

Abstraction in parameterised Boolean equation systems

Citation for published version (APA):

Cranen, S., Gazda, M. W., Wesselink, J. W., & Willemse, T. A. C. (2013). *Abstraction in parameterised Boolean equation systems*. (Computer science reports; Vol. 1301). Technische Universiteit Eindhoven.

Document status and date:

Published: 01/01/2013

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Technische Universiteit Eindhoven
Department of Mathematics and Computer Science

Abstraction in Parameterised Boolean Equation Systems

S. Cranen, M.W. Gazda, J.W. Wesselink and T.A.C. Willemse

13/01

ISSN 0926-4515

All rights reserved

editors: prof.dr. P.M.E. De Bra
prof.dr.ir. J.J. van Wijk

Reports are available at:

<http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Author&level=1> and

<http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Year&Level=1>

Computer Science Reports 13-01
Eindhoven, January 2013

Abstraction in Parameterised Boolean Equation Systems

S. Cranen, M.W. Gazda, J.W. Wesselink, and T.A.C. Willemse

Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract. We present a general theory of abstraction for a variety of verification problems. Our theory is set in the framework of parameterised Boolean equation systems. The power of our abstraction theory is compared to that of generalised Kripke modal transition systems (GTSs). We show that for model checking the modal μ -calculus, our abstractions can be exponentially more succinct than GTSs and our theory is as complete as the GTS framework for abstraction. Furthermore, we investigate the completeness of our theory for verification problems other than the modal μ -calculus. We illustrate the potential of our theory through case studies using the first-order modal μ -calculus and a real-time extension thereof, conducted using a prototype implementation of a new syntactic transformation for equation systems.

1 Introduction

Parameterised Boolean equation systems [14], or *equation systems* for short, can be used to encode a diverse set of verification problems, including the (first-order) modal μ -calculus model checking problem [12, 13]; behavioural equivalence problems such as bisimilarity, similarity and branching bisimilarity [1]; model checking real-time systems [23]. By solving an equation system resulting from an encoding of a particular verification problem, an answer to the encoded problem is obtained. Advanced tool suites, such as mCRL2 [11], rely on the use of equation systems for solving their verification problems. Solving equation systems is generally an undecidable problem, but the problem is decidable for fragments of the equation systems such as *Boolean equation systems* [17]; moreover, there are many syntax-based transformations that effectively reduce the complexity of a concrete equation system. Examples of such transformations are static analysis techniques that can detect and remove unimportant data variables [20]; invariant strengthening [21]; and the use of pattern-matching techniques to simplify equations using standard solution templates [14]. Note that such techniques can be applied regardless of the encoded verification problem.

A key instrument in scaling verification techniques is *abstraction*. Automated verification using abstraction relies on the abstraction applied to a system being (1) sufficiently coarse so that the abstract system's behaviour has a finite representation, and (2) sufficiently detailed permitting one to prove the properties of interest or find a valid counterexample. The mathematical richness of the abstract domain used in an abstraction framework affects the set of properties that

can be verified: frameworks using Kripke modal transition systems (MTSs) [10] as the target of abstraction are typically more powerful than frameworks using Kripke structures as their target of abstraction.

In this paper, we study the concept of abstraction in equation systems. Abstraction in equation systems is based on a notion of under-approximation (resp. over-approximation), captured by a coinductive relation on equation systems. This notion, called *consistent consequence*, was introduced in [8] for the fragment of Boolean equation systems; in that setting, a proof system for the relation and the computational complexity of deciding the relation were studied. In this paper, we generalise the relation to arbitrary equation systems. We note that since our abstractions are again equation systems, there is no need to invest in dedicated tools and theories for dealing with the artefacts obtained after abstraction. All available transformations and solving algorithms for equation systems remain applicable to the approximations. We consider this a strong positive.

Since in equation systems, abstraction is defined independent of the encoded verification problems, it is a natural question how its power relates to that of well-known frameworks for abstraction. This can be measured by comparing the degree of *completeness* and *succinctness* of the frameworks. Here, completeness refers to the capacity of a framework to answer a decision problem (*e.g.*, proving a property expressed in a logic) by answering it on a suitable finite abstraction of the original object, see [4]. Completeness in the equation system setting boils down to proving the truth of a particular variable, using a finite abstraction of the equation system, *i.e.*, using a Boolean equation system.

We first focus on the model checking problem for Kozen’s modal μ -calculus [15]. For this, we compare our framework to the framework of *generalised Kripke modal transition systems* (GTSs) [22, 7, 9], also known as *Disjunctive Transition Systems* [16]. These GTSs extend ordinary Kripke modal transition systems (MTSs) [10] with *must hyper-transitions*, *i.e.*, must transitions to sets of states. For the modal μ -calculus, we find that:

- Abstractions in our framework can be exponentially more succinct than those in the framework of GTSs;
- The GTS framework for abstraction and our abstraction framework are equally powerful for proving properties using finite abstractions; that is, they are as complete.

It is known that GTSs are complete for at least the least fixpoint-free fragment of the modal μ -calculus, see *e.g.* [6, 9]. We thus find that also our abstraction framework is complete for this fragment. The succinctness result is particularly interesting from a practical viewpoint, as space is often a limiting factor.

When we consider verification problems *beyond* the modal μ -calculus, such as, *e.g.*, the first-order modal μ -calculus model checking problem [12] and the behavioural equivalence checking problem [1], we obtain the following results:

- Our abstraction framework is complete for all verification problems that map to the least fixpoint-free fragment of equation systems.

- The abstraction framework is complete for all verification problems that map to universal quantification-free right-hand side, least fixpoint-only equation systems.

The first fragment includes the model checking problem for the least fixpoint-free fragment of the first-order modal μ -calculus, but also for its real-time extensions. Moreover, it also includes behavioural equivalence checking problems such as strong bisimilarity and similarity. Typical examples that fit in the second category of least fixpoint-only equation systems are reachability properties.

As a proof of concept of our abstraction theory for equation systems, we develop a transformation, which, given an equation system produces an abstraction thereof. The manipulation uses *homomorphisms* for abstraction, and is inspired by the abstraction techniques on Kripke structures, outlined by Clarke *et al.* in [2]. In a similar vein, one could implement predicate abstraction techniques. To the best of our knowledge, no homomorphism based tooling for abstraction exists that has the power of abstraction of our framework. We demonstrate the capabilities of the syntactic manipulation by reporting on a verification of Lamport’s Bakery Protocol and a small real-time system modelling a ball game.

Outline. In Section 2, we recall the basics of the framework of parameterised Boolean equation systems. Then, in Section 3, we introduce our abstraction framework. In Section 4, we compare our abstraction framework to the framework of GTSs. The syntactic transformation for equation systems and two example verifications conducted using these transformations are presented in Section 5. Related work is addressed in Section 6, and we finish with a brief discussion of future work. Detailed proofs are omitted from the current exposition but can be found in [3].

2 Preliminaries

We briefly review the theory of parameterised Boolean equation systems, or *equation systems* for short. For an in-depth treatment of this framework, including more elaborate examples of its use, we refer to [12, 14].

2.1 Data

We assume a theory of abstract data types for describing data and data transformations. This means that there are nonempty data sorts, which we generally denote using letters D , E and F . We have a set \mathcal{D} of *data variables*, with typical elements d, d_1, \dots , and we assume that there is some data language that is sufficiently rich to denote all relevant *data terms*. With every sort D we associate a semantic set \mathbb{D} . We use a *data environment* δ that assigns a value to the variables that can occur in the term. The interpretation of a data term t is given by $\delta(t)$, where δ is extended from a variable mapping to a mapping on terms in the standard way.

We assume the existence of a sort $B = \{\top, \perp\}$ representing the Booleans \mathbb{B} , and a sort $N = \{0, 1, \dots\}$ representing the natural numbers \mathbb{N} . If the context is clear, we write constants and operators in the syntactic domain the same as their semantic counterparts. For example, we have $\mathbb{B} = \{\top, \perp\}$, and the syntactic operator $_ \wedge _ : B \times B \rightarrow B$ corresponds to the usual, semantic conjunction $_ \wedge _ : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B}$.

2.2 Equation Systems.

A parameterised Boolean equation is a fixpoint equation that ranges over a formula; it is of the form $\sigma X(d_1:D_1, \dots, d_n:D_n) = \phi$. The right-hand side of the equation is a *predicate formula*. Its left-hand side consists of a *fixpoint symbol* $\sigma \in \{\mu, \nu\}$, where μ indicates a least fixpoint, and ν indicates a greatest fixpoint, and a (*sorted*) *predicate variable* X , taken from some sufficiently large set \mathcal{P} . The *domain* of a predicate variable $X:D_1 \times \dots \times D_n \rightarrow B$ is $D_1 \times \dots \times D_n$. In case of a nullary domain, *i.e.*, $X:B$, we say X is a *propositional variable* rather than a predicate variable. Apart from examples, we typically restrict ourselves (without loss of generality) to predicate variables X with a unary domain.

Definition 1. We denote the empty equation system by \emptyset . Equation systems \mathcal{E} and predicate formulae ϕ are defined through the following grammar:

$$\begin{aligned} \mathcal{E} &::= \emptyset \mid (\mu X(d:D) = \phi) \mathcal{E} \mid (\nu X(d:D) = \psi) \mathcal{E} \\ \phi, \psi &::= b \mid \phi \wedge \psi \mid \phi \vee \psi \mid \forall d:D. \phi \mid \exists d:D. \phi \mid X(e) \end{aligned}$$

Here, b is a Boolean data term, e a data expression, X a predicate variable and d a data term of sort D . In case an equation system is non-empty, we omit the trailing \emptyset .

Remark 1. We often write ϕ_X when we refer to the right-hand side of an equation for predicate variable X in an equation system. Likewise, we write d_X for the data variable occurring at the left-hand side of X 's equation, and D_X refers to the sort of d_X .

Throughout this paper, we only consider equation systems in which no two equations have the same left-hand side predicate variables. We require that the only data variables occurring freely in an equation's predicate formula are those listed in its left-hand side.

The left-hand side predicate variables of an equation system \mathcal{E} are collected in the set $\mathbf{bnd}(\mathcal{E})$; predicate variables occurring in the right-hand sides of the equation system are collected in the set $\mathbf{occ}(\mathcal{E})$. An equation system \mathcal{E} is *closed* if $\mathbf{occ}(\mathcal{E}) \subseteq \mathbf{bnd}(\mathcal{E})$, and \mathcal{E} is *open* otherwise.

A predicate formula is a *propositional formula* if it only contains $\wedge, \vee, \top, \perp$ and *propositional variables* X . The *size* of a propositional formula ϕ , denoted $|\phi|$, is defined as the number of propositional variable occurrences and operators it contains. We say that \mathcal{E} is a *Boolean equation system* [17] if all right-hand side formulae are propositional formulae and all variables are propositional variables.

The *size* of a Boolean equation system \mathcal{E} , denoted $|\mathcal{E}|$, is the sum of the sizes of all its right-hand side propositional formulae. Note that a Boolean equation system is always a *finite* equation system; that is, for all Boolean equation systems \mathcal{E} , $|\mathcal{E}| < \infty$.

