Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Abstraction Refinement-Based Verification of Timed Automata

Ph.D. Thesis Booklet

Tamás Tóth

Thesis supervisor:
István Majzik, Ph.D. (BUTE)

Budapest
2021

1 Preliminaries and Objectives

The prevalence of *embedded systems* in everyday use is ever increasing. This also includes their application in *safety critical systems*, e.g. in the automotive, railway or avionic domain. Often, safety critical systems are also *real-time systems* with time-dependent behavior and requirements. The correctness of such systems is crucial, as a system level failure might lead to disastrous consequences, such as environmental harm, loss of valuable equipment, or even human injury. Therefore, in order to reduce the probability and seriousness of faults, these systems are specified in detail, and conformance to the specification is thoroughly verified.

Formal methods are mathematical techniques that enable the rigorous specification and verification of hardware and software systems, typically in design time. *Formal verification* techniques are formal methods for reasoning about the correctness of systems with respect to a formal specification or property. *Model checking* [EC82; QS82] is an automatic formal verification technique that is based on exhaustive traversal of the design model's state space. Its main advantage to more conventional verification methods (e.g. testing) is that it is not only able to detect faults in faulty systems, but can also show that a correct system is fault-free. However, a major difficulty in the successful application of model checking to verification of practical systems is its high computational cost: the cardinality of a system's state space is typically exponential in the size of the input specification describing the system's behavior, a phenomenon commonly known as *state space explosion*. In addition, the state space is not necessarily finite, in particular for real-time systems, where continuous variables with time dimension are part of the specification.

Therefore, to make the problem more tractable, advanced model checkers rely on *symbolic techniques* [Bur+92], where, instead of individual states, sets of states are considered during state space traversal; and *abstraction* [CGL94], where only parts of the system that are relevant for the requirement are considered. As a result, the abstracted system is a simpler system whose behavior overapproximates that of the original system, therefore, if the abstract system is correct, so is the original one. However, as the abstracted system might admit false negatives, that is, spurious faulty behavior that is not present in the original system, the key challenge is finding the right abstraction granularity. This process can be automated using *abstraction refinement* [Cla+00] techniques: in case of a false negative, the abstraction is refined in a way that excludes the discovered faulty behavior.

Goals. In order for model checking to be applicable for the verification of a given system, one has to model the examined aspects of the system's behavior in a suitable modeling formalism beforehand. Most model checking algorithms solve a particular verification task for a given formalism. However, as new designs to verify emerge, more generic tools are also needed since the appropriate formalism and algorithm may vary based on the characteristics of the task itself, and might not be known initially. Our goal is to provide a generic, modular and configurable *model checking framework* that supports the development and evaluation of *abstraction refinement-based algorithms* for checking *safety properties* over different formalisms. In particular, by specific instantiations of our framework, we aim to provide efficient algorithms for the model checking of *real-time systems*. We focus primarily on classical *timed automata* with continuous *clock variables*, a formalism prominently used in the area of model checking real-time systems, and its extension with *discrete variables*. Moreover, we investigate methods for proving *liveness properties* of industrial real-time systems with asynchronous message passing.

Structure. The rest of this thesis booklet is structured as follows. Section 1.1 summarizes the challenges addressed in this work. Section 2 briefly describes the applied research methodology, and outlines the new results of the dissertation. In Section 3, a short summary of the application of the results is given. Finally, Section 4 lists publications of the author relevant to the dissertation.

1.1 Summary of Challenges

In the dissertation, we aim to address the following challenges.

Challenge 1. *Configurable abstraction refinement-based model checking.* Most tools focus on a specific algorithm and formalism to solve a particular verification task. Is it possible to provide a generic, modular and configurable model checking framework that supports the development, evaluation and application of abstraction refinement-based algorithms for the reachability analysis of models in different formalisms?

Challenge 2. *Abstraction refinement for timed automata.* Abstraction refinement has been successfully used in model checking, and in particular for model checking software. Is it possible to provide abstraction refinement algorithms that are efficient in the domain of real-time systems?

Challenge 3. *Model checking timed automata with discrete variables.* For practical real-time systems, design models typically contain discrete data variables with nontrivial data flow besides real-valued clock variables. Is it possible to provide methods for alleviating state space explosion in such models?

Challenge 4. *Liveness checking for industrial real-time systems.* Requirements for industrial real-time systems are often formalized in terms of liveness properties. Is it possible to provide methods for liveness checking of such systems, while still supporting the various semantic features that are present in such models?

Each one of these challenges is addressed in the dissertation in its own thesis.

2 Research Methods and New Results

During my research, I adopted the following research methodology. Building on the literature, I proposed novel verification methods and *algorithms*, and formally *proved* their correctness. To demonstrate that it is feasible in practice, I built *prototype* implementations of the algorithms. Finally, I performed *experiments* on a wide range of input models to evaluate the usability and performance of the proposed algorithms.

2.1 Research Context

In this subsection, we give a short formal description for the formalisms whose model checking the theses address.

2.1.1 Timed Automata

Timed automata [AD94] is a widely used formalism for the modeling and verification of real-time systems. Intuitively, a timed automaton is similar to a finite automaton with locations, edges (or transitions) and a distinguished initial location; however, it also contains real-valued clock variables – and, to label edges, guards and assignments over clock variables – to express time-dependent behavior.

Let C be a set of clock variables over $\mathbb{R}_{\geq 0}$. A *clock constraint* over C is a conjunction of atoms of the form $x \prec k$ or $x - y \prec k$ where $x, y \in C$ and $k \in \mathbb{Z}$ and $\prec \in \{<, \leq\}$. We denote the set of clock constraints over C by $Constr_C$. Syntactically, a timed automaton is a tuple $\mathcal{A} = (L, C, T, \ell_0)$ where

- L is a finite set of locations,
- C is a finite set of clock variables,
- $T \subseteq L \times Constr_C \times \mathcal{P}(C) \times L$ is the transition relation where for a transition $(\ell, g, R, \ell') \in T$, constraint g is a guard and R is a set containing clocks to be reset, and

- $\ell_0 \in L$ is the initial location.

