From Scenarios To Optimally Allocated Timed Automata

Sandeep Vuppula

May 26, 2017

University of Minnesota Duluth

Table of contents

Objectives Of The Research

Our main focus of the research is,

1. To synthesize a timed automaton from a set of scenarios.
2. To optimally allocate clocks in the constructed timed automaton.

Introduction

Introduction

- Model-based design is a very effective method for designing real-time systems.
- Building formal models for systems is challenging because of the lack of good formal requirements specifications.
- In many real-time systems where safety is critical.
- Modeling a system formally can help us to understand the desired and undesired behaviours of the system.

Introduction

- To construct a formal model, the following questions are to be answered first:
  1. How the requirements should be expressed formally, and
  2. How the formal model of a real-time system can be constructed from requirements.

Introduction

- To construct a formal model, the following questions are to be answered first:
  1. How the requirements should be expressed formally, and
  2. How the formal model of a real-time system can be constructed from requirements.
- The formal model that we build is timed automata [1].

Introduction

- We use scenarios to build a formal model. A scenario is a partial description of the behaviour of a system.
- We introduce Timed Event Sequences (TES) to formally represent the scenarios formally.
- We use mode graphs to specify the legal events that can occur in the system.

Introduction

- We use scenarios to build a formal model. A scenario is a partial description of the behaviour of a system.
- We introduce Timed Event Sequences (TES) to formally represent the scenarios formally.
- We use mode graphs to specify the legal events that can occur in the system.

We synthesize a *minimal*, *acyclic* and *deterministic* timed automaton using TES and mode graph.

Introduction

Our timed automaton belongs to a class of timed automata that satisfies the following properties:

- A clock t_j can be reset only on the transitions emanating from a state labelled j and,
- A clock in a clock constraint on a transition r from a state q can only refer to a clock that has been reset on a transition leaving a state that dominates q. We call this *dominance assumption*.

Introduction

- Given a timed automaton $\mathcal{A}$, the problem of deciding whether there exists another timed automaton $\mathcal{B}$ that accepts the same language as that of $\mathcal{A}$ but with fewer number of clocks is undecidable.
- The number of clocks in a given timed automaton has a direct impact on verification of the system.

Introduction

- Given a timed automaton $\mathcal{A}$, the problem of deciding whether there exists another timed automaton $\mathcal{B}$ that accepts the same language as that of $\mathcal{A}$ but with fewer number of clocks is undecidable.
- The number of clocks in a given timed automaton has a direct impact on verification of the system.

Given a timed automaton that belongs to our class, we use *liveness analysis* of clocks to optimally allocate clocks.

Background

Real-time systems

- A real-time system takes input from its surrounding environment and produces results within a stipulated amount of time.
- In the real world, the behaviour of almost every system changes according to time.
- We can model such real time systems with the help of timed automata.

Modeling Time

There are three approaches for modeling time [1].

- Discrete time model: Time is considered as discrete and monotonically increasing sequence of integers. Limits the preciseness: in real-time systems, the events do not occur at integer times.

Modeling Time

There are three approaches for modeling time [1].

- Discrete time model: Time is considered as discrete and monotonically increasing sequence of integers. Limits the preciseness: in real-time systems, the events do not occur at integer times.
- Fictitious-clock model: It is similar to that of discrete time model except that it assumes sequence of times to be non decreasing integers. Limits accuracy, as the exact time values at which the events occur are not considered.

Modeling Time

There are three approaches for modeling time [1].

- Discrete time model: Time is considered as discrete and monotonically increasing sequence of integers. Limits the preciseness: in real-time systems, the events do not occur at integer times.
- Fictitious-clock model: It is similar to that of discrete time model except that it assumes sequence of times to be non decreasing integers. Limits accuracy, as the exact time values at which the events occur are not considered.
- Dense time model: In this model, the domain of time is considered as a dense set and the time of occurrences of events as real numbers, which increase monotonically without any limit. Difficulty in transforming dense time traces into formal languages.

Finite State Automata

A finite state automaton (FSA) or a finite state machine (FSM) is an abstract machine which has a finite number of states. On an input, the machine changes from one state to another state.

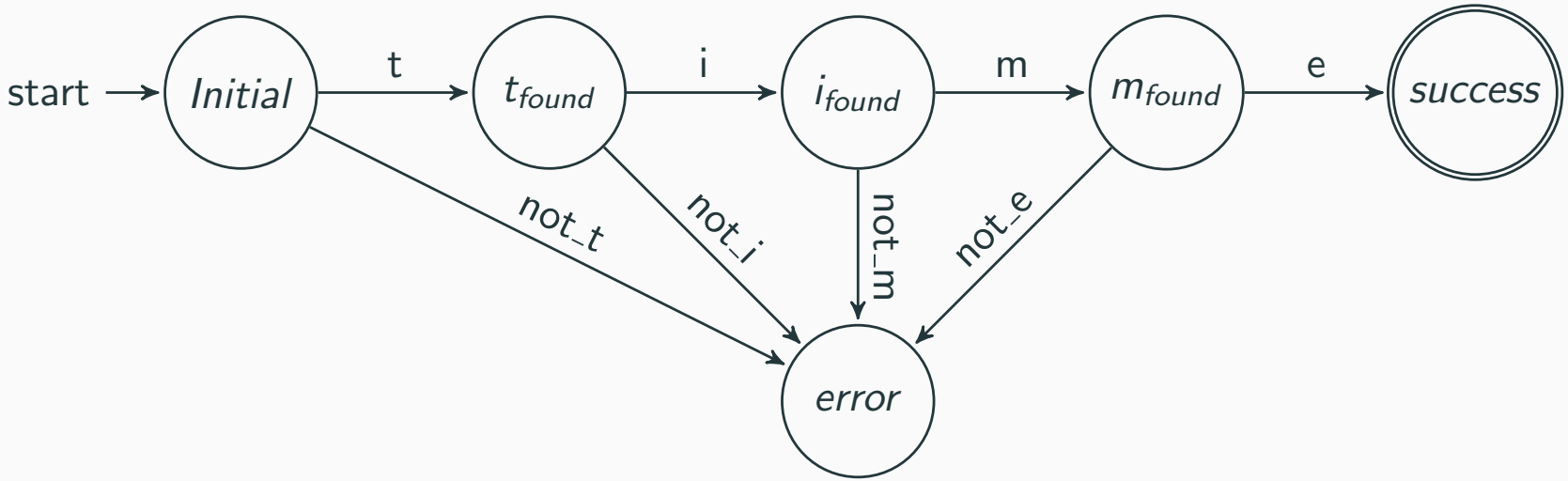


Figure 1: Simple FSM parsing the string "time"

Timed Automata

- A timed automaton [1] is a finite state automaton extended with a finite set of real-valued clocks.
- Upon an input, the selection of next state is based not only on the input symbol but also on the time of the current symbol with respect to the formerly read symbols.

Example: Consider a simple timed automaton in Figure 2. This automaton accepts an input sequence 'a' followed by 'b' such that, there is 2 units of time difference between any two consecutive a's and b's.

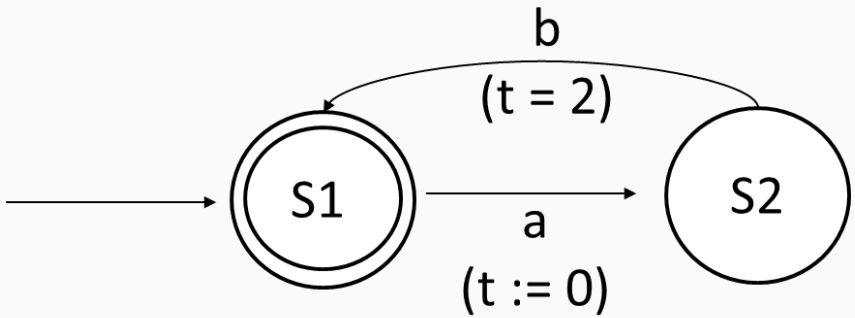


Figure 2: A simple timed automaton

Synthesis of Timed Automata from Scenarios

Synthesis of Timed Automata from Scenarios

Our method of synthesizing a timed automaton model of a real-time system from scenarios involves two steps:

1. Constructing a time annotated graph from scenarios, and
2. Constructing a timed automaton from time annotated graph.

Synthesis of Timed Automata from Scenarios

- Our approach for building a formal model is using scenarios.
  - A scenario is a partial description of the behaviour of a system.
  - A scenario not only describes the events, but also the timing relations among the events.
  - Set of scenarios together can capture the behaviour of the real-time system.
- We use Timed Event Sequences to describe the scenarios formally.
- We use Mode graphs to specify the legal events that can occur in the system.