Definition 2. *The semantics of a predicate formula ϕ is defined in the context of an environment δ , assigning meaning to data variables and an environment θ assigning a Boolean valued function to each predicate variable:*

$$\begin{aligned} \llbracket b \rrbracket \theta \delta &= \delta(b) \\ \llbracket X(e) \rrbracket \theta \delta &= \theta(X)(\delta(e)) \\ \llbracket \phi \wedge \psi \rrbracket \theta \delta &= \llbracket \phi \rrbracket \theta \delta \wedge \llbracket \psi \rrbracket \theta \delta \\ \llbracket \phi \vee \psi \rrbracket \theta \delta &= \llbracket \phi \rrbracket \theta \delta \vee \llbracket \psi \rrbracket \theta \delta \\ \llbracket \forall d:D. \phi \rrbracket \theta \delta &= \forall v \in \mathbb{D}. \llbracket \phi \rrbracket \theta \delta[v/d] \\ \llbracket \exists d:D. \phi \rrbracket \theta \delta &= \exists v \in \mathbb{D}. \llbracket \phi \rrbracket \theta \delta[v/d] \end{aligned}$$

Here, $\delta[v/x]$ overrides the environment δ by assigning value v to variable x .

A predicate formula ϕ can, semantically, be viewed as a function that assigns for a given data variable d of sort D , a truth value. That is, ϕ induces a Boolean functional $\lambda v \in \mathbb{D}. \llbracket \phi \rrbracket \theta(\delta[v/d])$. We denote the semantic functional obtained in this way by $\llbracket \phi_{\langle d \rangle} \rrbracket \theta \delta$. Similarly, given environments θ, δ and a predicate variable $X:D \rightarrow B$ a Boolean functional can be lifted to a predicate transformer as follows:

$$\lambda f \in \mathbb{B}^{\mathbb{D}}. (\llbracket \phi_{\langle d \rangle} \rrbracket (\theta[f/X]) \delta)$$

Such predicate transformers are monotonic over the complete lattice $(\mathbb{B}^{\mathbb{D}}, \sqsubseteq)$ of functions with domain \mathbb{D} and co-domain \mathbb{B} , ordered by \sqsubseteq , the point-wise lifting of implication [14]. As a result, least and greatest fixpoints are guaranteed to exist. We replace the λ in the above transformer with μ to denote the least solution to the transformer and ν to denote its greatest solution.

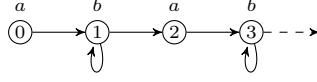
Definition 3. *The solution to an equation system \mathcal{E} , given a context of predicate and data environments θ, δ , is defined inductively as follows:*

$$\begin{aligned} \llbracket \emptyset \rrbracket \theta \delta &= \theta \\ \llbracket (\sigma X(d:D) = \phi) \mathcal{E} \rrbracket \theta \delta &= \llbracket \mathcal{E} \rrbracket \theta_X \delta, \end{aligned}$$

with $\theta_X = \theta[\sigma f \in \mathbb{B}^{\mathbb{D}}. \llbracket \phi_{\langle d \rangle} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta[f/X] \delta) \delta / X]$ $\sigma \in \{\mu, \nu\}$

For closed equation systems we have $\llbracket \mathcal{E} \rrbracket \eta \delta(X) = \llbracket \mathcal{E} \rrbracket \eta' \delta'(X)$, for arbitrary $\eta, \eta', \delta, \delta'$, and we therefore drop the environments from the semantic brackets for closed systems.

Example 1. Consider the (infinite state) Kripke structure \mathcal{K} depicted below and the modal μ -calculus formula ψ given by $\nu Z. a \wedge \Diamond \Diamond Z$, expressing that there is an infinite path on which on all even positions a holds (for the formal details on syntax and semantics, we refer to Section 4).



The equation system that encodes which states of the Kripke structure satisfy ψ (see, *e.g.* [13]) is as follows:

$$\begin{aligned}
 (\nu Z(n:N) = & \text{even}(n) \wedge \\
 & ((\text{even}(n) \wedge (Z(n+1) \vee Z(n+2))) \vee \\
 & (\text{odd}(n) \wedge (Z(n) \vee Z(n+1) \vee Z(n+2))))
 \end{aligned}$$

The equation system is obtained by combining property ψ with \mathcal{K} . Parameter n represents the states of \mathcal{K} and $\text{even}(n)$ holds exactly when property a holds. The encoding of the construct $\Diamond\Diamond X$ can be understood as follows: in even states, states $n+1$ and $n+2$ are reachable in exactly two steps; in odd states, states n , $n+1$ and $n+2$ are reachable in exactly two steps; for one of those states, property Z must hold again. The solution to the equation system is $\lambda n \in \mathbb{N}. \text{even}(n)$, which can be readily found by a standard fixpoint approximation. Computing the solution to $Z(0)$ by instantiating to a Boolean equation system, see [18], fails due to the presence of an infinite chain of dependencies:

$$(\nu Z_0 = Z_1 \vee Z_2) (\nu Z_1 = \perp) (\nu Z_2 = Z_3 \vee Z_4) \dots$$

Next, consider the alternating modal μ -calculus formula ϕ given by $\nu X. \mu Y. ((a \wedge \Diamond X) \vee (b \wedge \Diamond Y))$, which expresses that there is an infinite a, b -path on which a holds infinitely often. The equation system \mathcal{E} encoding which states of the Kripke structure satisfy ϕ is as follows:

$$\begin{aligned}
 (\nu X(n:N) &= Y(n)) \\
 (\mu Y(n:N) &= (\text{even}(n) \wedge X(n+1)) \vee \\
 & (\text{odd}(n) \wedge (Y(n) \vee Y(n+1))))
 \end{aligned}$$

The solution to X and Y is $\lambda n \in \mathbb{N}. \top$. Note that in this case, computing this solution using Def. 3 is rather involved (and not the point of the example). Attempting to compute the solution to $X(0)$ by instantiating the equation system to a Boolean equation system again fails due to the presence of an infinite chain of dependencies. Finally, note that swapping the equations in our example leads to solution $\lambda n \in \mathbb{N}. \perp$ for both X and Y , illustrating that the ordering of equations matters.

3 Abstraction via Consistent Consequence

Before we formally define the notion of abstraction for equation systems, we start with the following elementary observations. Suppose that for a closed equation system \mathcal{E} , with $X, X' \in \text{bnd}(\mathcal{E})$, we know that for $v, v' \in \mathbb{D}$, the Boolean value $\llbracket \mathcal{E} \rrbracket(X)(v)$ *under-approximates* $\llbracket \mathcal{E} \rrbracket(X')(v')$. If, moreover, $\llbracket \mathcal{E} \rrbracket(X)(v) = \top$, then also $\llbracket \mathcal{E} \rrbracket(X')(v') = \top$ and one can skip computing the solution to $X'(v')$.

Of course, the problem of computing the solution to $X'(v')$ in the above example is now delegated to establishing that $X(v)$ under-approximates $X'(v')$. In this section, we introduce the notion of a *consistent consequence*. Note that this notion was previously introduced in [8] for the fragment of Boolean equation systems. The absence of data in that setting greatly simplifies its formalisation; here, we generalise the definition to arbitrary equation systems. We proceed in small steps.

Definition 4. Let $X:D_X \rightarrow B$ be a predicate variable. We say that X 's signature, notation $\text{sig}(X)$, is the product $\{X\} \times \mathbb{D}_X$. We lift the notion of a signature of a predicate variable to sets of variables $P \subseteq \mathcal{P}$ in the natural way, i.e., $\text{sig}(P) = \bigcup_{X \in P} \text{sig}(X)$. Instead of $(X, v) \in \text{sig}(X)$, we prefer to write the more readable $X(v) \in \text{sig}(X)$ if no confusion is possible.

In addition, we require a definition of *rank* of a predicate variable. Intuitively, the rank of a predicate variable X is a measure for the nesting depth of X 's equation in the equation system.

Definition 5. Let \mathcal{E} be an equation system. The rank of a predicate variable $X \in \text{bnd}(\mathcal{E})$, denoted $\text{rank}_{\mathcal{E}}(X)$ is the smallest natural number coinciding with the number of (syntactic) least and greatest fixpoint alternations that precede X 's equation, counting from 1 if the first equation is a least fixpoint equation and 0 otherwise.

Example 2. Consider again the equation system \mathcal{E} of Example 1. Observe that the signatures of both X and Y are infinite: $\text{sig}(X) = \{(X, v) \mid v \in \mathbb{N}\}$ and $\text{sig}(Y) = \{(Y, v) \mid v \in \mathbb{N}\}$. Furthermore, we have $\text{rank}_{\mathcal{E}}(X) = 0$ and $\text{rank}_{\mathcal{E}}(Y) = 1$.

Definition 6. Let $R \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$ be a relation on signatures. We say that an environment θ is consistent with R if $X(v) R X'(v')$ implies that $\theta(X)(v) \Rightarrow \theta(X')(v')$. The set of all environments consistent with R is denoted Θ_R .

Next, we formalise the notion of a consistent consequence. We then proceed to show that the resulting preorder under-approximates (resp. over-approximates) the solution to an equation system, generalising the result of [8].

Definition 7. Let \mathcal{E} be an equation system. A relation $R \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$ is a consistent consequence on \mathcal{E} if for all $(\sigma X(d_X:D_X) = \phi_X)$ and $(\sigma' X'(d_{X'}:D_{X'}) = \phi_{X'})$ in \mathcal{E} for which $X(v) R X'(v')$ we have:

- $\text{rank}_{\mathcal{E}}(X) = \text{rank}_{\mathcal{E}}(X')$.
- for all $\theta \in \Theta_R$ and all δ we have $\llbracket \phi_X \rrbracket \theta \delta[v/d_X] \Rightarrow \llbracket \phi_{X'} \rrbracket \theta \delta[v'/d_{X'}]$.

We write $X(v) \leq X'(v')$ if there is a consistent consequence $R \subseteq \text{sig}(\text{bnd}(\mathcal{E})) \times \text{sig}(\text{bnd}(\mathcal{E}))$ on \mathcal{E} such that $X(v) R X'(v')$.

If $X(v) \leq X'(v')$ we say that $X'(v')$ is a consistent consequence of $X(v)$. The relation \leq is itself a consistent consequence for a given equation system, and it is the largest such relation, which immediately follows from the proposition below.

Proposition 1. *Let \mathcal{C} be a collection of consistent consequence relations on \mathcal{E} . Then $\bigcup \mathcal{C}$ is again a consistent consequence on \mathcal{E} .*

Proof. Suppose R, R' are both consistent consequences on \mathcal{E} . Assume $X(v) R \cup R' X'(v')$. Then either $X(v) R X'(v')$ or $X(v) R' X'(v')$. In both cases, we immediately find that $\text{rank}_{\mathcal{E}}(X) = \text{rank}_{\mathcal{E}}(X')$ if $X, X' \in \text{bnd}(\mathcal{E})$. Now, let $\theta \in \Theta_{R \cup R'}$. Then $\llbracket \phi_X \rrbracket \theta_{\mathcal{E}}[v/d_X] \Rightarrow \llbracket \phi_{X'} \rrbracket \theta_{\mathcal{E}}[v/d_X]$ follows from the observation that $\theta \in \Theta_R \cap \Theta_{R'}$ and the fact that the desired implication already holds for $\theta \in \Theta_R$ and $\theta \in \Theta_{R'}$. \square

The theorem below characterises the relation between a consistent consequence on a closed equation system, and the solution to the equation system.

Theorem 1. *Let \mathcal{E} be a closed equation system. Let $R \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$ be a consistent consequence on \mathcal{E} . We have:*

$$X(v) R X'(v') \text{ implies } (\llbracket \mathcal{E} \rrbracket)(X)(v) \Rightarrow (\llbracket \mathcal{E} \rrbracket)(X')(v')$$

Proof. See Appendix.