A state of a timed automaton is a pair (ℓ, η) where $\ell \in L$ and $\eta : C \to \mathbb{R}_{\geq 0}$. Let $\eta_0 = x \mapsto 0$. Let $delay_d(\eta)(x) = \eta(x) + d$ for all $x \in C$. Finally, let $reset_R(\eta)(x) = 0$ if $x \in R$ and $reset_R(\eta)(x) = \eta(x)$ if $x \notin R$. The operational semantics of a timed automaton is given by a labeled transition system with initial state (ℓ_0, η_0) and two kinds of transitions:

$$\frac{d \geq 0}{(\ell, \eta) \xrightarrow{d} (\ell, delay_d(\eta))} \text{ delay}$$

$$\frac{t = (\ell, g, R, \ell') \qquad t \in T \qquad \eta \models g}{(\ell, \eta) \xrightarrow{t} (\ell', reset_R(\eta))} \text{ discrete}$$

A *path* of a timed automaton is a sequence transitions $\pi \in T^*$. A path is *feasible* iff, according to semantics of the automaton, there is a corresponding sequence of consecutive states, starting with an initial state. The location reachability problem deals with the question whether a given error location is reachable from an initial state along a feasible path of the automaton, or network of automata.

2.1.2 Calendar Systems

The calendar system formalism is a higher level formalism for describing timed systems as transition systems. Its main idea is based on that of discrete event simulation: instead of clocks of timed automata (that store the time elapsed since a past event), it uses variables to store events scheduled to occur at a point of time in the future. Although this way time progresses to infinity, resulting in an infinite state space, the formalism is easy to handle with induction.

Let $\Delta = \{[a, b], (b, c], [b, c), (b, c) \mid 0 \leq a \leq b < c \text{ and } a, b, c \in \mathbb{N}\}$ and $A^? = A \cup \{\mathsf{none}\}$ and $A^! = A \setminus \{\mathsf{none}\}$. A calendar system is a tuple $(L, T, M, \to, \ell_0, T_0)$ where

- L is a finite set of locations,
- T is a finite set of timeouts,
- M is a finite set of messages,
- $\to \subseteq L \times Event \times Action_M \times \mathcal{P}(Action_T) \times L$ is the transition relation, where $Event = T \cup M$ is the set of *triggering events*, $Action_M = (M \times \Delta)^?$ is the set of *message sending actions* and $Action_T = T \times \Delta^?$ is a set of *timeout setting actions*,
- $\ell_0 \in L$ is the initial location, and finally,
- $T_0 : T \to \Delta^?$ is a function that assigns timeouts their initial value.

A state of a calendar system is a pair (ℓ, σ) with $\ell \in L$ and σ a function with domain $T \cup \{C, \tau\}$ such that $\sigma(\tau) \in \mathbb{R}_{\geq 0}$ tracks the current time, $\sigma(x) \in \mathbb{R}^?_{\geq 0}$ tracks the current value of a timeout $x \in T$, and with multiset $\sigma(C)$, called the *event calendar*, where for an element $(m, t) \in \sigma(C)$, number $t \in \mathbb{R}_{\geq 0}$ is the point in time message $m \in M$ is scheduled to occur. Initial states are of the form (ℓ_0, σ_0) where $\sigma_0(\tau) = 0$ and $\sigma_0(C) = \emptyset$ and for all $x \in T$ we have $\sigma_0(x) \in T_0(x)$ if $T_0(x) \in \Delta$ and $\sigma_0(x) = \mathsf{none}$ otherwise. Moreover, for each transition $\ell \xrightarrow{e,\mu,S} \ell'$ of the calendar system, there is a transition $(\ell, \sigma) \xrightarrow{e} (\ell', \sigma')$ in the transition system defining its semantics such that $\sigma'(\tau) = \min(\sigma(T)^! \cup (\mathsf{snd} \circ \sigma)(C))$, and the following conditions hold.

- For all $x \in T$, exactly one of the following rules applies for the next value of timeout x.

$$\frac{(x, \delta) \in S \qquad \delta = \mathsf{none}}{\sigma'(x) = \mathsf{none}} \text{ invalidate } x$$

$$\frac{(x,\delta) \in S \qquad \delta \in \Delta \qquad d \in \delta}{\sigma'(x) = \sigma(x) + d} \text{ set } x$$

$$\frac{\forall \delta.(x,\delta) \notin S}{\sigma'(x) = \sigma(x)} \text{ skip } x$$

- Exactly one of the following rules applies for the next value of the calendar C.

$$\frac{e \in T \qquad \sigma(e) = \sigma(\tau) \qquad \mu = \text{none}}{\sigma'(C) = \sigma(C)} \; e \text{ over / send none}$$

$$\frac{e \in T \qquad \sigma(e) = \sigma(\tau) \qquad \mu = (m,\delta) \qquad d \in \delta}{\sigma'(C) = \sigma(C) \cup \{(m, \sigma(t) + d)\}} \; e \text{ over / send } m$$

$$\frac{e \in M \qquad (e,\sigma(t)) \in \sigma(C) \qquad \mu = \text{none}}{\sigma'(C) = \sigma(C) \setminus \{(e,\sigma(t))\}} \; e \text{ received / send none}$$

$$\frac{e \in M \qquad (e,\sigma(t)) \in \sigma(C) \qquad \mu = (m,\delta) \qquad d \in \delta}{\sigma'(C) = \sigma(C) \setminus \{(e,\sigma(t))\} \cup \{(m, \sigma(t) + d)\}} \; e \text{ received / send } m$$