Mode Graph

A *mode graph* is a deterministic state machine in which the states are called modes and the transitions triggered by the events in the system. It is a tuple $\mathcal{M} = (M, m_0, m_f, \Sigma, T)$ where,

- M is a finite set of modes,
- m_0 is the initial mode,
- m_f is the final mode,
- Σ is a set of events, and
- $T : M \times \Sigma \to M$ is a transition function.

Mode Graph Example

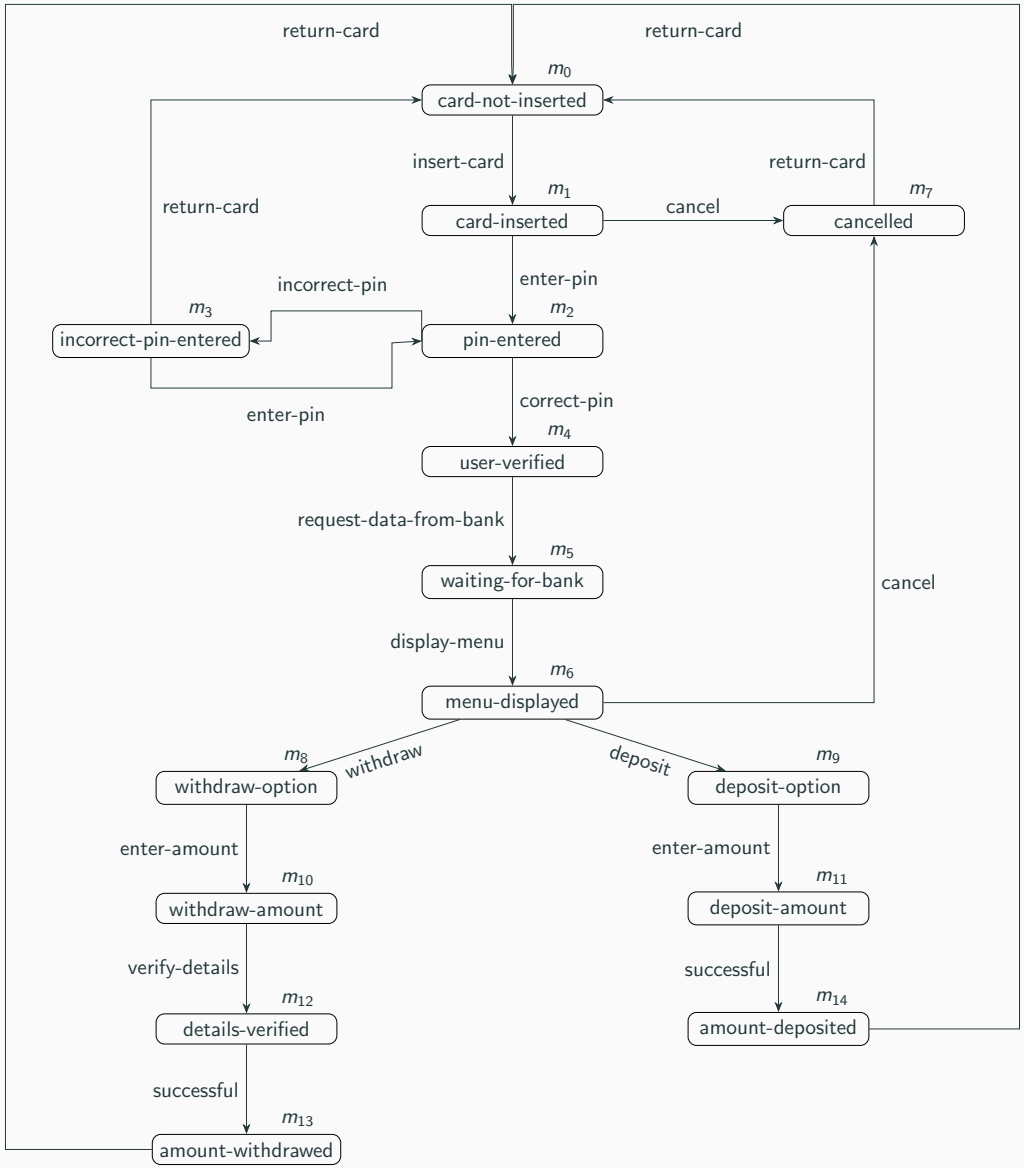


Figure 3: Mode graph of the ATM

Timed Event Sequences

The scenarios describing the partial behaviours of a real-time system are expressed formally in the form of Timed Event Sequences (TES).
A Timed Event Sequence ξ contains:

1. The initial mode of the scenario,
2. The final mode of the scenario,
3. A set of events and their corresponding time annotations.

$m^{initial}$: card-not-inserted

(insert-card, $\{\}$)
(enter-pin, $\{W - t_0 \geq 5,\ W - t_0 \leq 60\}$)
(incorrect-pin, $\{\}$)
(re-enter-pin, $\{W - t_0 \geq 5,\ W - t_0 \leq 60\}$)
(correct-pin, $\{\}$)
(request-data-from-bank, $\{\}$)
(display-menu, $\{W - t_4 \leq 5\}$)

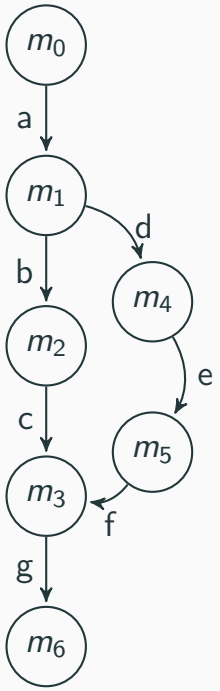
m^{final}: menu-displayed

TES of ATM scenario

Dominance Assumption

- Given two modes m_i and m_j, m_i is said to be the *dominating mode* of m_j iff all the paths to m_j from the initial mode in the mode graph pass through m_i [2]. We call this the *Dominance* relation and denote it as m_i *DOM* m_j.
- Dominance assumption ensures that time variables are well defined.

Dominance Assumption Example



$m^{initial}$: m_0
$(a,\ \{\})$
$(d,\ \{W - t_0 < 3\})$
$(e,\ \{W - t_1 > 1\})$
$(f,\ \{\})$
$(g,\ \{W - t_0 > 3, W - t_1 \leq 3\})$
m^{final} : m_6

Figure 4: A mode graph and a scenario satisfying the dominance assumption

- m_0 and m_1 are dominating modes of m_3,
- Transition g is dominated by all the modes that dominate m_3,
- t_0 and t_1, on transition g refer to the modes m_0 and m_1.

Constructing a Time Annotated Graph from Scenarios

Given a mode graph $\mathcal{M}$ and a set of Timed Event Sequences $\Xi = \{\xi_1, \xi_2, .., \xi_k\}$ as inputs, our algorithm synthesizes a time annotated graph (TAG) G [3]. Initially we start with an empty graph, G_0 and perform the following steps:

1. Build a partial graph G_1 using the first scenario ξ_1,
2. The algorithm repeatedly takes a partially built graph G_k, and a scenario ξ_{k+1} $(1 < k < n)$ and then generates a new partial graph G_{k+1}

Decision on whether to create new states and transitions is resolved with the help of state labels (modes). A new state s is created and labelled with a mode m_j if there is an event e from state q such that $L(q) = m_i$ and $(m_i, e, m_j) \in T$.

Properties of Constructed Time Annotated Graph

The graph constructed by algorithm has the following properties:

1. It is acyclic,
2. The graph is connected,
3. By construction, two states have the same label only if one is a predecessor of the other,
4. The graph is finite,
5. The graph is deterministic,
6. The graph is minimal,
7. After construction, every scenario is a partial run of the constructed graph, and
8. Every path in the final graph is identical to that of the mode graph.

Constructing a Timed Automaton from Time Annotated Graph

To convert a time annotated graph to a timed automaton we have to:

1. Determine the required number of clocks,
2. Add clock resets, and
3. Replace the time annotations with the clock constraints

Example: If there is a time annotation $W - t_0 > 5$ on a transition, then the clock constraint $c_0 > 5$ is added to that transition and clock c_0 is reset on all transitions from the state labelled with mode m_0.

Example

$m^{initial}$: card-not-inserted

(insert-card, $\{\}$)
(enter-pin, $\{W - t_0 \geq 5,\ W - t_0 \leq 60\}$)
(incorrect-pin, $\{\}$)
(re-enter-pin, $\{W - t_0 \geq 5,\ W - t_0 \leq 60\}$)
(correct-pin, $\{\}$)
(request-data-from-bank, $\{\}$)
(display-menu, $\{W - t_4 \leq 5\}$)

m^{final}: menu-displayed

TES of Scenario 1

$m^{initial}$: card-not-inserted

(insert-card, $\{\}$)
(enter-pin, $\{W - t_0 \geq 5,\ W - t_0 \leq 60\}$)
(correct-pin, $\{\}$)
(request-data-from-bank, $\{\}$)
(display-menu, $\{W - t_4 \leq 5\}$)

m^{final}: menu-displayed

TES of Scenario 2

Figure 5: Timed Event Sequences of the ATM

Example (Cont.)