Note that the reverse of the above theorem does not necessarily hold: if $X(v)$ has a solution below that of $X'(v')$, then not necessarily $X(v) < X'(v')$.

We next lift the notion of a consistent consequence to a relation on signatures between different equation systems that are *compatible*. The compatibility requirement ensures that bound variables of both systems are disjoint and free variables in one system are not bound by the other. Formally, two equation systems $\mathcal{E}, \mathcal{E}'$ are *compatible* iff $\text{bnd}(\mathcal{E}) \cap \text{bnd}(\mathcal{E}') = \text{bnd}(\mathcal{E}) \cap \text{occ}(\mathcal{E}') = \text{bnd}(\mathcal{E}') \cap \text{occ}(\mathcal{E}) = \emptyset$. Two compatible equation systems can be *merged*, creating a single equation system containing blocks of equations that had equal rank in the original equation systems.

Definition 8. *Let $\mathcal{E}, \mathcal{E}'$ be compatible equation systems. A relation $R \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$ is a consistent consequence between \mathcal{E} and \mathcal{E}' if R is a consistent consequence on some equation system \mathcal{F} consisting of the equations of \mathcal{E} and \mathcal{E}' satisfying:*

- $\text{rank}_{\mathcal{F}}(Z_{\mathcal{E}}) = \text{rank}_{\mathcal{E}}(Z_{\mathcal{E}})$ for all $Z_{\mathcal{E}} \in \text{bnd}(\mathcal{E})$, and
- $\text{rank}_{\mathcal{F}}(Z_{\mathcal{E}'}) = \text{rank}_{\mathcal{E}'}(Z_{\mathcal{E}'})$ for all $Z_{\mathcal{E}'} \in \text{bnd}(\mathcal{E}')$.

We write $X(v) < X'(v')$ iff there is a consistent consequence $R \subseteq \text{sig}(\text{bnd}(\mathcal{E}) \cup \text{bnd}(\mathcal{E}')) \times \text{sig}(\text{bnd}(\mathcal{E}) \cup \text{bnd}(\mathcal{E}'))$ between \mathcal{E} and \mathcal{E}' such that $X(v) R X'(v')$.

Example 3. Reconsider the equation system \mathcal{E} of Example 1. As we concluded there, the solution to $X(v)$, for arbitrary v , cannot be computed using instantiation. Next, consider the following equation system:

$$\begin{aligned} (\nu \bar{X}(b:B) &= \bar{Y}(b)) \\ (\mu \bar{Y}(b:B) &= (b \wedge \bar{X}(\neg b)) \vee (\neg b \wedge (\bar{Y}(b) \vee \bar{Y}(\neg b)))) \end{aligned}$$

We find that $\bar{X}(\top) \leq X(v)$ for even v and $\bar{X}(\perp) \leq X(w)$ for odd w . That means that if we can prove that $\bar{X}(\top)$ has solution \top , we know that also $X(v)$ has solution \top for all even v . The solution to $\bar{X}(\top)$ is represented by the variable \bar{X}_\top in the Boolean equation system below, which is obtained by instantiating the above equation system:

$$(\nu \bar{X}_\top = \bar{Y}_\top) (\nu \bar{X}_\perp = \bar{Y}_\perp) (\mu \bar{Y}_\top = \bar{X}_\perp) (\mu \bar{Y}_\perp = \bar{Y}_\perp \vee \bar{Y}_\top)$$

The solution to all variables in the above Boolean equation system is \top . We thus have an effective way of proving that $X(v)$, for arbitrary v , is \top , too.

4 Relative Completeness and Succinctness

One may wonder how our denotational consistent consequence framework for abstraction compares to the more traditional operational frameworks of abstraction. We address this question in Section 4.2 for an important abstraction framework for the propositional modal μ -calculus. In Section 4.3, we answer a more general question by identifying fragments of equation systems for which our abstraction framework is *complete*.

4.1 Transition Systems

Let us recall the framework of generalised Kripke modal transition systems (GTSs) [16, 22, 7, 9]. These generalise Kripke structures. Let AP be a set of atomic propositions, and let $\text{Lit} = \text{AP} \cup \{\neg p \mid p \in \text{AP}\}$.

Definition 9. A generalised Kripke modal transition system (GTS) is a tuple $M = \langle S, R^+, R^-, L \rangle$ where:

- S is a set of states,
- $R^- \subseteq S \times S$ is the may transition relation; we require R^- to be total,
- $R^+ \subseteq S \times 2^S$ is the must transition relation; we require that $s R^+ A$ implies $s R^- t$ for all $t \in A$,
- $L: S \rightarrow 2^{\text{Lit}}$ is a labelling function; we require that $L(s)$ contains at most one of p and $\neg p$ for all $s \in S, p \in \text{AP}$.

We say that a GTS G is a Kripke structure if for all $s, s' \in S$ we have $s R^+ \{s'\}$ iff $s R^- s'$, and $s R^+ A$ implies $|A| \leq 1$.

Remark 2. We only deal with GTSs that can be described finitely using, e.g., first order logic. Note that this is a common restriction and, in practice, not a restriction at all.

The *size* of a GTS $M = \langle S, R^+, R^-, L \rangle$, denoted $|M|$ is defined as $|S| + |R^+| + |R^-|$. A GTS is an abstraction of a more concrete GTS (e.g., a concrete Kripke structure) if there is a simulation relation that relates the two GTSs. Such a relation links every transition the abstract GTS *must* make to some transition the concrete GTS *must* make, and *may* transition of the concrete GTS to a *may* transition of the abstract GTS.

Definition 10. Let $M_1 = \langle S_1, R_1^+, R_1^-, L_1 \rangle$ and $M_2 = \langle S_2, R_2^+, R_2^-, L_2 \rangle$ be two GTSs. A relation $H \subseteq S_1 \times S_2$ is a generalised mixed simulation if $s_1 H s_2$ implies

- $L_2(s_2) \subseteq L_1(s_1)$,
- if $s_1 R_1^- s'_1$ then there is some $s'_2 \in S_2$ such that $s_2 R_2^- s'_2$ and $s'_1 H s'_2$,
- if $s_2 R_2^+ A_2$ then there is some $A_1 \subseteq S_1$ such that $s_1 R_1^+ A_1$ and for every $s'_1 \in A_1$ there exists some $s'_2 \in A_2$ such that $s'_1 H s'_2$.

We write $\langle M_1, s_1 \rangle \leq \langle M_2, s_2 \rangle$ if $s_1 H s_2$ for some generalised mixed simulation H .

The propositional modal μ -calculus, a highly expressive modal logic, can be interpreted over GTSs. Unlike the traditional setting where formulae either hold of a system or not, there is a third option in the GTS setting, in which the formula is neither true nor false for the GTS; this basically means that the truth of the formula cannot be determined. We here adopt the notation and semantics introduced by Shoham and Grumberg [22].

Definition 11. A μ -calculus formula (in positive form) is a formula generated by the following grammar:

$$f, g ::= \top \mid \perp \mid l \mid X \mid f \wedge g \mid f \vee g \mid \Box f \mid \Diamond f \mid \nu X. f \mid \mu X. f$$

where $l \in \text{Lit}$ and $X \in \mathcal{V}$ for a set of propositional variables \mathcal{V} . The semantics of a formula f is defined by two inductively defined functions $\llbracket \cdot \rrbracket_{\sqcup}$ and $\llbracket \cdot \rrbracket_{\sqcap}$, in the context of a GTS $M = \langle S, R^+, R^-, L \rangle$ and an environment $\rho: \mathcal{V} \rightarrow 2^S$:

f	$\llbracket f \rrbracket_{\sqcap}^{\rho}$	$\llbracket f \rrbracket_{\sqcup}^{\rho}$
\top	S	\emptyset
\perp	\emptyset	S
l	$\{s \in S \mid l \in L(s)\}$	$\{s \in S \mid \neg l \in L(s)\}$
X	$\rho(X)$	$\rho(X)$
$f \wedge g$	$\llbracket f \rrbracket_{\sqcap}^{\rho} \cap \llbracket g \rrbracket_{\sqcap}^{\rho}$	$\llbracket f \rrbracket_{\sqcup}^{\rho} \cup \llbracket g \rrbracket_{\sqcup}^{\rho}$
$f \vee g$	$\llbracket f \rrbracket_{\sqcap}^{\rho} \cup \llbracket g \rrbracket_{\sqcap}^{\rho}$	$\llbracket f \rrbracket_{\sqcup}^{\rho} \cap \llbracket g \rrbracket_{\sqcup}^{\rho}$
$\Diamond f$	$\tilde{\Diamond} \llbracket f \rrbracket_{\sqcap}^{\rho}$	$\Box \llbracket f \rrbracket_{\sqcup}^{\rho}$
$\Box f$	$\tilde{\Box} \llbracket f \rrbracket_{\sqcap}^{\rho}$	$\tilde{\Diamond} \llbracket f \rrbracket_{\sqcup}^{\rho}$
$\mu X. f$	$\mu U. \llbracket f \rrbracket_{\sqcap}^{\rho[X:=U]}$	$\nu U. \llbracket f \rrbracket_{\sqcup}^{\rho[X:=U]}$
$\nu X. f$	$\nu U. \llbracket f \rrbracket_{\sqcap}^{\rho[X:=U]}$	$\mu U. \llbracket f \rrbracket_{\sqcup}^{\rho[X:=U]}$

Here, we used the following abbreviations: $\tilde{\Diamond}U = \{s \mid \exists A \subseteq S : s R^+ A \wedge A \subseteq U\}$ and $\tilde{\Box}U = \{s \mid \forall t \in S : s R^- t \Rightarrow t \in U\}$. In case the formula f is closed (i.e., every propositional variable occurs within the scope of a fixpoint binding it), its semantics is independent of the environment ρ , and we drop ρ from the semantic brackets. A closed formula f is true in a state $s \in S$, denoted $M, s \models f$ if $s \in \llbracket f \rrbracket_{\sqcap}$; f is false in s , denoted $M, s \not\models f$ if $s \in \llbracket f \rrbracket_{\sqcup}$ and it is unknown in s otherwise.

Table 1. Encodings of the model checking problems into the corresponding equation systems $\mathbf{E}_G^\square(f)$ and $\mathbf{E}_G^\sqcup(f)$.