2.2 A Framework for Abstraction Refinement-Based Reachability Checking

There are several model checking tools implementing algorithms for various formalisms. Most tools focus on a specific algorithm and formalism to solve a particular verification task efficiently, typically by applying abstraction refinement over a particular domain. Several tools, e.g. SLAM [BR01], BLAST [Bey+07] and SATABS [Cla+05] are based on predicate abstraction. Lazy abstraction tools like IMPACT [McM06] and WOLVERINE [KW11] use Craig interpolation [McM03] to compute abstractions over the predicate domain without expensive post-image computation. Some tools apply abstraction refinement over domains other than predicates: the tool DAGGER [Gul+08] supports refinement for octagon and convex polyhedra domains, and the algorithm VINTA [AGC12] applies abstraction refinement over intervals. Frameworks CPACHECKER [BK11] and UFO [Alb+12] support configurability by the definition of abstract domains, post operators and refinement strategies, but only targeting software models. The LTSMIN tool supports various formalisms through its Partitioned Next-State Interface (PINS) [Kan+15], but instead of abstraction refinement, its main focus is on symbolic and parallel model checking algorithms.

However, as the appropriate formalism and algorithm may vary based on the characteristics of the task itself, and might not even be known initially, it would beneficial for a verification tool to offer a particular algorithm for several formalisms, or conversely, to offer several algorithms for a particular formalism. For this end, we developed the architecture of THETA, which is a generic, modular and configurable model checking framework, aiming to support the development and evaluation of abstraction refinement-based algorithms for the reachability analysis of different formalisms. It primarily aims to support researchers by providing a framework where new components and combinations can easily be implemented, evaluated and compared.

The main distinguishing characteristic of THETA is its architecture that allows the combination of various abstract domains and strategies for abstraction and refinement, applied to models of various formalisms with higher level language frontends. Figure 1 shows the architecture of THETA (with continuous arrows representing data flow, and dashed arrows representing dependence). The main parts of the framework are the formalism and language frontends, the analysis backend and the SMT solver interface. The core of the framework, the analysis backend, consists of three main parts: the abstract domain, the interpreter and the abstraction refinement loop for reachability analysis, with only the interpreter being strictly dependent on the formalisms.

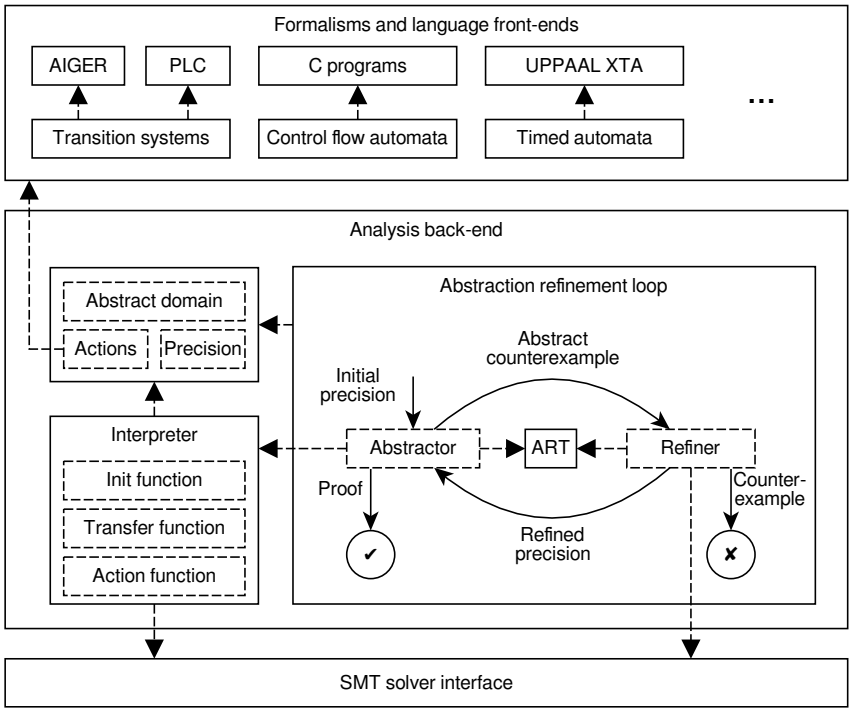

Figure 1. Architecture of the THETA framework

Abstract domain. The semantic basis of the analysis is an *abstract domain* with a set of abstract states, its bottom element and a preorder over the states. The accuracy of a given analysis is formally represented by an element of a set of precisions. (As a typical example, the precision might be a set of state variables that the analysis is expected to track – the larger the cardinality of this set is, the more precise the analysis is.) The formalism for which the analysis is performed defines a set of actions, that serve as input to post-image computation.

Interpreter. Given a precision, an *interpreter* defines an abstract operational semantics over the abstract domain and set of actions. The abstract initial states are given by an *init function*. For an action, the abstract successors of a state are computed by a *transfer function*. An *action function* determines for an abstract state a set of actions that are enabled from that state. The interpreter plays an important role in the generality of the framework, as it decouples the notion of abstraction (represented by the abstract domain over which it is defined) from the notion of formalism (represented by the set of actions over which it is defined).

Abstraction refinement loop. The reachability analysis is performed by the *abstraction refinement loop*. As usual for lazy abstraction methods [McM06], its central data structure is an *abstract reachability tree* (ART), with nodes annotated with abstract states that represent over-approximations of reachable states along a given path, and edges annotated with actions. The ART is manipulated by the two main components of the loop. Using an interpreter, the *abstractor* constructs the ART w.r.t the current precision and an abstraction strategy, the latter of which is determined by the following basic operations.