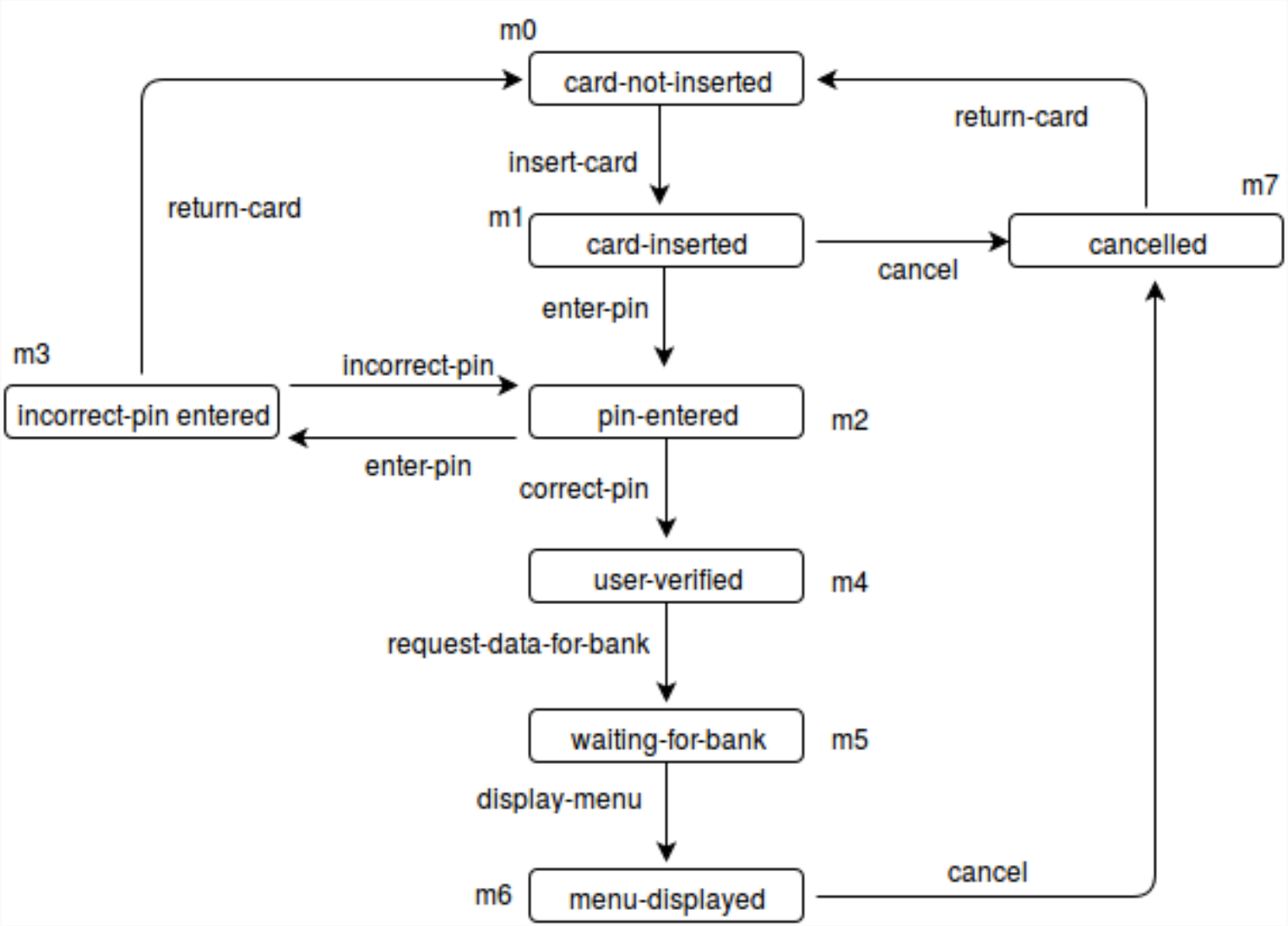


Figure 6: Mode Graph for ATM

Example (Cont.)

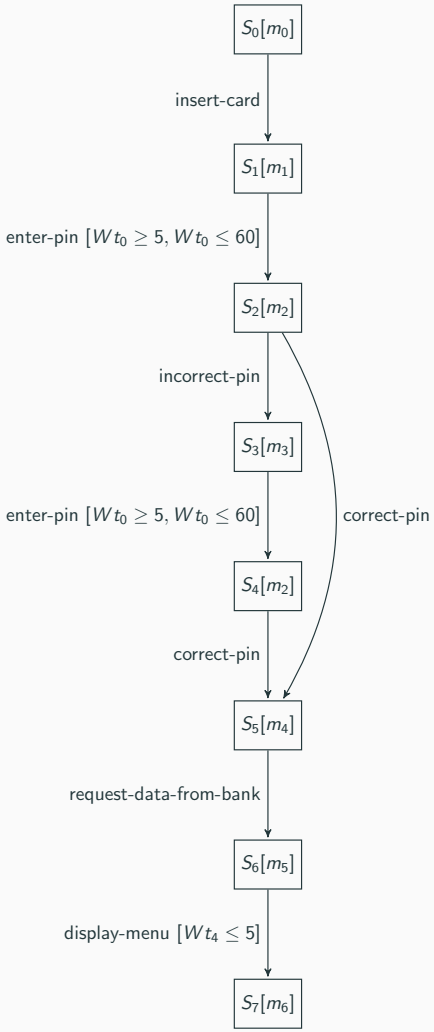


Figure 7: Time annotated graph synthesized from two TES in Figure 5

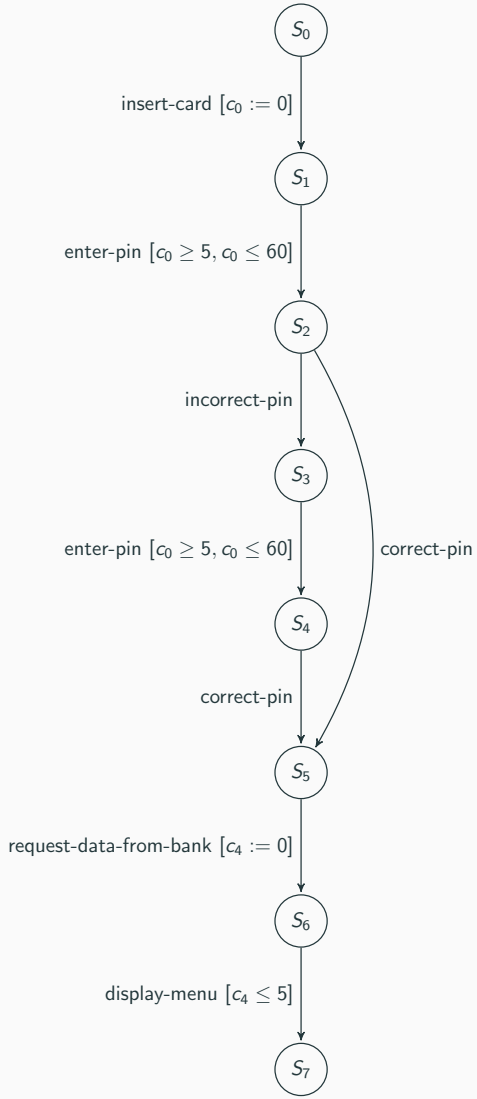


Figure 8: Timed automaton constructed from time annotated graph

Optimal Clock Allocation of Timed Automata

Class of Timed Automata

The timed automaton constructed as a result of our synthesis method belongs to the class of timed automata that satisfies these properties:

1. The automaton is connected and has a unique initial state s_0,
2. A clock constraint on a transition 'r' can only refer to the times of transitions from states that dominate the transition 'r', we call this *dominance assumption*,
3. A clock t_j can only be reset on a transition leaving a state s, where label is j, that is $L(s) = j$.

Optimal Clock Allocation of Timed Automata

Given a timed automaton $\mathcal{A}$ that belongs to the aforementioned class of timed automata, to transform it to an equivalent timed automaton $\mathcal{A}'$ with optimal allocation of clocks, we need to perform the following steps in order:

1. Calculate the liveness ranges of clocks in the timed automaton $\mathcal{A}$,
2. Replace the original clocks in $\mathcal{A}$ with a set of new clocks,
3. Rewrite the clock constraints and clock resets in $\mathcal{A}$ in terms of new clock variables.

Liveness Range Analysis

Liveness Ranges of all the clocks in a timed automaton helps us determine if a particular clock is only on these transitions.