$\mathbf{E}_G^m(b)$	$\triangleq \emptyset$
$\mathbf{E}_G^m(X)$	$\triangleq \emptyset$
$\mathbf{E}_G^m(f \wedge g)$	$\triangleq \mathbf{E}_G^m(f) \mathbf{E}_G^m(g)$
$\mathbf{E}_G^m(f \vee g)$	$\triangleq \mathbf{E}_G^m(f) \mathbf{E}_G^m(g)$
$\mathbf{E}_G^m(\star f)$	$\triangleq \mathbf{E}_G^m(f)$
$\mathbf{E}_G^m(\sigma X.f)$	$\triangleq (\sigma X^m(d : \mathcal{S}) = \mathbf{RHS}_G^m(f)) \mathbf{E}_G^m(f)$
$\mathbf{RHS}_G^m(X)$	$\triangleq X^m(d)$
$\mathbf{RHS}_G^m(\sigma X.f)$	$\triangleq X^m(d)$
$\mathbf{RHS}_G^\square(b)$	$\triangleq b$
$\mathbf{RHS}_G^\square(l)$	$\triangleq l \in \mathcal{L}(d)$
$\mathbf{RHS}_G^\square(f \wedge g)$	$\triangleq \mathbf{RHS}_G^\square(f) \wedge \mathbf{RHS}_G^\square(g)$
$\mathbf{RHS}_G^\square(f \vee g)$	$\triangleq \mathbf{RHS}_G^\square(f) \vee \mathbf{RHS}_G^\square(g)$
$\mathbf{RHS}_G^\square(\diamond f)$	$\triangleq \exists A:2^D. \mathcal{R}^+(d, A) \wedge$ $\forall d':D. d' \in A \Rightarrow ((\mathbf{RHS}_G^\square(f))[d'/d])$
$\mathbf{RHS}_G^\square(\square f)$	$\triangleq \forall d':D. \mathcal{R}^-(d, d') \Rightarrow ((\mathbf{RHS}_G^\square(f))[d'/d])$
$\mathbf{RHS}_G^\sqcup(b)$	$\triangleq \neg b$
$\mathbf{RHS}_G^\sqcup(l)$	$\triangleq l \notin \mathcal{L}(d)$
$\mathbf{RHS}_G^\sqcup(f \wedge g)$	$\triangleq \mathbf{RHS}_G^\sqcup(f) \vee \mathbf{RHS}_G^\sqcup(g)$
$\mathbf{RHS}_G^\sqcup(f \vee g)$	$\triangleq \mathbf{RHS}_G^\sqcup(f) \wedge \mathbf{RHS}_G^\sqcup(g)$
$\mathbf{RHS}_G^\sqcup(\diamond f)$	$\triangleq \forall d':D. \mathcal{R}^-(d, d') \Rightarrow ((\mathbf{RHS}_G^\sqcup(f))[d'/d])$
$\mathbf{RHS}_G^\sqcup(\square f)$	$\triangleq \exists A:2^D. \mathcal{R}^+(d, A) \wedge$ $\forall d':D. d' \in A \Rightarrow ((\mathbf{RHS}_G^\sqcup(f))[d'/d])$

Note that if M is a Kripke structure, then it always holds that either $M, s \models f$ or $M, s \not\models f$, and the satisfaction relation coincides with the usual semantics for the μ -calculus. For the purpose of comparing the GTS framework with our consistent consequence framework in the next subsection, we present the formal encoding of the model checking problem for GTSs into equation systems below.

Definition 12. Let $M = \langle S, R^+, R^-, L \rangle$ be a GTS and let f be a closed μ -calculus formula. The schemes for encoding the satisfaction and refutation model checking problems into equation systems are given in Table 1, in which $\mathbf{E}_G^\square(f)$ encodes the satisfaction problem, and $\mathbf{E}_G^\sqcup(f)$ the refutation problem. The sort \mathcal{S} represents the set S of the GTS G , and $\mathcal{R}^-, \mathcal{R}^+$ and \mathcal{L} are predicates describing the semantic artefacts of G , viz. R^-, R^+ and L . We use $m \in \{\square, \sqcup\}$, $b \in \{\top, \perp\}$, $X \in \mathcal{V}$, $X^\square, X^\sqcup \in \mathcal{P}$ and $\star \in \{\diamond, \square\}$.

For the ease of readability, we often assume that the outermost symbol in the formula is the fixpoint definition, i.e. we consider the *guarded form* $\sigma X.f$. If this is the case, then given a state s of a GTS G , the corresponding element of

the signature in the equation system encoding is $X^\sqcap(s)$ (satisfaction) or $X^\sqcup(s)$ (refutation).

Proposition 2. *The encodings $\mathbf{E}_M^\sqcap(f)$ and $\mathbf{E}_M^\sqcup(f)$ are sound for the verification resp. refutation model checking problems, that is:*

$$\begin{aligned} \llbracket \mathbf{E}_M^\sqcap(\sigma X.f) \rrbracket(X^\sqcap)(s) = \top &\Rightarrow \langle M, s \rangle \models \sigma X.f \\ \llbracket \mathbf{E}_M^\sqcup(\sigma X.f) \rrbracket(X^\sqcup)(s) = \top &\Rightarrow \langle M, s \rangle \not\models \sigma X.f \end{aligned}$$

In case M is a Kripke structure, the encodings are sound and complete:

$$\begin{aligned} \llbracket \mathbf{E}_M^\sqcap(\sigma X.f) \rrbracket(X^\sqcap)(s) = \top &\Leftrightarrow \langle M, s \rangle \models \sigma X.f \\ \llbracket \mathbf{E}_M^\sqcup(\sigma X.f) \rrbracket(X^\sqcup)(s) = \top &\Leftrightarrow \langle M, s \rangle \not\models \sigma X.f \end{aligned}$$

For a Kripke structure M , we only need the satisfaction encoding, which we denote with $\mathbf{E}_M(f)$. Note that the transformation $\mathbf{E}_M(f)$ is easily automated, even for richer logics such as the first-order modal μ -calculus. An implementation can be found in μCRL and mCRL2 , see *e.g.* [14].

4.2 Succinctness and Relative Completeness

The power of an abstraction framework for the modal μ -calculus is often measured as the degree to which the framework is able to prove μ -calculus formulae that hold of an infinite system by means of a finite abstraction of the infinite system. This is known as *completeness*, see, *e.g.* [4].

Definition 13. *Let F be a collection of modal μ -calculus formulae. GTSs with \leq are complete for F , if, for all formulae $f \in F$ and all Kripke structures $M_1, s_1 \models f$, there is a finite GTS M_2, s_2 such that $\langle M_1, s_1 \rangle \leq \langle M_2, s_2 \rangle$ and $M_2, s_2 \models f$.*

Next, we introduce a corresponding notion of completeness for equation systems.

Definition 14. *We say that the framework of equation systems with \prec is complete for a class of closed equation systems \mathcal{C} , if for each $\mathcal{E} \in \mathcal{C}$ and all $X(v) \in \text{sig}(\text{bnd}(\mathcal{E}))$ satisfying $\llbracket \mathcal{E} \rrbracket(X)(v) = \top$, there is a Boolean equation system $\bar{\mathcal{E}}$ and $\bar{X} \in \text{bnd}(\bar{\mathcal{E}})$ such that $\bar{X} \prec X(v)$ and $\llbracket \bar{\mathcal{E}} \rrbracket(\bar{X}) = \top$.*

We now show that for model checking μ -calculus formulae, the equation systems with consistent consequence framework and the GTSs with generalised mixed simulation framework are *equally complete*. That is, both are equally powerful for using finite abstractions for the μ -calculus. Formally, this is expressed by the following theorem.

Theorem 2. *Let F be a collection of guarded, closed μ -calculus formulae. Then GTSs with \leq are complete for F iff equation systems with \prec are complete for $\{\mathbf{E}_M(f) \mid \text{Kripke Structure } M \text{ and } f \in F\}$.*

Proof. Here we only sketch the key ideas, we refer to the Appendix for details.

The proof from left to right follows from Proposition 2 and by showing that a generalised mixed simulation relation induces a consistent consequence on the resulting Boolean equation systems and $\mathbf{E}_{M_1}(\sigma X.f)$.

The proof in the other direction is technically quite involved. We first observe that the consistent consequence preserves the boundedness of the μ -dominated chains of dependencies in \mathcal{E} in $\mathbf{E}_{M_1}(\sigma X.f)$. Next, one can show that there is a stronger least fixpoint-free formula $\sigma \bar{X}.f$ that can be obtained from $\sigma X.f$ by under-approximating the least fixpoint-formulae in $\sigma X.f$, by unfolding every least fixpoint subformula at most $|\mathcal{E}|$ times. The boundedness of the μ -dominated chains in $\mathbf{E}_{M_1}(\sigma X.f)$ can be used to establish a correspondence between the equation systems $\mathbf{E}_{M_1}(\sigma X.f)$ and $\mathbf{E}_{M_1}(\sigma \bar{X}.f)$ and their solutions, and, in particular, show that $\bar{X}^\square(s_1) = \top$. Since GTSs are complete for the least fixpoint-free fragment of the μ -calculus, see, *e.g.* [22], and $\sigma \bar{X}.f$ is (semantically) least fixpoint-free.

Observe that contrary to GTSs, *Kripke modal transition systems*, which are another generalisation of Kripke structures, often used for abstraction, are known to be strictly less complete than GTSs for abstraction for the modal μ -calculus. This follows from the incompleteness example for MTSs in [4] and the fact that GTSs are complete for least fixpoint-free fragment of the μ -calculus [6, 9]. As a result, we find that our abstraction framework for equation systems is strictly more complete than such abstraction frameworks, too.

We next focus on the *succinctness* of the abstract objects that can be used for proving properties. The lemma below states that the size of our “abstract” equation systems is at most linear in the size of the formula and the abstract GTSs capable of proving that formula.

Lemma 1. *Let $\sigma X.f$ be a closed μ -calculus formula and let $M_1 = \langle S_1, R_1^+, R_1^-, L_1 \rangle$ be a Kripke structure. Suppose there is a finite GTS $M_2 = \langle S_2, R_2^+, R_2^-, L_2 \rangle$ such that $\langle M_1, s_1 \rangle \leq \langle M_2, s_2 \rangle$ and $M_2, s_2 \models \sigma X.f$. Then there is a closed Boolean equation system $\bar{\mathcal{E}}$ with at most $|\sigma X.f| \times |S_2|$ equations and size at most $\mathcal{O}(|\sigma X.f| \times |M_2|)$ such that $\bar{X} \triangleleft X^\square(s_1)$ and $\llbracket \bar{\mathcal{E}} \rrbracket(\bar{X}) = \top$.*

Proof. The model checking problem $M_2, s_2 \models \sigma X.f$ is encoded by the equation system $\mathbf{E}_{M_2}^\square(\sigma X.f)$. Since M_2 is finite, instantiating $\mathbf{E}_{M_2}^\square(\sigma X.f)$ yields a Boolean equation system of size at most $\mathcal{O}(|\sigma X.f| \times |M_2|)$. We denote this Boolean equation system by $\bar{\mathcal{E}}$, and we assume that the solution to $X^\square(s_2)$ in $\mathbf{E}_{M_2}^\square(\sigma X.f)$ is encoded by the variable \bar{X} in $\bar{\mathcal{E}}$. Instantiation ensures in particular that $\bar{X} \triangleleft X^\square(s_2)$.

It remains to show that $X^\square(s_1)$ is a consistent consequence of \bar{X} . This follows from the fact that the generalised mixed simulation relation between $\langle M_1, s_1 \rangle$ and $\langle M_2, s_2 \rangle$ induces a consistent consequence between $\mathbf{E}_{M_1}^\square(\sigma X.f)$ and $\mathbf{E}_{M_2}^\square(\sigma X.f)$ such that $X^\square(s_2) \triangleleft X^\square(s_1)$. From transitivity of \triangleleft , we obtain $\bar{X} \triangleleft X^\square(s_1)$.

Note that the above lemma provides an upper bound on the size of our abstract objects. In fact, we can tighten these bounds significantly, using the following lemma.

Lemma 2. *Let \mathcal{E} be a closed Boolean equation system consisting of n equations. Suppose we have $\llbracket \mathcal{E} \rrbracket(X) = \top$. Then there is a Boolean equation system \mathcal{E}^\square of size $\mathcal{O}(n^2)$ such that $X^\square \leq X$ and $\llbracket \mathcal{E}^\square \rrbracket(X^\square) = \top$.*

Proof. Following [17, Proposition 3.36], each right-hand side in \mathcal{E} can be strengthened by (non-deterministically) selecting one predicate variable in each disjunction over predicate variables, without changing the solution to X in \mathcal{E} . In the resulting equation system, rename all predicate variables Y to Y^\square . Observe that each right-hand side contains no proper disjunctions and can therefore be rewritten to a logically equivalent conjunctive formula of size at most n . Let \mathcal{E}^\square be the resulting equation system. We then have $|\mathcal{E}^\square|$ is $\mathcal{O}(n^2)$ and, by construction, $X^\square \leq X$ and $\llbracket \mathcal{E}^\square \rrbracket(X^\square) = \top$.