- *Expand*. When should the abstractor expand a node, i.e. grow the tree by computing and adding to the tree all its abstract successors?
- *Cover*. How should the abstractor attempt to prune the search by looking for covering nodes, i.e. nodes that represent abstract states that entail the abstract state of the current node?

- *Terminate.* Under what conditions should the state space exploration terminate?

If no target nodes – nodes that are deemed unsafe based on the input model – are encountered, the constructed ART serves as an evidence for the safety of the input model. Otherwise, given a target node, the *refiner* is invoked to analyze the abstract path for feasibility. If the path is feasible, it is a counterexample to safety. Otherwise, the refiner carries out its refinement strategy to ensure that the analysis can continue without encountering the same spurious counterexample again (refinement progress). This can typically be achieved by pruning nodes and computing a new analysis precision (overapproximation-driven approach), or by uncovering nodes and strengthening labels (underapproximation-driven approach), both of which includes partial deconstruction of the ART.

Concrete tools were also built for the verification of transition systems, control flow automata and timed automata, combining different abstract domains (including predicates, explicit values and zones) and refinement strategies (including interpolation and unsat cores). In particular, by defining an abstraction refinement loop, and several abstract domains with interpreters, we addressed the location reachability problem of timed automata with discrete variables. Overall, we propose a *formal algorithmic framework* that enables the uniform formalization of several abstract domains and refinement strategies for both clock and discrete variables. The main advantage of the framework is that, based on the notion of the direct product abstract domain, it allows the *seamless combination* of various abstraction methods, resulting in many distinct algorithm configurations that together admit efficient verification of a wide range of timed automata models. The framework provides a generic algorithm for reachability checking, and is configurable in the given abstract domain, and the particular abstraction refinement strategy applied over the domain. The configurability of this framework also allowed the integration of existing efficient lazy abstraction algorithms for clock variables, thus admitting the combination and comparison of our methods with the state-of-the-art.

Thesis 1 *A framework for abstraction refinement-based reachability checking.* I proposed solutions for making abstraction refinement based model checking configurable in terms of modeling formalism, abstract domain, and refinement strategy.

1.1 *Architecture of a configurable model checking framework.* I designed the architecture, interfaces and generic algorithmic components of Theta, a generic, modular, and configurable model checking framework that enables the combination of various abstract domains, interpreters, and strategies for abstraction and refinement, applied to models of various formalisms.

1.2 *A uniform formalization of abstraction refinement strategies for timed automata.* I proposed and proved correct a formal algorithmic framework that enables the uniform formalization and combined use of various abstract domains and abstraction refinement strategies for the location reachability checking of timed automata.

Our results enabled the definition, implementation and empirical evaluation of novel algorithms and algorithm combinations. The results of Thesis 1 are presented in Chapter 3 and Chapter 4 of the dissertation. Related publications are the following: [*j2*; *c6*; *c10*] .

2.3 Lazy Reachability Checking for Timed Automata using Interpolants

The reachability problem of timed automata [AD94] deals with the question whether a given error location is reachable from an initial state along the transitions of the automaton. The standard solution of this problem involves performing a forward exploration in the so-called zone-graph induced by the automaton [DT98]. There, each abstract state is a zone, a special set of concrete states that can be represented as the solution set for a set of clock constraints.

To ensure performance and termination, model checkers for timed automata usually apply some sort of generalization of zones based on maximal lower- and upper bounds [Beh+04] (LU-bounds) appearing in the guards of the automaton. This can be performed directly by extrapolation [Beh+04] parameterized by bounds obtained by static analysis [Beh+03]. Alternatively, bounds can be propagated on-the-fly for all transitions [Her+11] or along a spurious path [HSW13], which, combined with an efficient method for inclusion checking [HSW12] with respect to an abstraction induced by the bounds, results in an efficient method for reachability checking of timed automata. This latter approach is a form of lazy abstraction, a variant of counterexample-guided abstraction refinement [Cla+03] (CEGAR), where – instead of eagerly computing abstractions using an abstract post-image operator, a typically expensive operation – abstraction is computed on-the-fly and locally in the state space along a single execution path where more precision is necessary.

We propose a similar lazy algorithm for reachability checking of timed automata. However, instead of propagating the bounds appearing in guards, the algorithm considers the guards themselves: if the abstraction is too coarse to exclude an infeasible path, we compute an interpolant [McM03], that is, an inductive sequence of zones weak enough to contain all reachable states, while strong enough to block the infeasible transition. In a similar fashion, we use interpolation to effectively prune the search space by enforcing coverage of a newly discovered state with an already visited state when possible. We propose two refinement strategies in this framework: the "forward" zone interpolation strategy propagates the interpolant forward using post-image computation; whereas the "backward" zone interpolation strategy propagates bad zones, obtained as the complement of the interpolant, backward using pre-image computation. Both methods are a combination of forward search, backward search and zone interpolation, and can be considered as a generalization of zone interpolation to sequences of transitions of a timed automaton.

Thesis 2 *Lazy reachability checking for timed automata using interpolants.* I proposed a solution for the location reachability problem of timed automata based on the following steps.

- I defined interpolation for zones, and gave an algorithm for computing a zone interpolant from two inconsistent zones, represented as canonical difference bound matrices.
- Based on pre- and post-image computation for timed automata in the zone abstract domain, I generalized the notion of zone interpolation to sequences of interpolants, this way enabling its use for abstraction refinement-based location reachability checking of timed automata.
- I proposed forward and backward zone interpolation as approaches to lazy abstraction refinement.
- I experimentally evaluated the performance of the proposed abstraction refinement strategies, and showed that these compare favorably to known methods based on efficient lazy non-convex abstractions.