- Let $\mathcal{A} = (E, Q, \{q^0\}, Q_f, V, R, L)$ be the timed automaton and $r = (s, s', e, \lambda_r, \phi_r) \in R$ be a transition.
- Let $N = \{j \mid t_j \sim a \in \phi \vee t_j \in \lambda$, where $(s, s', e, \lambda_r, \phi_r) \in R\}$, be a set of *clock numbers* used to denote subscripts of the clocks on all the transitions in R.

Liveness Range Analysis

The following are a set of functions used to calculate the liveness ranges:

- **clock_ref**: $clock\_ref(r)$ is the set of clocks which are referred to in the clock constraints on r.
- **born**: $born(r)$ identifies a clock that is reset on r whose value can be used on some transition reachable from r.
- **active**: $active(r)$ identifies clocks that are "alive" on r (i.e., their values may be subsequently used). Notice that $born(r) \subseteq active(r)$.
- **needed**: Maps transition r to $active(r) \cup clock\_ref(r)$.

Liveness Range Analysis Example

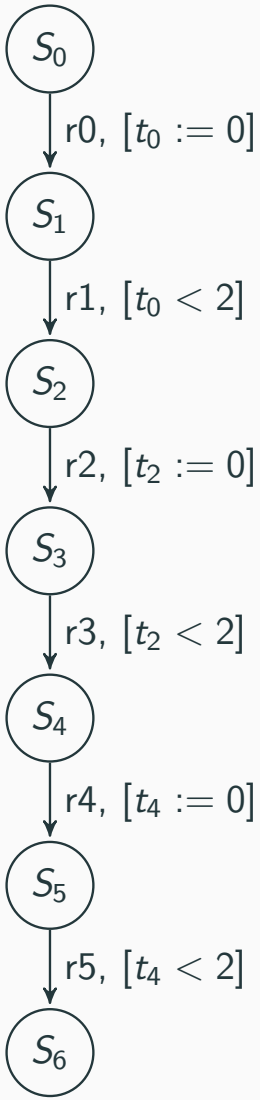


Figure 9: A simple timed automaton

Table 1: *born* and *active* values

Transition	Born	Active
r_0	$\{0\}$	$\{0\}$
r_1	ϕ	ϕ
r_2	$\{2\}$	$\{2\}$
r_3	ϕ	ϕ
r_4	$\{4\}$	$\{4\}$
r_5	ϕ	ϕ

Clock Allocation

The liveness analysis algorithm calculates liveness ranges and generates extended transitions of the form $(r, \mathit{born}(r), \mathit{active}(r))$. Our method to optimally allocate the clocks revolves around the idea that:

- A clock can be reused if the active range of the clock has ended,
- The clock cannot be reused on a transition if the transition belongs to active range of that clock.

Clock Allocation

Definition

Given a timed automaton $\mathcal{A}$ with the set R of (extended) transitions and the set N of clock numbers, a *clock allocation* for $\mathcal{A}$ is a relation $alloc \subset R \times P_0 \times N$ such that $(r, c, j) \in alloc \Rightarrow j \in active(r)$.

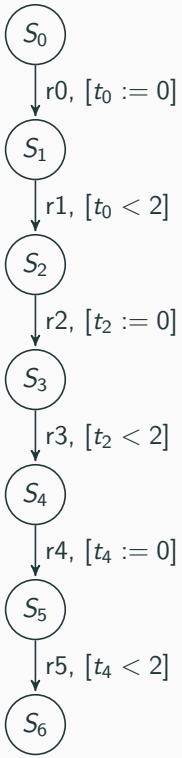


Figure 10: A simple timed automaton

$$
\begin{aligned}
alloc = \{&(r_0, c_0, 0), (r_1, c_0, 0), (r_2, c_0, 0), \\
&(r_3, c_0, 0), (r_4, c_0, 0), (r_5, c_0, 0), (r_6, c_0, 0), \\
&(r_1, c_1, 1), (r_2, c_1, 1), (r_3, c_1, 1), (r_4, c_1, 1)\}
\end{aligned}
$$

Problematic states

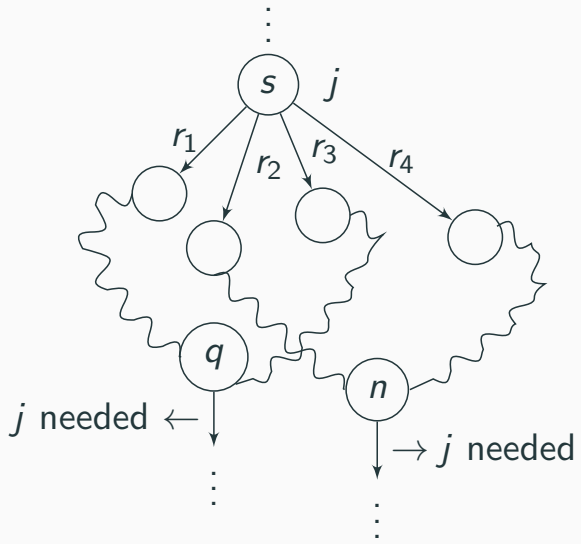


Figure 11: A timed automaton with problematic states

- Clock t_j is born on all outgoing transition of state s.
- r_1, r_3 meet at state q and r_2, r_4 meet at state n.

States q and n are the *problematic* states. So, r_1, r_3 should be assigned the same clock and r_2, r_4 should be assigned the same clock.