The above lemmata together give us the following theorem.

Theorem 3. *Let $\sigma X.f$ be a closed formula and let $M_1 = \langle S_1, R_1^+, R_1^-, L_1 \rangle$ be a Kripke structure. If there is a finite GTS $M_2 = \langle S_2, R_2^+, R_2^-, L_2 \rangle$ such that $\langle M_1, s_1 \rangle \leq \langle M_2, s_2 \rangle$ and $M_2, s_2 \models \sigma X.f$, then there is a closed Boolean equation system $\bar{\mathcal{E}}$ of size at most $\mathcal{O}(|\sigma X.f| \times |S_2|^2)$ such that $\bar{X} \leq X^\square(s_1)$ and $\llbracket \bar{\mathcal{E}} \rrbracket(\bar{X}) = \top$.*

It is rather straightforward to construct examples that demonstrate that our framework can indeed be more succinct. The example below is a point in case.

Example 4. Let $A \neq \emptyset$ be a finite index set and let $\text{AP} = \{f_i, a_i \mid i \in A\}$. Consider the Kripke structure \mathcal{K} with the set of states 2^A and transitions and labellings given by:

- $\zeta \rightarrow \zeta \cup \{i\}$ for all $i \in A \setminus \zeta$, and $\zeta \rightarrow \zeta \setminus \{i\}$ for all $i \in \zeta$ and all $\zeta \subseteq A$;
- $L(\zeta) = \{a_i \mid i \in \zeta\} \cup \{f_i \mid i \in A \setminus \zeta\}$ for all $\zeta \subseteq A$.

Next, consider the μ -calculus formula g defined as $\nu X. \bigwedge_{i \in A} (a_i \vee f_i) \wedge \Box X$; note that g trivially holds of \mathcal{K} . Observe that there is no GTS M , smaller than \mathcal{K} that simulates \mathcal{K} and is capable of proving the property. Hence, the size of the smallest GTS capable of proving g is of size $\mathcal{O}(2^{|A|})$. Also, observe that the equation system $\mathbf{E}_{\mathcal{K}}(g)$ is as follows:

$$\begin{aligned} \nu X(\zeta; 2^A) = & \bigwedge_{i \in A} (i \in \zeta \vee i \in A \setminus \zeta) \wedge \\ & \bigwedge_{j \in A \setminus \zeta} X(\zeta \cup \{j\}) \wedge \bigwedge_{k \in \zeta} X(\zeta \setminus \{k\}) \end{aligned}$$

The smallest Boolean equation system of which the above equation system is a consistent consequence is of size $\mathcal{O}(1)$: we have $\bar{X} \leq X(\emptyset)$ for Boolean equation system $\nu \bar{X} = \bar{X}$.

4.3 Completeness for Fragments of Equation Systems

In the previous section, we focused on the power of our abstraction framework for model checking of the propositional modal μ -calculus. However, equation systems are not restricted to a single type of verification problem. We therefore investigate the notion of completeness in equation systems in isolation of the encoded verification problems.

The theorem below states that our framework is complete for the least fixpoint-free fragment of equation systems.

Theorem 4. *Equation systems with \leq are complete for the class of closed least fixpoint-free equation systems.*

Proof. Due to [17, Theorem 9.4], we know that there is an equation system \mathcal{E}^\square , obtained by replacing all equations of the form $\nu X(d:D) = \phi$ in \mathcal{E} by $\nu X^\square(d:D) = \psi^\square$, where ψ and ψ^\square are such that:

- ψ is conjunctive and for all η, δ , $\llbracket \psi \rrbracket \eta \delta$ implies $\llbracket \phi \rrbracket \eta \delta$,
- ψ^\square is obtained from ψ by replacing each variable $Y \in \text{occ}(\psi)$ with Y^\square ,
- $\llbracket \mathcal{E}^\square \rrbracket (X^\square)(v) = \top$.

Observe that, by construction, we have $X^\square(v) \leq X(v)$. Next, note that \mathcal{E} is a consistent consequence of any equation system $\bar{\mathcal{E}}$ of which \mathcal{E}^\square is a consistent consequence. In particular, the finite equation system $\nu \bar{X} = \bar{X}$ is such that $\bar{X} \leq X^\square(v)$. The theorem then follows from transitivity of \leq .

As a result of this theorem, we find that our abstraction framework is complete for a large set of verification problems, as stated by the below corollary.

Corollary 1. *The consistent consequence abstraction framework of equation systems is complete for the least fixpoint-free fragment of the first-order extensions of the modal μ -calculus [12, 14], the problem of deciding strong bisimilarity and similarity between two infinite processes [1] and the least fixpoint-free fragments of real-time extensions of the (first-order) modal μ -calculus, see e.g. [23].*

Example 5. Reconsider Example 4, and assume that the set A can be infinite. Generalise the formula g of that example to the following first-order μ -calculus formula (see e.g. [12] for an action-based variant): $\nu X.(\forall i \in A. f_i \vee a_i) \wedge \Box X$. Then there is no finite GTS capable of proving g for the infinite Kripke structure \mathcal{K} . Note that this illustrates the fact that our abstraction framework is complete for elementary first order extensions of the least fixpoint-free fragment of the μ -calculus, in contrast to the GTS framework.

Theorem 5. *Equation systems with \leq are complete for the class of closed least fixpoint-only equation systems with right-hand side formulae containing no universal quantifications.*

Proof. Let \mathcal{E} be an equation system of the required form. Assume that $\llbracket \mathcal{E} \rrbracket(X)(v) = \top$. Observe that, due to their shape, each right-hand side can be brought into a logically equivalent disjunctive normal form with existential quantification prefixes. Consider the (infinite) equation system one can obtain using instantiation of \mathcal{E} starting in $X(v)$, using *e.g.* [18]; the resulting infinite system \mathcal{E}_∞ is in disjunctive normal form. Assume that variable X_v in $\text{bnd}(\mathcal{E}_\infty)$ represents the solution to $X(v)$ in \mathcal{E} , which, by assumption, is \top .

Again, due to [17, Theorem 9.4], we know that there is an equation system $\bar{\mathcal{E}}_\infty$ with the same solution as \mathcal{E}_∞ , which can be obtained from \mathcal{E}_∞ by selecting exactly one clause among the disjunctions in each right-hand side. Next, observe that since variable X_v has solution \top and $\bar{\mathcal{E}}_\infty$ is least fixpoint-only, there must be a well-founded descending path through equations in $\bar{\mathcal{E}}_\infty$. Consider the finite sub equation system $\bar{\mathcal{E}}_\infty^\square$ of $\bar{\mathcal{E}}_\infty$ containing only those equations on the well-founded path from X_v . It follows by construction of $\bar{\mathcal{E}}_\infty^\square$ that $X_v < X(v)$, which is what we needed to show.

By combining the above result and Theorem 2 it follows that also GTSs with a finite set of initial states are complete for the disjunctive, least fixpoint-only fragment of the modal μ -calculus.

Corollary 2. *GTSs with \leq are complete for the set of least fixpoint-only modal μ -calculus formula containing no \square modalities.*

5 Abstraction Through Syntactic Manipulations

The problem of checking whether there is a consistent consequence between signatures of two Boolean equation systems is coNP-complete already [8]. We show that our abstraction framework can nevertheless lead to effective tooling. For this, we develop a syntax-based transformation—inspired by [2]—that, semantically, provably approximates $<$, and we briefly demonstrate its potentials using two case studies. In a similar vein, our abstraction framework can be combined with predicate abstraction.

5.1 Existential and Universal Abstractions

Consider a data sort D that represents a complex or infinite semantic set (*e.g.* the natural numbers). Using a *homomorphism*, all elements, operators and relations of the domain D can be mapped to a corresponding simpler data structure \hat{D} . Formally, an abstract interpretation of the concrete elements in a domain D is given by a surjective mapping $h_D: D \rightarrow \hat{D}$, the so-called *abstraction function*. In addition, every operator and relation on the concrete data domain D is assumed to have a corresponding abstract operation. Note that the result of an abstract operation is no longer unique, as two different concrete elements may map to the same abstract element.

A standard solution to the non-unicity problem is to lift the codomain of a concrete operation or relation $f: D_1 \times \dots \times D_n \rightarrow D$ to a set when defining its

abstract counterpart $\widehat{f}:\widehat{D}_1 \times \dots \times \widehat{D}_n \rightarrow 2^{\widehat{D}}$. For reasons of consistency, also the domains of the abstract functions and relations \widehat{f} are lifted to sets for all $A_j \subseteq \widehat{D}_j, 1 \leq j \leq n$:

$$\widehat{f}(A_1, \dots, A_n) = \bigcup_{a_1 \in A_1, \dots, a_n \in A_n} \widehat{f}(a_1, \dots, a_n)$$

It is then possible to define an abstraction operator $\widehat{\cdot}$ that converts a term t of sort D to its corresponding sort \widehat{D} . Assuming the following grammar for our data terms:

$$t ::= d \mid c \mid f(t_1, \dots, t_n)$$

where d is a data variable, c is a closed term and f is an n -ary operator or relation. Assuming, that we have an abstraction function h , we can define the operator $\widehat{\cdot}$ as follows:

$$\widehat{d} = \{\bar{d}\} \quad \widehat{c} = \{h(c)\} \quad \widehat{f(t_1, \dots, t_n)} = \widehat{f}(\widehat{t_1}, \dots, \widehat{t_n})$$

Here, we introduce a fresh variable \bar{d} for every variable d of some concrete sort D . Computations in the abstract domain using the abstract operators are sensible as long as the *safety condition* holds for all terms t with free variables d_1, \dots, d_n , and all closed terms c_1, \dots, c_n : $\llbracket h(t[c_1/d_1, \dots, c_n/d_n]) \rrbracket \in \llbracket \widehat{t}[h(c_1)/\bar{d}_1, \dots, h(c_n)/\bar{d}_n] \rrbracket$. Essentially, this condition ensures that \widehat{t} always represents a set of concrete values that includes t . This condition is met whenever for all $f:D_1 \times \dots \times D_n \rightarrow E$:

$$\forall d_1 \in D_1, \dots, d_n \in D_n. \\ h_E(f(d_1, \dots, d_n)) \in \widehat{f}(\{h_{D_1}(d_1)\}, \dots, \{h_{D_n}(d_n)\})$$

Given a set of abstraction functions for a set of concrete domains, we can use these, together with the trivial abstraction (identity) functions for those domains not equipped with a user-defined abstraction function, to strengthen or weaken the predicate formulae we encounter in a given equation system. For simplicity, and without loss of generality, we assume that all concrete sorts in an equation system, with the exception of the Boolean sort, have a homomorphism h mapping these sorts to some abstract domain.