The proposed method is applicable to more expressive variants of timed automata, e.g. to automata with diagonal constraints in guards [BLR05], or to updatable timed automata [Bou04]. The results of Thesis 2 are presented in Chapter 5 of the dissertation. Related publications are the following: [*j2*; *c8*; *c9*] .

2.4 Lazy Reachability Checking for Timed Automata with Discrete Variables

In the context of timed automata, methods rarely address the problem of *abstraction for discrete data variables* that often appear in specifications for practical real-time systems, or do so by

applying a fully SMT based approach, relying on the efficiency of underlying decision procedures for the abstraction of both continuous and discrete variables.

In our work, we address the location reachability problem of timed automata with discrete variables by proposing an abstraction method that can be used to *lazily control the visibility of discrete variables* occurring in such specifications: if the abstraction is too coarse to exclude an infeasible transition – a transition that makes the whole path infeasible –, we then extract a set of visible variables [Kur94; CGS04; Cha+02; Gru06] whose value is sufficient to be tracked along the given path to exclude the infeasible transition from the abstract state space. We combine interpolation [McM03] – defined for variable assignments – and weakest precondition computation for backward propagation to compute a coarse abstraction. We propose two refinement strategies in this framework: the "forward" valuation interpolation strategy propagates the interpolant forward using post-image computation; whereas the "backward" valuation interpolation strategy propagates the formula characterizing the interpolant backward using weakest precondition computation. Our method does not rely on an interpolating SMT solver, and can be freely combined with (eager or lazy) zone-based forward search methods for efficient handling of clock variables.

Thesis 3 *Lazy reachability checking for timed automata with discrete variables.* I proposed a solution for the location reachability problem of timed automata with discrete variables based on the following steps.

- I defined interpolation between a valuation and a formula, and gave an algorithm for computing valuation interpolants.
- Based on weakest precondition computation for transitions of timed automata, I generalized the notion of valuation interpolation to sequences of interpolants, this way enabling its use for abstraction refinement-based location reachability checking.
- I proposed forward and backward valuation interpolation as approaches to lazy abstraction refinement.
- I experimentally evaluated the performance of the proposed abstraction refinement strategies, and showed that these are suitable to significantly reduce the number of states generated during state space exploration of timed automata models with many discrete variables.

The proposed method does not rely on SMT solving, and is thus applicable to models with arbitrary expressions and statements over discrete variables, e.g. division, multiplication between variables, etc. The results of Thesis 3 are presented in Chapter 6 of the dissertation. Related publications are the following: [*j*1; *j*2; *c*4; *c*7; *c*11; *c*12; *e*13] .

2.5 Improved Methods for Liveness Checking of Industrial Real-Time Protocols

Safety properties are properties of systems that can be decided by inspecting finite prefixes of infinite executions. Reachability properties – whose verification the previous theses address – are the simplest kind of safety properties that express that no bad state along an execution of the system is reachable. In contrast to safety properties, liveness properties are properties where each finite execution can be extended to an infinite execution that satisfies the property, typically expressing that some desired behavior eventually occurs. Although many efficient symbolic model checkers exist for the verification of safety properties of timed systems, the verification of liveness properties is less frequently addressed. During our research, we addressed the problem of liveness verification of industrial real-time protocols.

For asynchronous message passing systems, we proposed a method based on the extension of k-induction [SSS00; BC00; MRS03; ES03]. The main idea of k-induction is the application of mathematical induction on the length of the paths in the system's state space to prove its safety,

however, with an unrolling of transitions in length k, instead of the usual 1, to make the induction step more likely to succeed. Although k-induction has originally been applied for model checking safety properties of hardware circuits, we extended it to liveness checking of timed systems. The starting point of the proposed method is a system model, specified in an high-level modeling language for timed systems, called the calendar system formalism, which enables the convenient description of timed asynchronous message passing systems. The intermediate model is translated to a symbolic transition system specification. The specification is then extended with observer components that reduce liveness checking to safety checking, and with simple lemmas extracted from the original model that are used for strengthening the checked property, which helps the induction step succeed. The complete specification is then checked with a k-induction model checker.

Safety critical systems are mainly composed hierarchically, where different layers of functions rely on each other. In addition, the specified properties in real-life systems are typically complex in the sense that they are usually combinations of safety and liveness queries. For the verification of such systems, we propose a method to decompose this layered structure of the specification to smaller and more tractable verification problems.

Thesis 4 *Improved methods for liveness checking of industrial real-time protocols.* During my research, I proposed improved methods for liveness verification of industrial real-time protocols.

4.1 *K-induction based liveness checking of real-time systems.* I proposed the calendar system formalism that allows convenient modeling of the core protocols of communicating real-time systems. By a series of transformation steps, I extended k-induction based model checking to support the verification of both safety and liveness properties of calendar systems. Moreover, I provided a tool-supported solution for the derivation of lemmas required for successful k-induction based automated verification.

4.2 *A decomposition method for liveness checking of hierarchical real-time protocols.* I proposed a generic decomposition scheme for the verification of real-time systems with a hierarchical structure in functionality. The method is applicable when a combination of safety and liveness properties shall be verified.

We successfully applied the method during the verification of a distributed safety critical protocol, whose main functionality is to guarantee reliable communication between components in a distributed SCADA (Supervisory Control and Data Acquisition) system. The results of Thesis 4 are presented in Chapter 7 and Chapter 8 of the dissertation. Related publications are the following: [*c*3; *c*5] .

3 Application of the New Results

The algorithms proposed in Theses 1-3 have been implemented in the open source model checking framework Theta[1]. Related applications are as follows.

- We successfully used Theta in an *industrial collaboration* with CERN for safety verification of PLC code.
- Theta is intensively used in our *education*, both in our labs to teach students the basic principles of model checking and abstraction, as well as in individual student projects as a basis for fast prototyping formal verification tools.
- *Research* based on Theta unrelated to this dissertation is currently in progress[2].