Mode Graph

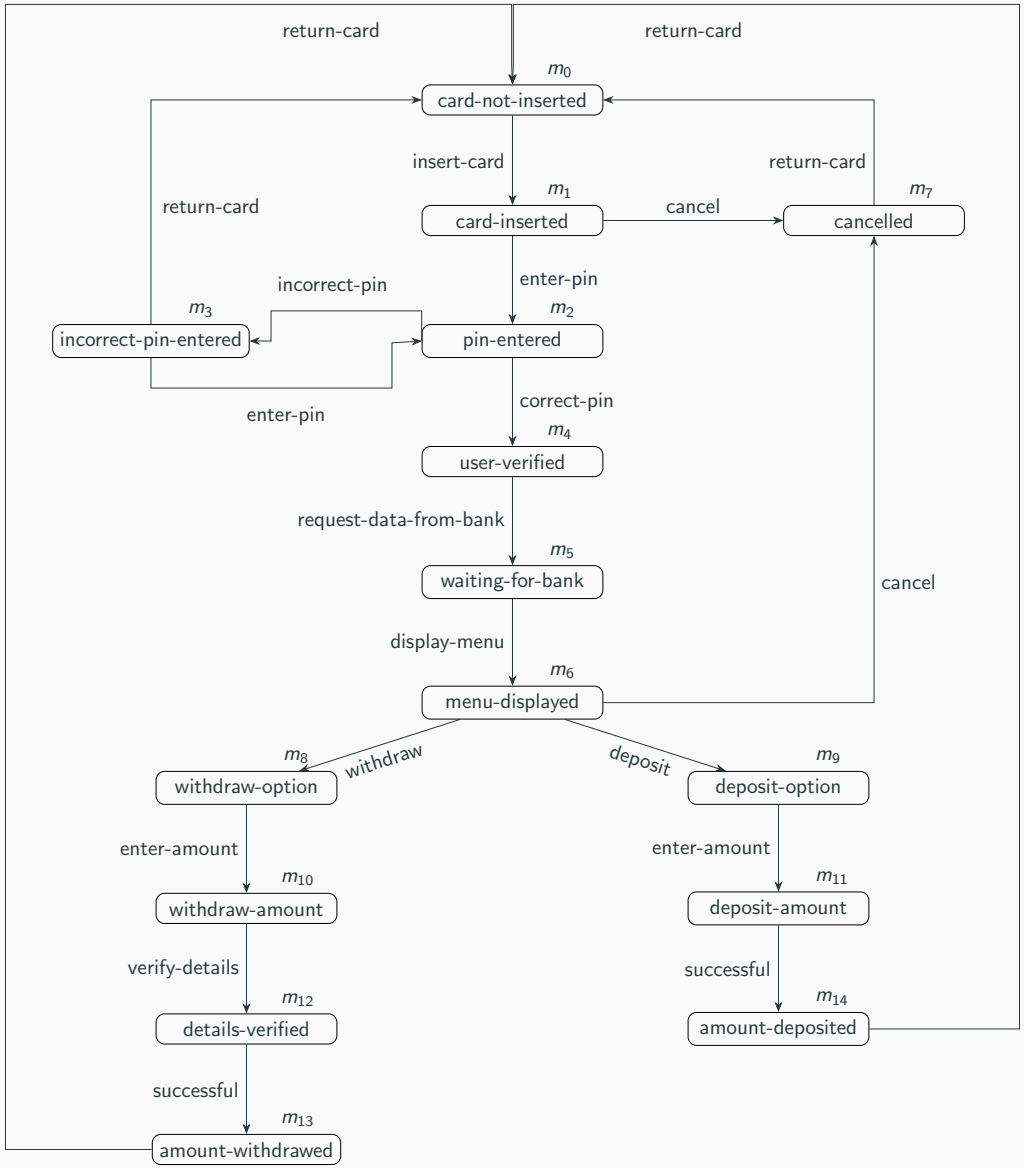


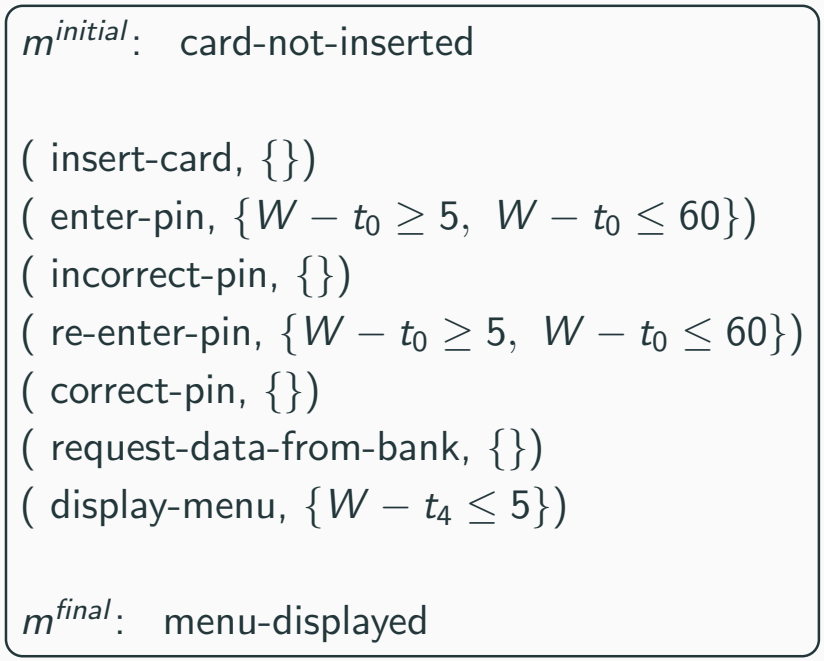
Figure 12: Mode graph of the ATM

Case Studies

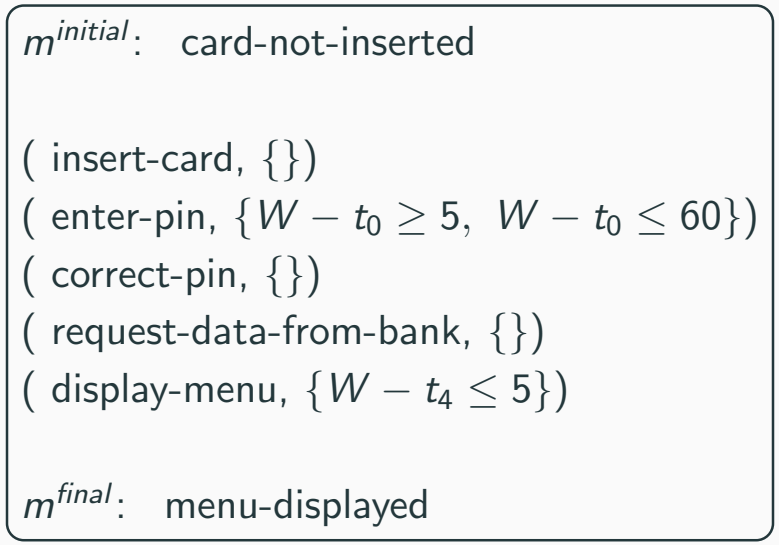
Automated Teller Machine (ATM)

Explain original scenario

Automated Teller Machine (ATM)



TES of Scenario 1



TES of Scenario 2

Figure 13: Timed Event Sequences of the ATM

Automated Teller Machine (ATM)

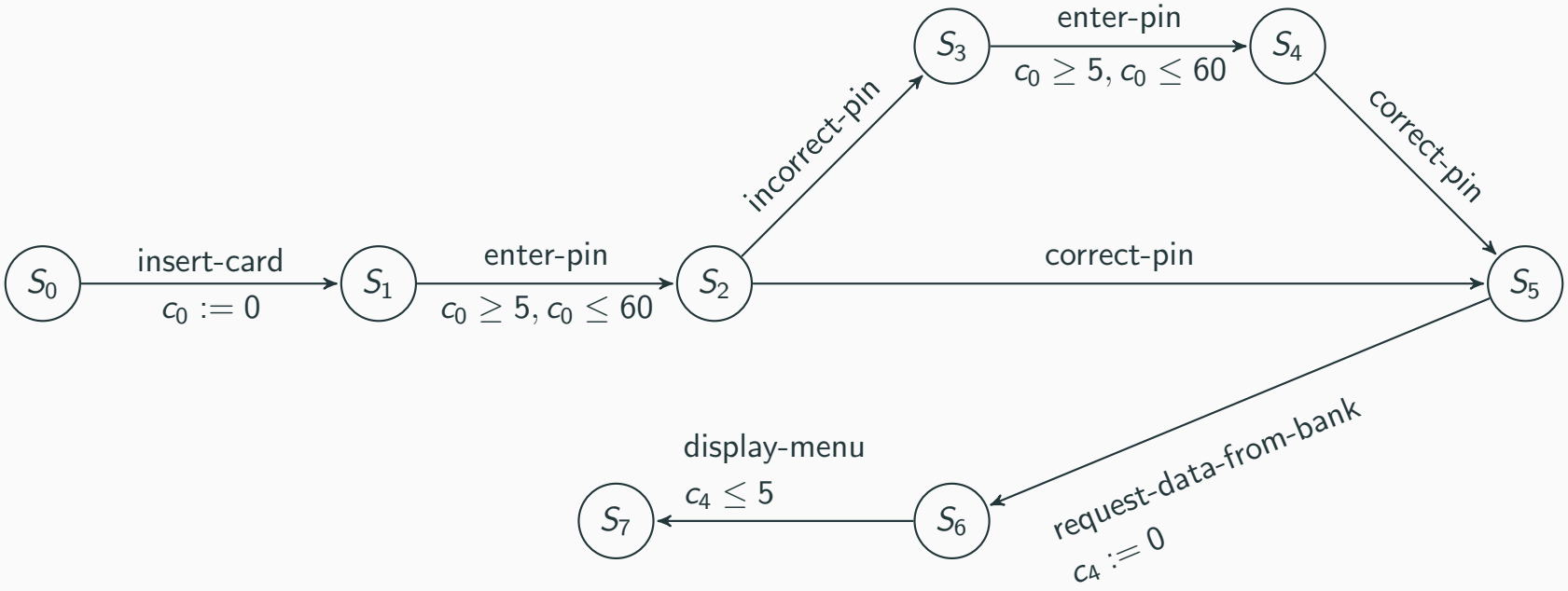
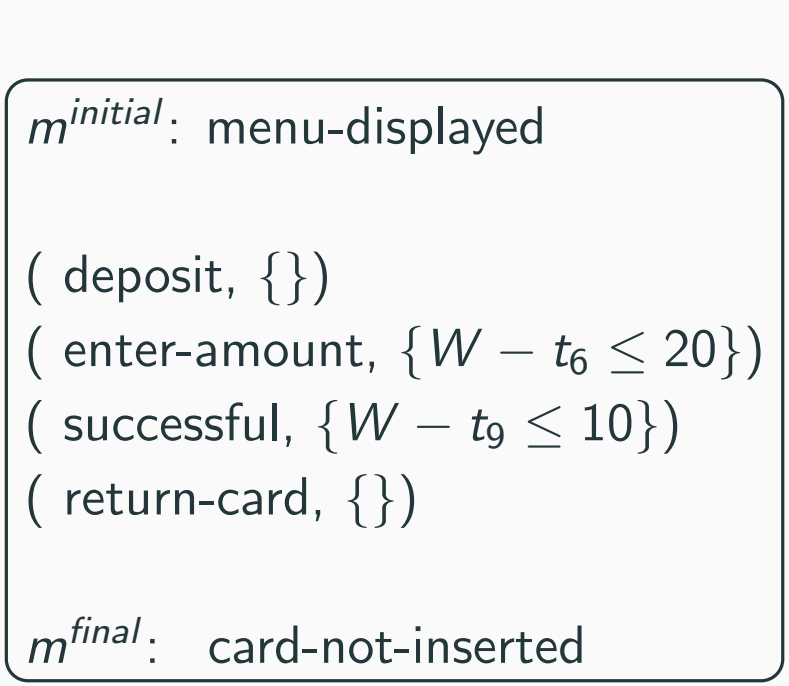