Definition 15. Let ϕ be an arbitrary predicate formula. We inductively define the under-approximation $\mathcal{A}_\sqcap(\phi)$ (resp. the over-approximation $\mathcal{A}_\sqcup(\phi)$) as follows:

$$\mathcal{A}_m(b) = \begin{cases} \exists v:B. v \in \widehat{b} \wedge v & \text{if } m = \sqcup \\ \forall v:B. v \in \widehat{b} \Rightarrow v & \text{otherwise} \end{cases}$$

$$\mathcal{A}_m(X(e)) = \begin{cases} \exists v:\widehat{D_X}. v \in \widehat{e} \wedge \widehat{X}_\sqcup(v) & \text{if } m = \sqcup \\ \forall v:\widehat{D_X}. v \in \widehat{e} \Rightarrow \widehat{X}_\sqcap(v) & \text{otherwise} \end{cases}$$

$$\mathcal{A}_m(\phi \oplus \psi) = \mathcal{A}_m(\phi) \oplus \mathcal{A}_m(\psi) \quad \text{for } \oplus \in \{\wedge, \vee\}$$

$$\mathcal{A}_m(Q d_1:D. \phi) = Q \bar{d}_1:\widehat{D}. \mathcal{A}_m(\phi) \quad \text{for } Q \in \{\forall, \exists\}$$

Note that, by definition of $\hat{\succsim}$, the predicate formula $\mathcal{A}_m(\phi)$ contains data variables \bar{d} for the variables d in ϕ , and predicate variables \hat{X}_m in $\mathcal{A}_m(\phi)$ for X in ϕ . We extend the above operator $\mathcal{A}_m(-)$ to equation systems in the natural way.

Definition 16. Let \mathcal{E} be an equation system. The abstraction operator $\mathcal{A}_m(-)$ for equation systems, where $m \in \{\sqcup, \sqcap\}$, is defined inductively as follows:

$$\begin{aligned}\mathcal{A}_m(\emptyset) &= \emptyset \\ \mathcal{A}_m((\sigma X(d:D) = \phi) \mathcal{E}) &= (\sigma \hat{X}_m(\bar{d}:\hat{D}) = \mathcal{A}_m(\phi)) \mathcal{A}_m(\mathcal{E})\end{aligned}$$

The following theorem states that we have achieved an under-approximation (resp. over-approximation) of the solution to the concrete equation system.

Theorem 6. Let \mathcal{E} be an arbitrary closed equation system. Then for all $X \in \text{bnd}(\mathcal{E})$ and all closed terms $v:D_X$, we have:

$$\begin{aligned}\llbracket \mathcal{A}_{\sqcap}(\mathcal{E}) \rrbracket(\hat{X}_{\sqcap})(\llbracket h(v) \rrbracket) &\Rightarrow \llbracket \mathcal{E} \rrbracket(X)(\llbracket v \rrbracket) \\ &\Rightarrow \llbracket \mathcal{A}_{\sqcup}(\mathcal{E}) \rrbracket(\hat{X}_{\sqcup})(\llbracket h(v) \rrbracket)\end{aligned}$$

Proof. The theorem follows essentially from Theorem 1 if $\hat{X}_{\sqcap}(\llbracket h(v) \rrbracket) \prec X(\llbracket v \rrbracket) \prec \hat{X}_{\sqcup}(\llbracket h(v) \rrbracket)$. The latter statement follows from the following observation.

Let $R \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$ be defined as $\hat{X}_{\sqcap}(\llbracket h(v) \rrbracket) R X(v)$ and $X(v) R \hat{X}_{\sqcup}(\llbracket h(v) \rrbracket)$, for closed terms $v:D_X$. Then, for all $\theta \in \Theta_R$ and all data environments δ , we have:

$$\begin{aligned}\llbracket \mathcal{A}_{\sqcap}(\phi) \rrbracket \theta \delta[\llbracket h(v) \rrbracket / \bar{d}] &\Rightarrow \llbracket \phi \rrbracket \theta \delta[\llbracket v \rrbracket / d] \\ &\Rightarrow \llbracket \mathcal{A}_{\sqcup}(\phi) \rrbracket \theta \delta[\llbracket h(v) \rrbracket / \bar{d}]\end{aligned}$$

The proof thereof follows using a structural induction. See Appendix for details.

Example 6. Consider the infinite Kripke structure \mathcal{K} depicted left below.



Assume that the states of the Kripke structure are represented by the Cartesian product $B \times N$. The equation system encoding $\mathcal{K} \models \nu X. (\Diamond(b \vee a) \vee \neg b) \wedge \Box X$ is as follows:

$$\begin{aligned}\nu X(c:B, n:N) &= (\neg c \vee (c \wedge n > 0) \vee (c \wedge n = 0)) \wedge \\ &\quad (\forall m:N. \neg c \Rightarrow X(\top, m)) \wedge \\ &\quad (c \wedge n > 0 \Rightarrow X(c, n-1)) \wedge \\ &\quad (c \wedge n = 0 \Rightarrow X(c, n))\end{aligned}$$

Through abstraction using a mapping $h_N: N \rightarrow B$, defined as $h_N(n) = (n = 0)$, we obtain, after logical simplification, the equation system depicted below:

$$\begin{aligned} \nu \hat{X}_\square(c, \bar{n}:B) = & (\neg c \vee (c \wedge \neg \bar{n}) \vee (c \wedge \bar{n})) \wedge \\ & (\forall \bar{m}:B. \neg c \Rightarrow \hat{X}_\square(\top, \bar{m})) \wedge \\ & (c \wedge \neg \bar{n} \Rightarrow (\hat{X}_\square(c, \top) \wedge \hat{X}_\square(c, \perp))) \wedge \\ & (c \wedge \bar{n} \Rightarrow \hat{X}_\square(c, \bar{n})) \end{aligned}$$

We observe that $\hat{X}_\square(c, \bar{n}) = \top$ for all c, \bar{n} , and, as a result, $X(c, n) = \top$ for all c, n . A GTS that simulates \mathcal{K} and can prove the same property is depicted next to the Kripke structure.

5.2 Case studies

We have implemented the theory outlined in the previous section in a prototype tool in the open source tool suite mCRL2 [11], and used it on several model checking examples using the (action-based) first-order modal μ -calculus [12, 13] and a real-time extension thereof.¹ Our tool takes a description of the abstraction mapping and mappings of concrete operations onto abstract operations; checking the safety condition can be delegated to mCRL2's provers or left to the user in case of too complex conditions.

Lamport's Bakery Protocol for Mutual Exclusion. This protocol has an infinite state space due to the unbounded ticket numbers it can hand out. We analysed the protocol with two processes (using Boolean IDs) competing for the critical section. Typical properties that could be verified using an under-approximation (or refuted using an over-approximation) are properties 1–4 in Table 2.

The abstraction used for verifying these formulae is a mapping of the natural numbers to elements *zero* and *more*. Property 1 is a progress property, stating that processes *requesting* to enter the critical section, always *can* enter it in some future. The stronger property that such processes *inevitably* enter the critical section (property 2) provably fails. Property 3, asserting that invariantly, both processes inevitably receive *some* ticket number also provably fails. Property 4 expresses that there is always a process that can get a ticket number at least as large as the ticket number currently circulating.

A real-time ball game. This is a small real-time system, describing a game in a dense time setting in which a player has one second to pocket a ball. The game ends when the player wins, for which he is required to pocket at least 10 balls. The real-time behaviour leads to an infinite state space. We verified several reachability properties, see properties A–D in Table 2, analysing whether the game *can* finish (property A), whether the game *must* finish (property B),

¹ The examples have been added to mCRL2's open source repository; see <http://www.mcrl2.org>

Table 2. Properties for the Bakery Protocol (1–4) and the real-time ball game (A–D), verified or refuted with *pbesabsinthe*.

1. $\nu X. [\top]X \wedge \forall b:B.$ $[\text{request}(b)]\mu Y. \langle \top \rangle Y \vee \langle \text{enter}(b) \rangle \top$	holds
2. $\nu X. [\top]X \wedge \forall b:B.$ $[\text{request}(b)]\mu Y. ([\top]Y \wedge \langle \top \rangle \top) \vee \langle \text{enter}(b) \rangle \top$	fails
3. $\nu X. ([\top]X \wedge \forall b:B.$ $\mu Y. (([\top]Y \wedge \langle \top \rangle \top) \vee \exists n:N. \langle \text{c}(b, n) \rangle \top))$	fails
4. $\nu Z(i:N = 0).$ $(\exists b:B. \exists n:N. \langle \text{c}(b, n) \rangle (n \geq i \wedge Z(n))) \vee$ $(\forall b':B. \forall n':N. \neg \text{c}(b', n')) Z(i)$	holds
A. $\mu X. \langle \top \rangle X \vee \langle \text{finished} \rangle \top$	holds
B. $\mu X. [\neg \text{finished}]X \wedge \langle \top \rangle \top$	fails
C. $\mu X(i:N = 0). (i < 10) \Rightarrow \langle \text{pocket} \rangle X(i + 1)$	holds
D. $\mu X. \langle \top \rangle X \vee \langle \text{pocket@5} \rangle \top$	holds

whether the player can pocket at least 10 balls (property C), and a hard real-time property expressing whether he can pocket a ball at time 5 (property D). The abstraction we used partitions the reals between 0 and 10 in open intervals $(k, k + 1)$, the closed singular intervals $[k, k]$ and the interval $(10, \infty)$, for $k = 0 \dots 10$.

6 Related Work

Our denotational framework provides an alternative angle to the traditional operational abstraction frameworks. It shows that abstraction can, in a sense, be seen as the generalisation of logical consequence to equation systems through the use of coinduction. Below, we briefly review some of the related work.

In [19], Namjoshi addresses the question under which conditions (semantical) completeness for branching time properties can be achieved. For this, he works on *alternating transition systems*, which are basically a variation on model checking games. By imposing extra conditions on the alternating simulation relation he defines on these transition systems, a *complete* framework for abstraction is obtained. Dams and Namjoshi continue their exploration of completeness in [4] in which they introduce *focused transition systems*.

In their follow-up work [5], they show that these focused transition systems are in fact variants of μ -automata, enabling a very brief and elegant argument for completeness of their framework. Interestingly, we can use identical arguments to prove the completeness of our equation system framework for the preorder that underlies the definition of a solution to an equation system. We remark that we think that this completeness result will not give rise to a practical framework for abstraction; as Dams and Namjoshi’s completeness result, we consider this to be mostly a theoretical result.

The GTS framework has received a lot of interest from the abstraction community. Shoham and Grumberg studied the precision of the framework in [22];

Fecher and Shoham, in [7] used the framework for a more algorithmic approach to abstraction, by performing abstraction in a lazy fashion using a variation on parity games.

7 Concluding Remarks

We defined an elegant form of abstraction within the framework of equation systems, and have shown that its power is on a par with the most advanced (operational) frameworks for abstraction in the transition system setting while staying more concise. Our framework can be extended in several directions. One line of research is to investigate what is a *minimal* set of modifications to our notion of consistent consequence that is needed to achieve completeness for *all* equation systems. A second topic for investigation, which has received a lot of attention in the transition system setting, is to incorporate techniques for automated detection and refinement of abstractions on equation systems.