[1] https://github.com/FTSRG/theta

[2] For a complete list of related papers, visit https://ftsrg.mit.bme.hu/theta/publications/.

- Theta has been integrated as a verification backend in Gamma [Mol+18], a tool for modeling and model integration based on statecharts. This way, Theta enabled the verification of selected protocols and algorithms of an electronic railway interlocking system modeled in Gamma.
- For the verification of C programs, the tool Gazer-Theta has been proposed [ÁSH21]. The tool has been submitted to the 10th International Competition on Software Verification (SV-COMP 2021) [Bey21], where it competed in 9 subcategories.

The results in Thesis 4 are the product of our collaboration with Prolan, a company that, among others, deals with development of railway control software. Our methods were applied in the verification of a communication protocol for a SCADA (Supervisory Control and Data Acquisition) system under development that is going to be applied at the highest safety integrity level (SIL 4).

4 Publication List

Number of publications:	16
Number of peer-reviewed journal papers (written in English):	2
Number of articles in journals indexed by WoS or Scopus:	2
Number of publications (in English) with at least 50% contribution of the author:	9
Number of peer-reviewed publications:	16
Number of independent citations:	9

4.1 Publications Linked to the Theses

	Journal papers	International conference and workshop papers	Local events	Technical reports
Thesis 1	[*j*2]*	[*c*6]; [*c*10]	—	—
Thesis 2	[*j*2]*	[*c*8]†; [*c*9]	—	—
Thesis 3	[*j*1]; [*j*2]*	[*c*4]; [*c*7]†; [*c*11]; [*c*12]	[*e*13]	—
Thesis 4	—	[*c*3]; [*c*5]	—	—

* These publications are attached to multiple theses.
† In the years 2016 and 2017, the PhD Minisymposium, organized by the BUTE Department of Measurement and Information Systems, had international participation.
This classification follows the faculty's Ph.D. publication score system.

Journal Papers

[*j*1] Tamás Tóth and István Majzik. Formal verification of real-time systems with data processing. *Periodica Polytechnica Electrical Engineering and Computer Science* 61(2), 2017, pp. 166–174. DOI: 10.3311/PPee.9766.

[*j*2] Tamás Tóth and István Majzik. Configurable verification of timed automata with discrete variables. *Acta Informatica* (online first), 2020. DOI: 10.1007/s00236-020-00393-4.

International Conference and Workshop Papers

[*c*3] Tamás Tóth, András Vörös, and István Majzik. K-induction based verification of real-time safety critical systems. In: *Proceedings of the 8th International Conference on Dependability and Complex Systems, DepCoS-RELCOMEX 2013*, AISC, vol. 224, pp. 469–478. Springer, 2013.

DOI: 10.1007/978-3-319-00945-2_43.
▷ *Own contributions (1) the calendar system formalism (2) the model checking approach (3) the implementation of the tool support (4) the modeling and verification of the case study.*

[*c*4] Tamás Tóth, András Vörös, and István Majzik. Verification of a real-time safety-critical protocol using a modelling language with formal data and behaviour semantics. In: *Computer Safety, Reliability, and Security. SAFECOMP 2014 Workshops*, LNCS, vol. 8696, pp. 207–218. Springer, 2014. DOI: 10.1007/978-3-319-10557-4_24.
▷ *Own contributions (1) the modeling formalism (2) the model checking approach (3) the implementation of the tool support (4) the modeling and verification of the case study.*

[*c*5] Tamás Tóth, András Vörös, and István Majzik. A decomposition method for the verification of a real-time safety-critical protocol. In: *Software Engineering for Resilient Systems. 7th International Workshop, SERENE 2015*, LNCS, vol. 9274, pp. 31–45. Springer, 2015. DOI: 10.1007/978-3-319-23129-7_3.
▷ *Own contributions (1) the model checking approach (2) the modeling and verification of the case study.*

[*c*6] Ákos Hajdu, Tamás Tóth, András Vörös, and István Majzik. A configurable CEGAR framework with interpolation-based refinements. In: *Formal Techniques for Distributed Objects, Components, and Systems. 36th IFIP WG 6.1 International Conference, FORTE 2016*, LNCS, vol. 9688, pp. 158–174. Springer, 2016. DOI: 10.1007/978-3-319-39570-8_11.
▷ *Own contributions (1) some insight on the model checking approach (2) partial implementation of the tool support.*

[*c*7] Tamás Tóth and István Majzik. Formal modeling of real-time systems with data processing. In: *Proceedings of the 23rd PhD Mini-Symposium*, pp. 46–49. BME Department of Measurement and Information Systems. Accommodated by IEEE Hungary, 2016.

[*c*8] Tamás Tóth and István Majzik. Timed automata verification using interpolants. In: *Proceedings of the 24th PhD Mini-Symposium*, pp. 82–85. BME Department of Measurement and Information Systems, 2017. DOI: 10.5281/zenodo.291907.

[*c*9] Tamás Tóth and István Majzik. Lazy reachability checking for timed automata using interpolants. In: *Formal Modeling and Analysis of Timed Systems. 15th International Conference, FORMATS 2017*, LNCS, vol. 10419, pp. 264–280. Springer, 2017. DOI: 10.1007/978-3-319-65765-3_15.

[*c*10] Tamás Tóth, Ákos Hajdu, András Vörös, Zoltán Micskei, and István Majzik. THETA: a framework for abstraction refinement-based model checking. In: *Proceedings of the 17th Conference on Formal Methods in Computer Aided Design, FMCAD 2017*, pp. 176–179. FMCAD Inc., 2017. DOI: 10.23919/FMCAD.2017.8102257.
▷ *Own contributions (1) the design of the architecture, interfaces, and generic algorithmic components of the framework (2) partial implementation of the tool support.*

[*c*11] Tamás Tóth and István Majzik. Lazy reachability checking for timed automata with discrete variables. In: *Model Checking Software. 25th International Symposium, SPIN 2018*, LNCS, vol. 10869, pp. 235–254. Springer, 2018. DOI: 10.1007/978-3-319-94111-0_14.