Figure 14: Timed automaton synthesized from Scenario 1 and Scenario 2

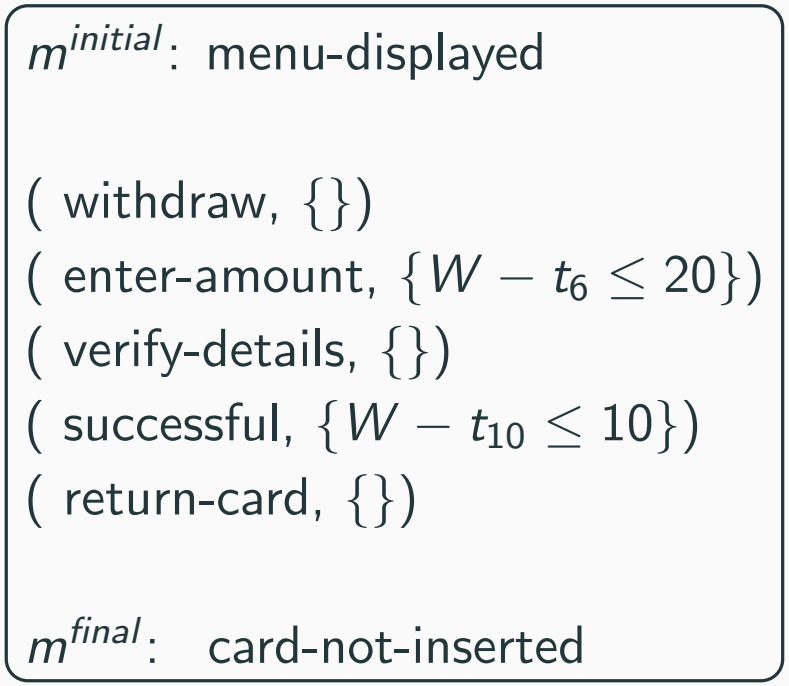
Automated Teller Machine (ATM)

Explain extended scenario

Automated Teller Machine (ATM)



TES of Scenario 3



TES of Scenario 4

Figure 15: Timed Event Sequences of the ATM with withdraw and deposit option

Automated Teller Machine (ATM)