References

1. T. Chen, B. Ploeger, J. van de Pol, and T.A.C. Willemse. Equivalence checking for infinite systems using parameterized Boolean equation systems. In *CONCUR*, LNCS 4703, pages 120–135. Springer, 2007.
2. E.M. Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. In *POPL*, pages 342–354, 1992.
3. S. Cranen, M. Gazda, J.W. Wesselink, and T.A.C. Willemse. Abstraction in parameterised Boolean equation systems. Technical report, TU/e, 2012. Download from <http://www.win.tue.nl/~timw/downloads/CSR2012.pdf>.
4. D. Dams and K.S. Namjoshi. The existence of finite abstractions for branching time model checking. In *LICS 2004*, 2004.
5. D. Dams and K.S. Namjoshi. Automata as abstractions. In *VMCAI*, volume 3385 of *LNCS*, pages 216–232, 2005.
6. L. de Alfaro, P. Godefroid, and R. Jagadeesan. Three-valued abstractions of games: Uncertainty, but with precision. In *LICS*, pages 170–179. IEEE Computer Society, 2004.
7. H. Fecher and S. Shoham. Local abstraction-refinement for the μ -calculus. *STTT*, 13(4):289–306, 2011.
8. M. Gazda and T.A.C. Willemse. Consistent consequence for Boolean equation systems. In *Proc. of SOFSEM 2012*, volume 7147 of *LNCS*, pages 277–288, 2012.
9. M. Gazda and T.A.C. Willemse. Expressiveness and completeness in abstraction. In *EXPRESS/SOS*, volume 89 of *EPTCS*, pages 49–64, 2012.
10. P. Godefroid, M. Huth, and R. Jagadeesan. Abstraction-based model checking using modal transition systems. In *CONCUR*, volume 2154 of *LNCS*, pages 426–440, 2001.
11. J.F. Groote, J. Keiren, A. Mathijssen, B. Ploeger, F. Stappers, C. Tankink, Y. Usenko, M. van Weerdenburg, W. Wesselink, T. Willemse, and J. van der Wulp. The mCRL2 toolset. In *WASDeTT 2008*, 2008.
12. J.F. Groote and R. Mateescu. Verification of temporal properties of processes in a setting with data. In *AMAST*, volume 1548 of *LNCS*, pages 74–90. Springer, 1999.

13. J.F. Groote and T.A.C. Willemse. Model-checking processes with data. *Sci. Comput. Program*, 56(3):251–273, 2005.
14. J.F. Groote and T.A.C. Willemse. Parameterised Boolean equation systems. *Theor. Comput. Sci*, 343(3):332–369, 2005.
15. D. Kozen. Results on the propositional μ -calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.
16. K.G. Larsen and L. Xinxin. Equation solving using modal transition systems. In *LICS*, pages 108–117. IEEE Computer Society, 1990.
17. A. Mader. *Verification of Modal Properties Using Boolean Equation Systems*. PhD thesis, Technische Universität München, 1997.
18. R. Mateescu and D. Thivolle. A model checking language for concurrent value-passing systems. In *FM*, LNCS 5014. Springer-Verlag, 2008.
19. K.S. Namjoshi. Abstraction for branching time properties. In *CAV*, volume 2725 of *Lecture Notes in Computer Science*, pages 288–300. Springer, 2003.
20. S. Orzan, J.W. Wesselink, and T.A.C. Willemse. Static analysis techniques for parameterised Boolean equation systems. In *TACAS*, volume 5505 of *LNCS*, pages 230–245, 2009.
21. S.M. Orzan and T.A.C. Willemse. Invariants for parameterised Boolean equation systems. *Theor. Comp. Sci*, 411(11-13):1338–1371, 2010.
22. S. Shoham and O. Grumberg. 3-valued abstraction: More precision at less cost. *Inf. Comput.*, 206(11):1313–1333, 2008.
23. D. Zhang and R. Cleaveland. Fast generic model-checking for data-based systems. In *FORTE*, volume 3731 of *LNCS*, pages 83–97. Springer, 2005.

A Proofs

A.1 Proofs of Section 3

We use an alternative definition for the notion of a solution to an equation system, inductively defined on *blocks* (*i.e.*, equations with the same rank) rather than single equations. The two notions are known to coincide, but, compared to the definition in the main text, which is slightly less dense than the one presented here, the latter is better suited for the proof of Theorem 1.

The *product lattice* $(\mathbb{B}^{\mathbb{D}_{X_1}} \times \dots \times \mathbb{B}^{\mathbb{D}_{X_n}}, \sqsubseteq)$, where \sqsubseteq denotes the ordering \sqsubseteq lifted to vectors in the standard way, is again complete. From hereon, we abbreviate $(\llbracket \phi_{X_1 \langle d_{X_1} \rangle} \rrbracket \theta \varepsilon, \dots, \llbracket \phi_{X_n \langle d_{X_n} \rangle} \rrbracket \theta \varepsilon)$ to $\llbracket (\phi_{X_1 \langle d_{X_1} \rangle}, \dots, \phi_{X_n \langle d_{X_n} \rangle}) \rrbracket \theta \varepsilon$. We denote the i -th element of a vector (f_1, \dots, f_n) by $(f_1, \dots, f_n)_i$.

Definition 17. *The solution to an equation system, given a context of predicate and data environments θ, ε , is defined as $\llbracket \varepsilon \rrbracket \theta \varepsilon = \theta$, and for any \mathcal{E} , and block \mathcal{B} given by $(\sigma X_1(d_{X_1}:D_{X_1}) = \phi_{X_1}) \dots (\sigma X_n(d_{X_n}:D_{X_n}) = \phi_{X_n})$:*

$$\begin{aligned} & \llbracket \mathcal{B} \ \mathcal{E} \rrbracket \theta \varepsilon \\ &= \llbracket \mathcal{E} \rrbracket \theta \left[\sigma(\mathcal{X}_1, \dots, \mathcal{X}_n) \in \mathbb{B}^{\mathbb{D}_{X_1}} \times \dots \times \mathbb{B}^{\mathbb{D}_{X_n}} \cdot \tau_{\llbracket \mathcal{E} \rrbracket \theta \varepsilon}^\varepsilon(\mathcal{X}_1, \dots, \mathcal{X}_n) / (X_1, \dots, X_n) \right] \varepsilon \end{aligned}$$

where τ_θ^ε is the monotone predicate transformer induced by block \mathcal{B} :

$$\begin{aligned} \tau_\theta^\varepsilon &= \lambda(\mathcal{X}_1, \dots, \mathcal{X}_n) \in \mathbb{B}^{\mathbb{D}_{X_1}} \times \dots \times \mathbb{B}^{\mathbb{D}_{X_n}}. \\ & \quad \llbracket (\phi_{X_1 \langle d_{X_1} \rangle}, \dots, \phi_{X_n \langle d_{X_n} \rangle}) \rrbracket \theta[\mathcal{X}_1/X_1, \dots, \mathcal{X}_n/X_n] \varepsilon \end{aligned}$$

The proof for Theorem 1 employs a transfinite approximation within an inductive proof. We first recall the notion of transfinite approximations of monotone functions.

Definition 18. *Let (D, \leq) be a complete lattice with \top and \perp as top and bottom element. Let $f:D \rightarrow D$ be a monotone function. Then $\sigma^\alpha X.f(X)$ is an approximant term, where α is an ordinal. The approximant terms are defined by transfinite induction, where λ is a limit ordinal:*

$$\begin{aligned} \sigma^0 X.f(X) &= \top \text{ if } \sigma = \nu \text{ and } \perp \text{ else} \\ \sigma^{\alpha+1} X.f(X) &= f(\sigma^\alpha X.f(X)) \\ \sigma^\lambda X.f(X) &= \bigwedge_{\alpha < \lambda} \sigma^\alpha X.f(X) \text{ if } \sigma = \nu \text{ and } \bigvee_{\alpha < \lambda} \sigma^\alpha X.f(X) \text{ else} \end{aligned}$$

We first rephrase Theorem 1 as Theorem 7 below; the correspondence between both theorems is straightforward and follows from the definition of consistent environments Θ_R .

Theorem 7. *Let \mathcal{E} be an equation system. Let $R \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$ be a consistent consequence on \mathcal{E} . Then for all $\theta \in \Theta_R$ and all ε , we have $\llbracket \mathcal{E} \rrbracket \theta \varepsilon \in \Theta_R$.*

Proof. We proceed by induction on the number of blocks in \mathcal{E} . Clearly, the statement holds vacuously for $|\mathcal{E}| = 0$. Assume that for all \mathcal{E} consisting of n blocks and all consistent consequences $R \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$, we have

$$\forall \theta \in \Theta_R, \varepsilon : (\llbracket \mathcal{E} \rrbracket \theta \varepsilon) \in \Theta_R \quad (\text{IH1})$$

Let $\mathcal{B} \equiv (\nu X_1(d_{X_1}:D_{X_1}) = \phi_{X_1}) \cdots (\nu X_n(d_{X_n}:D_{X_n}) = \phi_{X_n})$ be a block of ν -equations (the case for μ is symmetric), and assume (without loss of generality) that the first block of \mathcal{E} is a μ -block. This implies that \mathcal{B} is a maximal block in $\mathcal{B}\mathcal{E}$. Let $S \subseteq \text{sig}(\mathcal{P}) \times \text{sig}(\mathcal{P})$ be a consistent consequence on $\mathcal{B}\mathcal{E}$.

Let $X, X' \in \text{bnd}(\mathcal{B}\mathcal{E})$ and assume that $X(v) S X'(v')$. Assume that $\theta \in \Theta_S$, and let ε be arbitrary. We proceed to show that $\llbracket \mathcal{B}\mathcal{E} \rrbracket \theta \varepsilon(X)(v) \Rightarrow \llbracket \mathcal{B}\mathcal{E} \rrbracket \theta \varepsilon(X')(v')$, or, equivalently, that $(\llbracket \mathcal{B}\mathcal{E} \rrbracket \theta \varepsilon) \in \Theta_S$

$$\begin{aligned} & \llbracket \mathcal{B}\mathcal{E} \rrbracket \theta \varepsilon \\ &= \llbracket \mathcal{E} \rrbracket \theta [\sigma(\mathcal{X}_1, \dots, \mathcal{X}_n) \in \mathbb{B}^{\mathbb{D}^{X_1}} \times \cdots \times \mathbb{B}^{\mathbb{D}^{X_n}} \cdot \tau_{\llbracket \mathcal{E} \rrbracket \theta \varepsilon}^\varepsilon(\mathcal{X}_1, \dots, \mathcal{X}_n) / (X_1, \dots, X_n)] \varepsilon \end{aligned}$$

Maximality of block \mathcal{B} implies that S is a consistent consequence on \mathcal{E} as well. Moreover, the first requirement in Def. 7, combined with the maximality of \mathcal{B} , requires that $X \in \mathcal{B}$ iff $X' \in \mathcal{B}$. We therefore consider the following cases:

- Case $X, X' \in \text{bnd}(\mathcal{B})$. Then, for some i, j , we have $X = X_i$, $X' = X_j$. We show $\llbracket \mathcal{B}\mathcal{E} \rrbracket \theta \varepsilon(X_i)(v) \Rightarrow \llbracket \mathcal{B}\mathcal{E} \rrbracket \theta \varepsilon(X_j)(v')$ by means of a transfinite approximation of X_1, \dots, X_n . Let α be an ordinal and denote the α -th approximation of the fixed point vector $\nu(\mathcal{X}_1, \dots, \mathcal{X}_n) \in \mathbb{B}^{\mathbb{D}^{X_1}} \times \cdots \times \mathbb{B}^{\mathbb{D}^{X_n}}$ by $(X_1^\alpha, \dots, X_n^\alpha)$. We show that

$$\forall X_i(v), X_j(v') : X_i(v) S X_j(v') \Rightarrow X_i^\alpha(v) \Rightarrow X_j^\alpha(v')$$