[*c*12] Rebeka Farkas, Tamás Tóth, Ákos Hajdu, and András Vörös. Backward reachability analysis for timed automata with data variables. In: *Automated Verification of Critical Systems*, Electronic Communications of the EASST, vol. 76, pp. 1–20. 2018. DOI: 10.14279/tuj.eceasst.76.1076.
▷ *Own contributions (1) some insight on the model checking approach (2) partial implementation of the tool support.*

Local Conference and Workshop Papers

[*e*13] Tamás Tóth and István Majzik. A framework for formal verification of real-time systems. In: *Proceedings of the 22nd PhD Mini-Symposium*, pp. 12–13. BME Department of Measurement and Information Systems. Accommodated by IEEE Hungary, 2015.

Supplementary Material

[*s*14] Tamás Tóth and István Majzik. Supplementary Material for the Paper "Configurable Verification of Timed Automata with Discrete Variables". Zenodo. 2020. DOI: 10.5281/zenodo.3965792.

4.2 Additional Publications (Not Linked to Theses)

International Conference and Workshop Papers

[*c*15] Gyula Sallai and Tamás Tóth. Boosting software verification with compiler optimizations. In: *Proceedings of the 24th PhD Mini-Symposium*, pp. 66–69. BME Department of Measurement and Information Systems, 2017. DOI: 10.5281/zenodo.291903.

[*c*16] Bence Czipó, Ákos Hajdu, Tamás Tóth, and István Majzik. Exploiting hierarchy in the abstraction-based verification of statecharts using SMT solvers. In: *International Workshop on Formal Engineering approaches to Software Components and Architectures, FESCA 2017*, EPTCS, vol. 245, pp. 31–45. Open Publishing Association, 2017. DOI: 10.4204/EPTCS.245.3.

[*c*17] Gyula Sallai, Ákos Hajdu, Tamás Tóth, and Zoltán Micskei. Towards evaluating size reduction techniques for software model checking. In: *Fifth International Workshop on Verification and Program Transformation, VPT 2017*, EPTCS, vol. 253, pp. 75–91. Open Publishing Association, 2017. DOI: 10.4204/EPTCS.253.7.

References

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science* 126(2), 1994, pp. 183–235. DOI: 10.1016/0304-3975(94)90010-8.

[AGC12] Aws Albarghouthi, Arie Gurfinkel, and Marsha Chechik. Craig interpretation. In: *Static Analysis*, LNCS, vol. 7460, pp. 300–316. Springer, 2012. DOI: 10.1007/978-3-642-33125-1_21.

[Alb+12] Aws Albarghouthi, Yi Li, Arie Gurfinkel, and Marsha Chechik. UFO: a framework for abstraction- and interpolation-based software verification. In: *Computer Aided Verification*, LNCS, vol. 7358, pp. 672–678. Springer, 2012. DOI: 10.1007/978-3-642-31424-7_48.

[ÁSH21] Zsófia Ádám, Gyula Sallai, and Ákos Hajdu. GAZER-THETA: LLVM-based verifier portfolio with BMC/CEGAR (competition contribution). In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 12652, pp. 433–437. Springer, 2021. DOI: 10.1007/978-3-030-72013-1_27.

[BC00] Per Bjesse and Koen Claessen. SAT-based verification without state space traversal. In: *Formal Methods in Computer-Aided Design*, LNCS, vol. 1954, pp. 409–426. Springer, 2000. DOI: 10.1007/3-540-40922-X_23.

[Beh+03] Gerd Behrmann, Patricia Bouyer, Emmanuel Fleury, and Kim G. Larsen. Static guard analysis in timed automata verification. In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 2619, pp. 254–270. Springer, 2003. DOI: 10.1007/3-540-36577-X_18.

[Beh+04] Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelánek. Lower and upper bounds in zone based abstractions of timed automata. In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 2988, pp. 312–326. Springer, 2004. DOI: 10.1007/978-3-540-24730-2_25.

[Bey+07] Dirk Beyer, Thomas A. Henzinger, Ranjit Jhala, and Rupak Majumdar. The software model checker BLAST. *Software Tools for Technology Transfer* 9(5), 2007, pp. 505–525. DOI: 10.1007/s10009-007-0044-z.

[Bey21] Dirk Beyer. Software verification: 10th comparative evaluation (SV-COMP 2021). In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 12652, pp. 401–422. Springer, 2021. DOI: 10.1007/978-3-030-72013-1_24.

[BK11] Dirk Beyer and M. Erkan Keremoglu. CPACHECKER: a tool for configurable software verification. In: *Computer Aided Verification*, LNCS, vol. 6806, pp. 184–190. Springer, 2011. DOI: 10.1007/978-3-642-22110-1_16.

[BLR05] Patricia Bouyer, François Laroussinie, and Pierre-Alain Reynier. Diagonal constraints in timed automata: forward analysis of timed systems. In: *Formal Modelling and Analysis of Timed Systems*, LNCS, vol. 3829, pp. 112–126. Springer, 2005. DOI: 10.1007/11603009_10.

[Bou04] Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design* 24(3), 2004, pp. 281–320. DOI: 10.1023/B:FORM.0000026093.21513.31.

[BR01] Thomas Ball and Sriram K. Rajamani. The SLAM toolkit. In: *Computer Aided Verification*, LNCS, vol. 2102, pp. 260–264. Springer, 2001. DOI: 10.1007/3-540-44585-4_25.