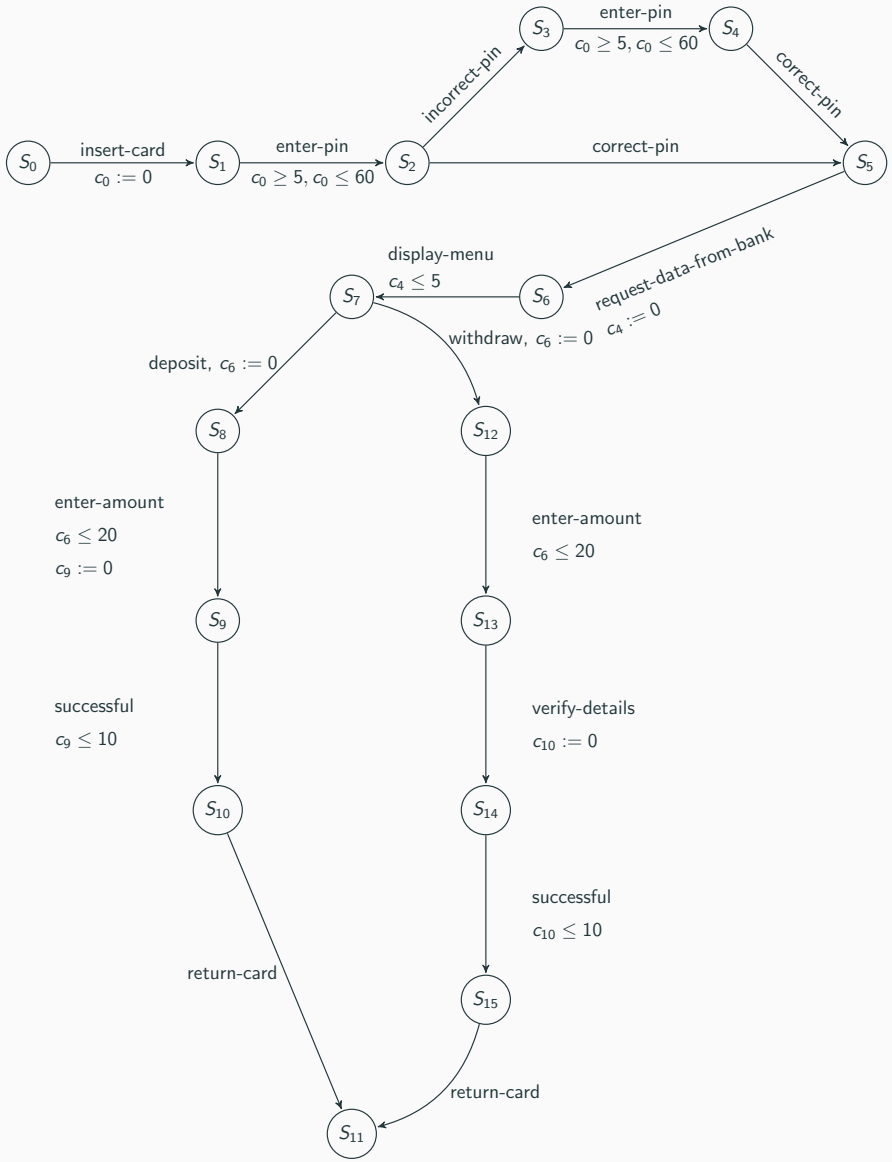


Figure 16: The synthesized timed automaton of the ATM

Light Control System

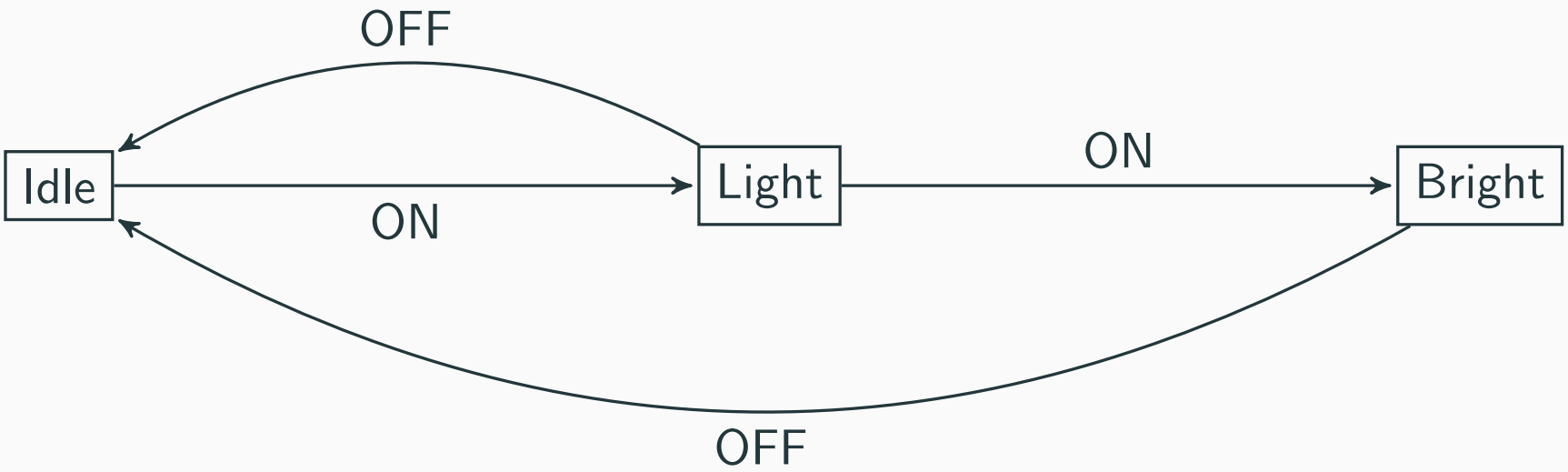


Figure 17: Mode graph of the Light Control System

Light Control System

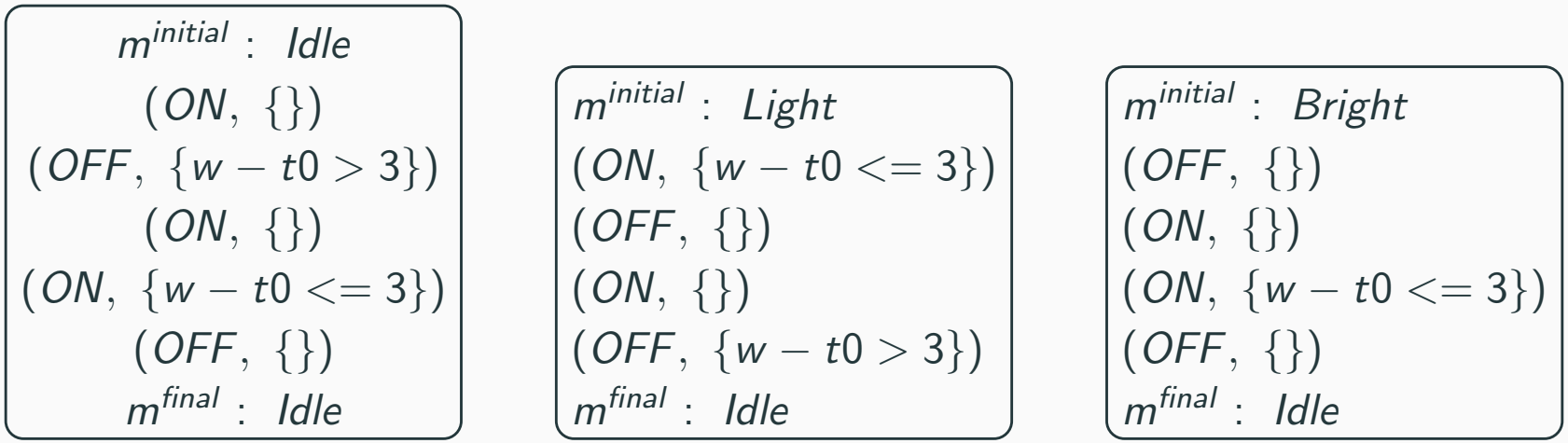


Figure 18: Timed Event Sequences of the Light Control System

Light Control System

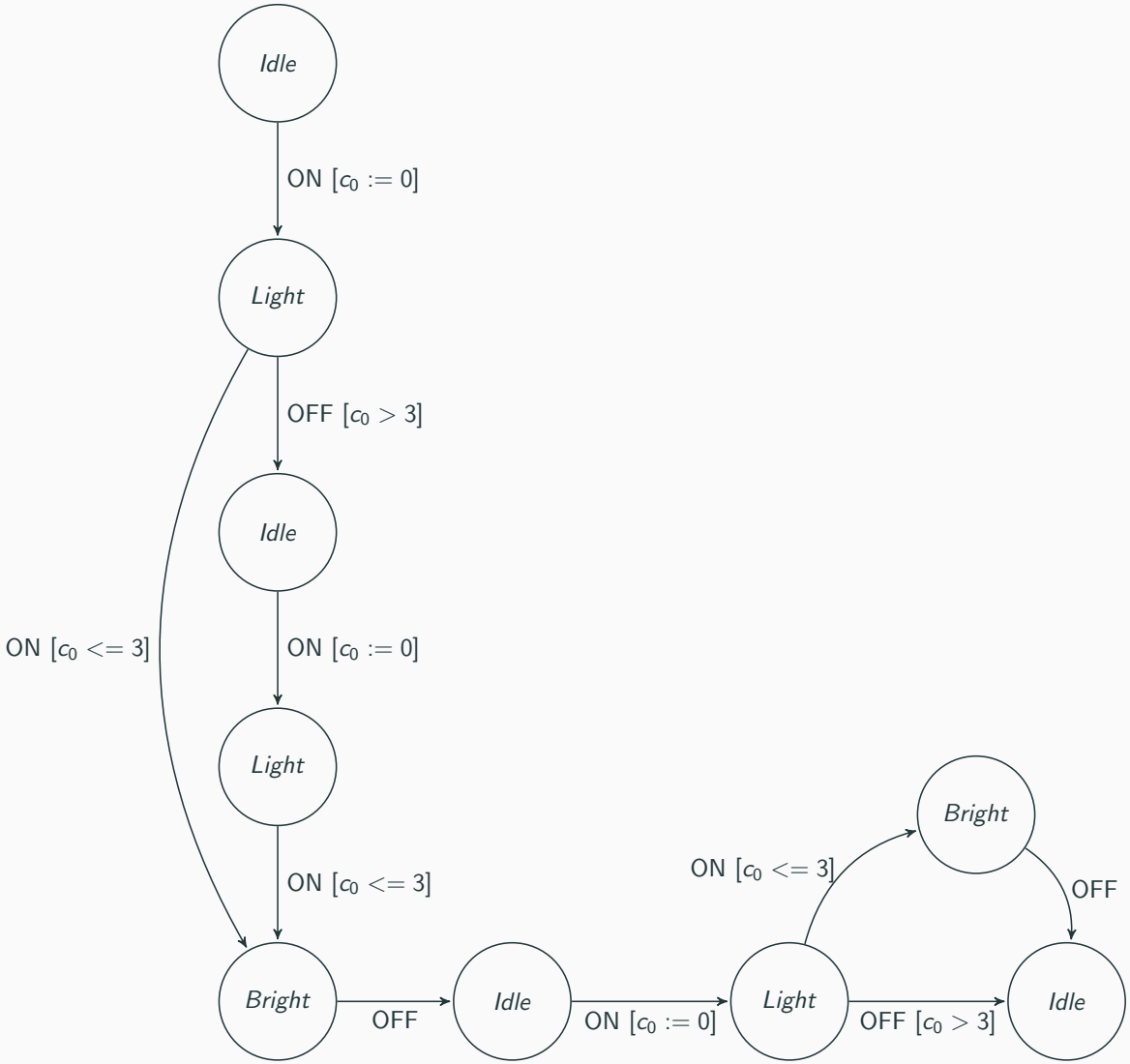


Figure 19: Timed automaton of the Light Control System

Traffic Light

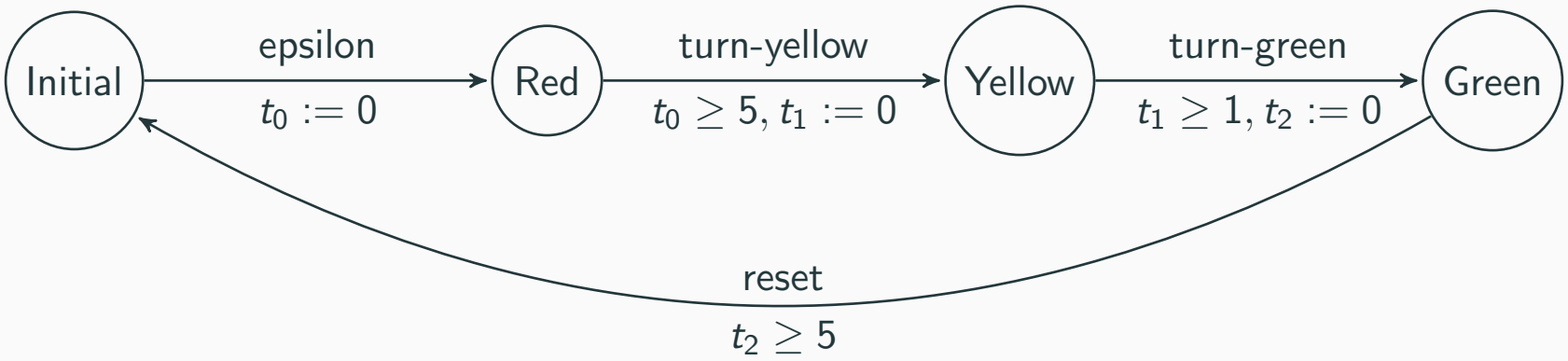


Figure 20: Timed automaton of the Traffic Light

Traffic Light

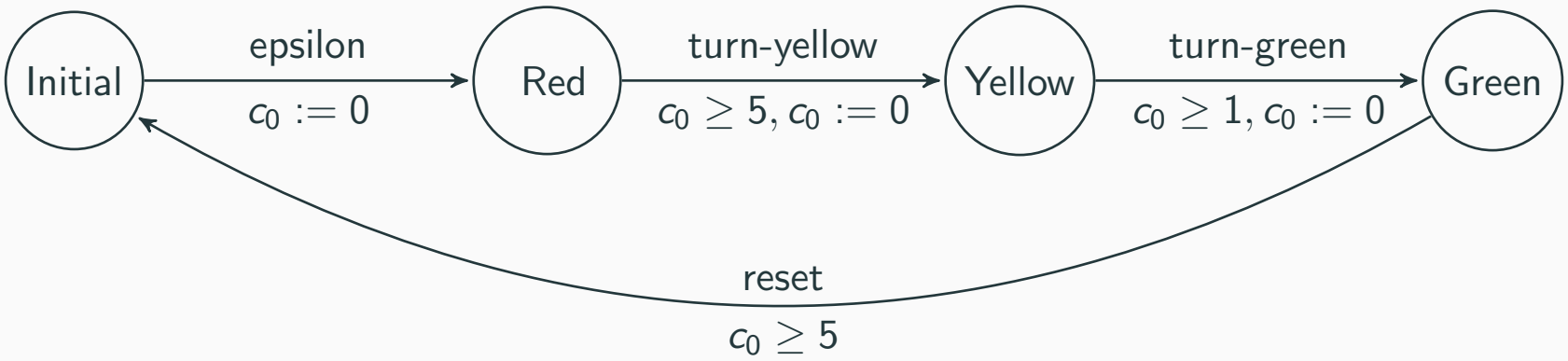


Figure 21: The optimally allocated timed automaton of the Traffic Light

CSMA/CD Protocol

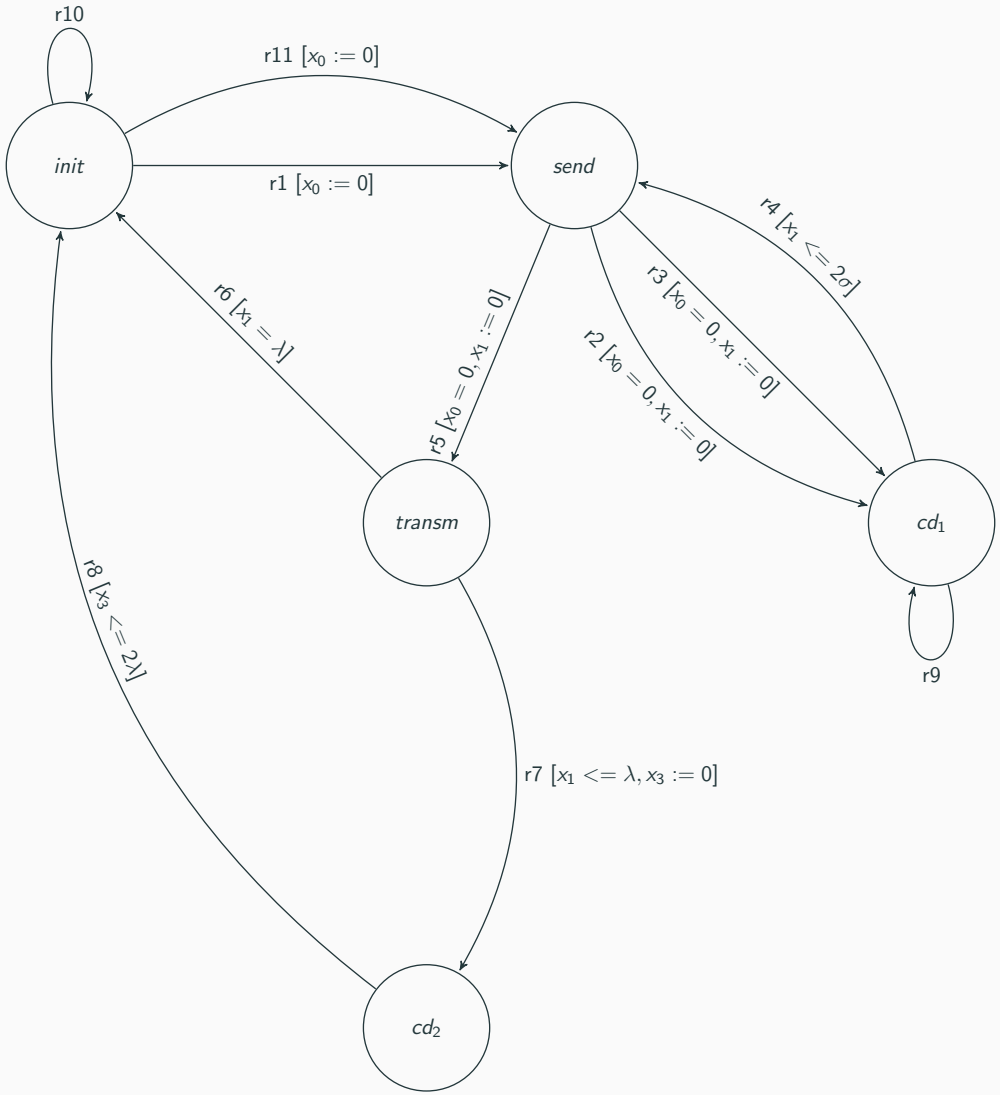


Figure 22: The timed automaton for the sender in CSMA/CD protocol

CSMA/CD Protocol

