sum_{i \in I} \alpha_i \varphi_i(t_i) \leq \beta + \left( \sum_{i \in I_R} |\alpha_i| \right) \eta.
\]

(67)
Now define $\Pi_i^\eta(T)$ as conjunction of inequalities $\text{Eq}^\eta(T)$ obtained from inequalities $\text{Eq}(T)$ of $\Pi_i(T)$.

For each $t \in T$ and each expression $f(t)$ that occurs in $\Phi$, where $f$ is an internal function (or predicate), we introduce a time variable $t_f$ (this notation refers to $t$ and to $f$). Another notations that we need are:

- $E_i = \{ f : f(t) \text{ occurs in } \Phi \}$. The variables $t_f$ play a role of $\varepsilon$-perturbed time instants corresponding to $f$ and $t$ that is represented by $\bigwedge_{f \in E_i} |t - t_f| \leq \varepsilon$ in the formula (68) below.
- $T' = \{ (t_f)_{t \in T, f \in E_i} \}$ is a list of these new time variables $t_f$.

Notice that the same $t$ can be in different functions $f(t), g(t), \ldots$ and the same $f$ may occur with different $t, t', \ldots; f(t), f'(t), \ldots$.

Consider the following formula $\Phi^{\varepsilon,\eta}$ that represents a $(\varepsilon, \eta)$-inflated requirement $\Phi$:

$$\Phi^{\varepsilon,\eta} = \forall T \bigcup_i \left( \Lambda I_i(T) \land \exists T' \bigwedge_{t \in T} \bigwedge_{f \in E_i} (|t - t_f| \leq \varepsilon \land \Lambda O_i(T') \land \Pi_i^\eta(T, T')) \right),$$

where $\Lambda O_i(T')$ is the result of replacement of each $t$ in $f(t)$ by $f(t_f)$, $\Pi_i^\eta(T, T')$ is obtained from polyhedron $\Pi_i^\eta(T)$ by replacing each occurrence of $t_i$ in $\varphi_i(t_i)$ for $i \in I_R$ by $t_i, \varphi_i$ (this notation is of the form $t_f$ above with $t = t_i$ and $f = \varphi_i$). Notice that all $\alpha_j \varphi_j(t_j)$ with $j \in I_{id}$, i.e., $\alpha_j t_j$ remain unchanged.

**Proposition 1.** If $A_1$ is an $(\varepsilon, \eta)$-implementation of $A_0$ then $(A_0 \models \Phi) \Rightarrow (A_1 \models \Phi^{\varepsilon,\eta})$.

**Proof.** Take any run $\rho_1$ of $A_1$ and any list of time instants $T$. By Definition 18 of $(\varepsilon, \eta)$-implementation the run $\rho_0$ of $A_0$ which is determined by a given input is $(\varepsilon, \eta)$-close to $\rho_1$. This run $\rho_0$ verifies $\Phi$. For $T$, there exists a conjunction $K$ in $\Phi$ that is satisfied by $\rho_0(T)$.

As the input predicates are the same in both runs, the run $\rho_1$ necessarily satisfies $\Lambda I_i(T)$. For each $t \in T$ and for each $f \in E_i$, there exists $t_f \in (t - \varepsilon, t + \varepsilon)$ such that $\rho_0[f](t) = \rho_1[f](t_f)$ or $|\rho_0[f](t) - \rho_1[f](t_f)| < \eta$ because $\rho_0$ and $\rho_1$ are $(\varepsilon, \eta)$-close. Take as $T'$ the set of these $t_f$: $T' = \{ (t_f)_{t \in T, f \in E_i} \}$.

One can see that $\Phi^\eta$ is true. The only part that may be not quite evident concerns the polyhedron. We know that $\Pi_i(T)$ is true. Take any inequality that defines this polyhedron. Suppose it has the form (66), and we know that it is true. In $\Phi^\eta$ this equality is transformed into inequality (69) as it follows from (67)

$$\sum_{i \in I_R} \alpha_i \varphi_i(t_i, \varphi_i) + \sum_{i \in I_{id}} \alpha_i t_i \leq \beta + \left( \sum_{i \in I_R} |\alpha_i| \right) \eta,$$

(69)

For the run $\rho_0$, we have

$$\sum_{i \in I_R} \alpha_i \varphi_i[\rho_0](t_i) + \sum_{i \in I_{id}} \alpha_i t_i \leq \beta,$$

(70)

By definition of $(\varepsilon, \eta)$-implementation and according to the choice of $t_i, \varphi_i$,

$$|\varphi_i[\rho_1](t_i, \varphi_i) - \varphi_i[\rho_0](t_i)| < \eta$$

(71)

Therefore from (71) and (70) we get
\[
\sum_{i \in I_n} \alpha_i \varphi_i[\rho_1(t_i, \varphi_i)] + \sum_{i \in I_d} \alpha_i t_i \leq \beta + (\sum_{i \in I_n} |\alpha_i|) \eta.
\] (72)

Thus, \(\rho_1\) satisfies (69). \(\blacksquare\)

Proposition 1 shows what happens with a safety-like property when we go from IA-ASM to its implementation as DA-ASM. What one needs in reality is the opposite direction. We have a property that should be satisfied by an implementation. So we wish to transform it into a property to impose on our high level specification in terms of IA-ASM. The latter is easier to analyze. So if construct an IA-ASM that satisfies that transformed property we are sure that our implementation satisfies the initial property. The open question is to find such a transformation of practical value. However, it is worth to start with any one.

References


One ASM A1 is constructed to reflect faithfully the algorithm. Then a more abstract ASM A2 is constructed. It is checked that A2 is safe and fair, and that A1 correctly implements A2. The proofs work for atomic as well as, mutatis mutandis, for durative actions.