[Bur+92] Jerry R. Burch, Edmund L. Clarke, Kenneth L. McMillan, David L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation* 98(2), 1992, pp. 142–170. DOI: 10.1016/0890-5401(92)90017-A.

[CGL94] Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. *Transactions on Programming Languages and Systems* 16(5), 1994, pp. 1512–1542. DOI: 10.1145/186025.186051.

[CGS04] Edmund M. Clarke, Anubhav Gupta, and Ofer Strichman. SAT-based counterexample-guided abstraction refinement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23(7), 2004, pp. 1113–1123. DOI: 10.1109/TCAD.2004.829807.

[Cha+02] Pankaj Chauhan, Edmund M. Clarke, James Kukula, Samir Sapra, Helmut Veith, and Dong Wang. Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis. In: *Formal Methods in Computer-Aided Design*, LNCS, vol. 2517, pp. 33–51. Springer, 2002. DOI: 10.1007/3-540-36126-X_3.

[Cla+00] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement. In: *Computer Aided Verification*, LNCS, vol. 1855, pp. 154–169. Springer, 2000. DOI: 10.1007/10722167_15.

[Cla+03] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM* 50(5), 2003, pp. 752–794. DOI: 10.1145/876638.876643.

[Cla+05] Edmund Clarke, Daniel Kroening, Natasha Sharygina, and Karen Yorav. SATABS: SAT-based predicate abstraction for ANSI-C. In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 3440, pp. 570–574. Springer, 2005. DOI: 10.1007/978-3-540-31980-1_40.

[DT98] Conrado Daws and Stavros Tripakis. Model checking of real-time reachability properties using abstractions. In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 1384, pp. 313–329. Springer, 1998. DOI: 10.1007/BFb0054180.

[EC82] E. Allen Emerson and Edmund M. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming* 2(3), 1982, pp. 241–266. DOI: 10.1016/0167-6423(83)90017-5.

[ES03] Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. *Electronic Notes in Theoretical Computer Science* 89(4), 2003, pp. 543–560. DOI: 10.1016/S1571-0661(05)82542-3.

[Gru06] Orna Grumberg. Abstraction and refinement in model checking. In: *Formal Methods for Components and Objects*, LNCS, vol. 4111, pp. 219–242. Springer, 2006. DOI: 10.1007/11804192_11.

[Gul+08] Bhargav S. Gulavani, Supratik Chakraborty, Aditya V. Nori, and Sriram K. Rajamani. Automatically refining abstract interpretations. In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 4963, pp. 443–458. Springer, 2008. DOI: 10.1007/978-3-540-78800-3_33.

[Her+11] Frédéric Herbreteau, Dileep Kini, Balaguru Srivathsan, and Igor Walukiewicz. Using non-convex approximations for efficient analysis of timed automata. In: *Foundations of Software Technology and Theoretical Computer Science*, LIPIcs, vol. 13, pp. 78–89. Dagstuhl, 2011. DOI: 10.4230/LIPIcs.FSTTCS.2011.78.

[HSW12] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. In: *Logic in Computer Science*, pp. 375–384. IEEE, 2012. DOI: 10.1109/LICS.2012.48.

[HSW13] Frédéric Herbreteau, Balaguru Srivathsan, and Igor Walukiewicz. Lazy abstractions for timed automata. In: *Computer Aided Verification*, LNCS, vol. 8044, pp. 990–1005. Springer, 2013. DOI: 10.1007/978-3-642-39799-8_71.

[Kan+15] Gijs Kant, Alfons Laarman, Jeroen Meijer, Jaco van de Pol, Stefan Blom, and Tom van Dijk. LTSmin: high-performance language-independent model checking. In: *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, vol. 9035, pp. 692–707. Springer, 2015. DOI: 10.1007/978-3-662-46681-0_61.

[Kur94] Robert P. Kurshan. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton University Press, 1994. DOI: 10.1515/9781400864041.

[KW11] Daniel Kroening and Georg Weissenbacher. Interpolation-based software verification with Wolverine. In: *Computer Aided Verification*, LNCS, vol. 6806, pp. 573–578. Springer, 2011. DOI: 10.1007/978-3-642-22110-1_45.

[McM03] Kenneth L. McMillan. Interpolation and SAT-based model checking. In: *Computer Aided Verification*, LNCS, vol. 2725, pp. 1–13. Springer, 2003. DOI: 10.1007/978-3-540-45069-6_1.

[McM06] Kenneth L. McMillan. Lazy abstraction with interpolants. In: *Computer Aided Verification*, LNCS, vol. 4144, pp. 123–136. Springer, 2006. DOI: 10.1007/11817963_14.

[Mol+18] Vince Molnár, Bence Graics, András Vörös, István Majzik, and Dániel Varró. The Gamma statechart composition framework: design, verification and code generation for component-based reactive systems. In: *International Conference on Software Engineering*, pp. 113–116. ACM, 2018. DOI: 10.1145/3183440.3183489.

[MRS03] Leonardo de Moura, Harald Rueß, and Maria Sorea. Bounded model checking and induction: from refutation to verification. In: *Computer Aided Verification*, LNCS, vol. 2725, pp. 14–26. Springer, 2003. DOI: 10.1007/978-3-540-45069-6_2.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In: *International Symposium on Programming*, LNCS, vol. 137, pp. 337–351. Springer, 1982. DOI: 10.1007/3-540-11494-7_22.

[SSS00] Mary Sheeran, Satnam Singh, and Gunnar Stålmarck. Checking safety properties using induction and a SAT-solver. In: *Formal Methods in Computer-Aided Design*, LNCS, vol. 1954, pp. 127–144. Springer, 2000. DOI: 10.1007/3-540-40922-X_8.