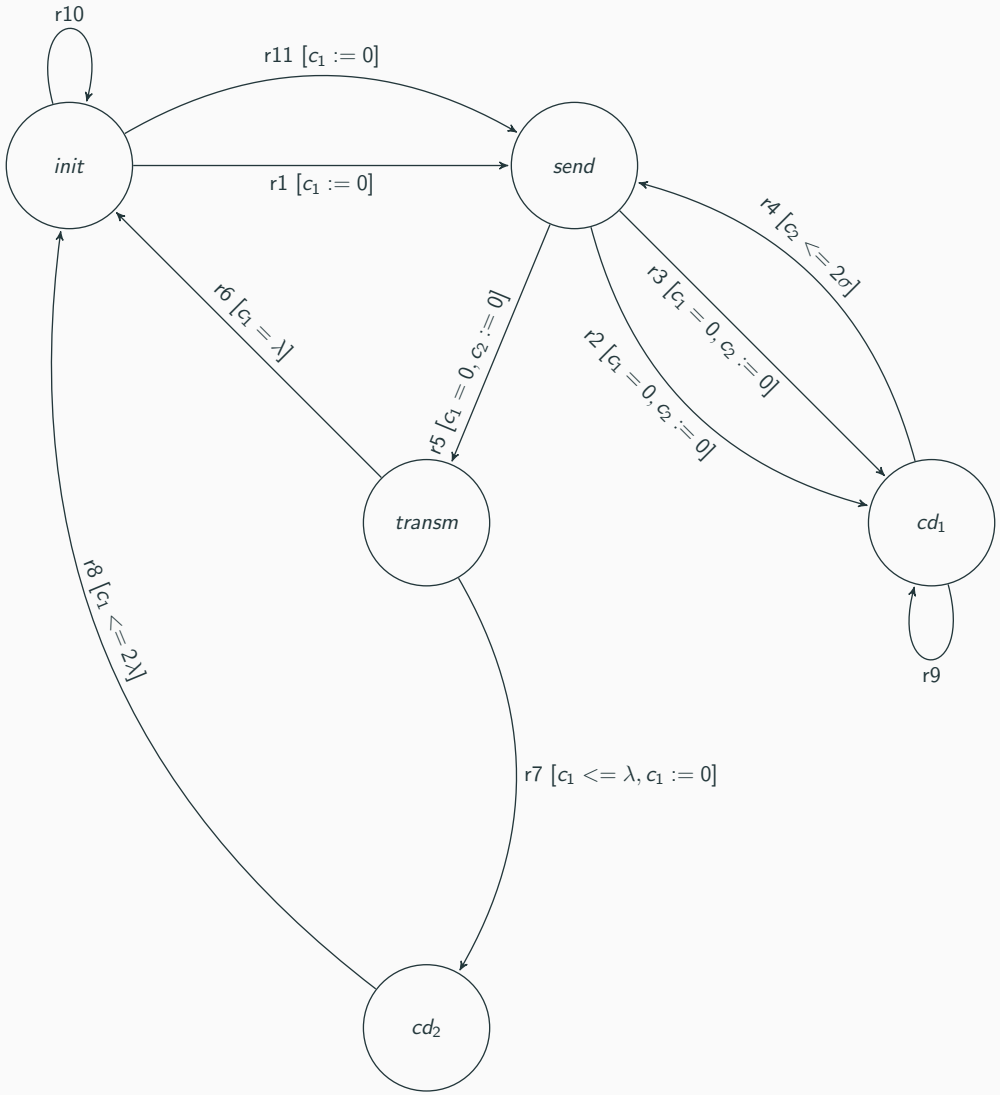


Figure 23: The optimally allocated timed automaton for the sender in CSMA/CD protocol

Conclusion

Conclusion

conclude here [3] [4]

THANK PROFESSORS??? Neda and Committee mem??

Questions?

References i

R. Alur and D. L. Dill.

A theory of timed automata.

Theor. Comput. Sci., 126(2):183–235, Apr. 1994.

T. Lengauer and R. E. Tarjan.

A fast algorithm for finding dominators in a flowgraph.

volume 1, pages 121–141, New York, NY, USA, Jan. 1979. ACM.

N. Saeedloei.

From scenarios to timed automata.

Technical report, Department of Computer Science, University of Minnesota Duluth, Duluth, MN, jun 2016.

References ii


N. Saeedloei and F. Kluzniak.
Optimal clock allocation for a class of timed automata.
Technical report, Department of Computer Science, University of Minnesota Duluth, Duluth, MN, Sept 2016.