- For $\alpha = 0$, it follows immediately, that for all i, j , we have $X_i^0(v) = \top = X_j^0(v')$ for all $X_i(v) S X_j(v')$.
- For $\alpha = \beta + 1$ a successor ordinal, we assume the following induction hypothesis:

$$\forall X_i(v), X_j(v') : X_i(v) S X_j(v') \Rightarrow X_i^\beta(v) \Rightarrow X_j^\beta(v') \quad (\text{IH2})$$

Assume that $X_i(v) S X_j(v')$. Next, we derive:

$$\begin{aligned} & (\tau_{\llbracket \mathcal{E} \rrbracket \theta \varepsilon}^\varepsilon(X_1^{\beta+1}, \dots, X_n^{\beta+1}))_i(v) \\ &= (\llbracket (\phi_{X_1(d_{X_1})}, \dots, \phi_{X_n(d_{X_n})}) \rrbracket (\llbracket \mathcal{E} \rrbracket \theta [X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon) \varepsilon)_i(v) \\ &= \llbracket \phi_{X_i(d_{X_i})} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta [X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon) \varepsilon(v) \\ &= \llbracket \phi_{X_i} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta [X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon) \varepsilon[v/d_{X_i}] \\ &\Rightarrow^\dagger \llbracket \phi_{X_j} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta [X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon) \varepsilon[v'/d_{X_j}] \\ &= \llbracket \phi_{X_j(d_{X_j})} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta [X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon) \varepsilon(v') \\ &= (\llbracket (\phi_{X_1(d_{X_1})}, \dots, \phi_{X_n(d_{X_n})}) \rrbracket (\llbracket \mathcal{E} \rrbracket \theta [X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon) \varepsilon)_j(v') \\ &= (\tau_{\llbracket \mathcal{E} \rrbracket \theta \varepsilon}^\varepsilon(X_1^{\beta+1}, \dots, X_n^{\beta+1}))_j(v') \end{aligned}$$

where at † , we used (IH1) and (IH2) to conclude that the environment obtained by $\llbracket \mathcal{E} \rrbracket \theta[X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon$ is consistent with S , *i.e.*, belongs to Θ_S , and the fact that S is a consistent consequence on \mathcal{E} .

- For α a limit ordinal, we assume the following for all $\beta < \alpha$:

$$\forall X_i(v), X_j(v') : X_i(v) S X_j(v') \Rightarrow X_i^\beta(v) \Rightarrow X_j^\beta(v') \quad (\text{IH3})$$

From this, we obtain the following consistency result:

$$\forall X_i(v), X_j(v') : X_i(v) S X_j(v') \Rightarrow \left(\left(\bigwedge_{\beta < \alpha} X_i^\beta \right)(v) \Rightarrow \left(\bigwedge_{\beta < \alpha} X_j^\beta \right)(v') \right) \quad (*)$$

Assuming that $X_i(v) S X_j(v')$, we continue:

$$\begin{aligned} & (\tau_{\llbracket \mathcal{E} \rrbracket \theta \varepsilon}^\varepsilon(X_1^\alpha, \dots, X_n^\alpha))_i(v) \\ &= \llbracket \phi_{X_i} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta[X_1^\alpha/X_1, \dots, X_n^\alpha/X_n] \varepsilon)[v/d_{X_i}] \\ &= \llbracket \phi_{X_i} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta[\bigwedge_{\beta < \alpha} X_1^\beta/X_1, \dots, \bigwedge_{\beta < \alpha} X_n^\beta/X_n] \varepsilon)[v/d_{X_i}] \\ &\Rightarrow^\dagger \llbracket \phi_{X_j} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta[\bigwedge_{\beta < \alpha} X_1^\beta/X_1, \dots, \bigwedge_{\beta < \alpha} X_n^\beta/X_n] \varepsilon)[v/d_{X_j}] \\ &= \llbracket \phi_{X_j} \rrbracket (\llbracket \mathcal{E} \rrbracket \theta[X_1^\alpha/X_1, \dots, X_n^\alpha/X_n] \varepsilon)[v'/d_{X_j}] \\ &= (\tau_{\llbracket \mathcal{E} \rrbracket \theta \varepsilon}^\varepsilon(X_1^\alpha, \dots, X_n^\alpha))_j(v') \end{aligned}$$

where at † , we used (IH1) and the consistency result (*) that followed from (IH3) to conclude that the environment $\llbracket \mathcal{E} \rrbracket \theta[X_1^\beta/X_1, \dots, X_n^\beta/X_n] \varepsilon$ belongs to Θ_S and the fact that S is a consistent consequence on \mathcal{E} .

- Case $X, X' \notin \text{bnd}(\mathcal{B})$. Then $\llbracket \mathcal{B} \mathcal{E} \rrbracket \theta \varepsilon(X)(v) \Rightarrow \llbracket \mathcal{B} \mathcal{E} \rrbracket \theta \varepsilon(X')(v')$ follows from the induction hypothesis, and

$$\theta[\sigma(\mathcal{X}_1, \dots, \mathcal{X}_n) \in \mathbb{B}^{\mathbb{D}_{X_1}} \times \dots \times \mathbb{B}^{\mathbb{D}_{X_n}}. \tau_{\llbracket \mathcal{E} \rrbracket \theta \varepsilon}^\varepsilon(\mathcal{X}_1, \dots, \mathcal{X}_n)/(X_1, \dots, X_n)] \in \Theta_S$$

which follows immediately from the first case. \square

A.2 Proofs of Section 4

For proving the main results in this section, we will use an alternative syntax of the μ -calculus, namely that of Modal Equation Systems (MESs), as used by *e.g.*, Andersen [Andersen, *Partial Model Checking*, LICS'96]. MESs are, essentially, an equational variant of the μ -calculus, and, notationally, are a mix between μ -calculus formulae and Boolean equation systems. Their expressive power equals that of the μ -calculus: both can be transformed into one another (at the expense of an exponential blow-up when moving from MESs to linear μ -calculus formulae). Since MESs are syntactically closer to equation systems, the link between the MES model checking problem and the (parameterised) Boolean equation system solving problem is much more direct.

Definition 19. A Modal Equation System (MES) is a sequence of equations defined with the following grammar:

$$\begin{aligned}\mathcal{M} &::= \epsilon \mid (\mu X = f) \mathcal{M} \mid (\nu X = f) \mathcal{M} \\ f, g &::= \top \mid \perp \mid l \mid \neg l \mid X \mid f \wedge g \mid f \vee g \mid \Box f \mid \Diamond f\end{aligned}$$

Let η be an environment, ranging over $\mathcal{X} \rightarrow 2^S$. The necessary semantics of a MES in the context of a GTS $M = \langle S, s_0, R^+, R^-, L \rangle$ and environment η is defined inductively as follows:

$$\begin{aligned}\llbracket \epsilon \rrbracket_M^\square \eta &= \eta \\ \llbracket (\mu X = f_X) \mathcal{M} \rrbracket_M^\square \eta &= \llbracket \mathcal{M} \rrbracket_M^\square \eta [(\mu U \in 2^S. \llbracket f_X \rrbracket_M^\square \llbracket \mathcal{M} \rrbracket_M^\square \eta [X := U]) / X] \\ \llbracket (\nu X = \phi_X) \mathcal{M} \rrbracket_M^\square \eta &= \llbracket \mathcal{M} \rrbracket_M^\square \eta [(\nu U \in 2^S. \llbracket \phi_X \rrbracket_M^\square \llbracket \mathcal{M} \rrbracket_M^\square \eta [X := U]) / X] \\ \llbracket \perp \rrbracket_M^\square \eta &= \emptyset \\ \llbracket \top \rrbracket_M^\square \eta &= S \\ \llbracket X \rrbracket_M^\square \eta &= \eta(X) \\ \llbracket f \vee g \rrbracket_M^\square \eta &= \llbracket f \rrbracket_M^\square \eta \cup \llbracket g \rrbracket_M^\square \eta \\ \llbracket f \wedge g \rrbracket_M^\square \eta &= \llbracket f \rrbracket_M^\square \eta \cap \llbracket g \rrbracket_M^\square \eta \\ \llbracket \Diamond f \rrbracket_M^\square \eta &= \{s \in S \mid \exists A \in 2^S : sR^+ A \wedge \forall q \in A : q \in \llbracket f \rrbracket_M^\square \eta\} \\ \llbracket \Box f \rrbracket_M^\square \eta &= \{s \in S \mid \forall s' \in S : sR^- s' \Rightarrow s' \in \llbracket f \rrbracket_M^\square \eta\}\end{aligned}$$

The possibly semantics of a MES is defined analogously, but since we do not need it for our results in this section, we omit it.

We permit ourselves to use the same notation for MESs as for equation systems; e.g., $\text{bnd}(\mathcal{M})$ contains the set of binding variables in MES \mathcal{M} . Likewise, whenever a MES \mathcal{M} is *closed*, we omit the environment from the semantic brackets, as the semantics for the binding variables is independent of this environment.

The encoding of a MES model checking problem as an equation system solving problem can be seen as creating instances of variables corresponding to every state, and replacing modal operators with the appropriate semantics. That is, the diamond operator is transformed to disjunctions by combining emanating from a certain state of a GTS with the formula, and, in a similar vein, the box operator is transformed to conjunctions.

Since the transformation of the MES model checking problem on GTSs to an equation system solving problem is not restricted to finite GTSs, from hereon, we tacitly assume that the constituents of a GTS $G = \langle S, s_0, R^+, R^-, L \rangle$ are represented by suitable predicates, where we use variable d ranging over some sort D to represent the states S of G ; R^- is a relation on variables d and d' , the latter representing the next states; R^+ is a relation on variables d and A , the latter representing *sets of states*. We write $R^-(d, d')$, $R^+(d, A)$ and $L(d)$ when we reason about the predicates rather than “explicit” counterparts.

Definition 20. Given a GTS $G = \langle S, s_0, R^+, R^-, L \rangle$, and a MES \mathcal{M} , we define the following encoding of the necessary model checking problem into an equation

system solving problem.

$$\begin{aligned}
\mathbf{E}_G^\square(\epsilon) &\triangleq \epsilon \\
\mathbf{E}_G^\square((\sigma X = f) \mathcal{M}) &\triangleq \{(\sigma X^\square(d:D) = \mathbf{RHS}_G^\square(f)d)\} \mathbf{E}_G^\square(\mathcal{M}) \\
\\
\mathbf{RHS}_G^\square(\top)d &\triangleq \top \\
\mathbf{RHS}_G^\square(\perp)d &\triangleq \perp \\
\mathbf{RHS}_G^\square(l)d &\triangleq l \in L(d) \\
\mathbf{RHS}_G^\square(\neg l)d &\triangleq \neg l \in L(d) \\
\mathbf{RHS}_G^\square(X)d &\triangleq X^\square(d) \\
\mathbf{RHS}_G^\square(f \oplus g)d &\triangleq \mathbf{RHS}_G^\square(f)d \oplus \mathbf{RHS}_G^\square(g)d \\
\mathbf{RHS}_G^\square(\diamond f)d &\triangleq \exists A:2^D. R^+(d, A) \wedge \forall d':D. d' \in A \Rightarrow ((\mathbf{RHS}_G^\square(f)d')[d'/d]) \\
\mathbf{RHS}_G^\square(\Box f)d &\triangleq \forall d':D. R^-(d, d') \Rightarrow ((\mathbf{RHS}_G^\square(f)d')[d'/d])
\end{aligned}$$

Note that if a MES is closed, the above transformation will yield a closed equation system. The proposition below states the correspondence between MES model checking on GTSs and solving an equation system.

Proposition 3 (Soundness of encoding necessary semantics on GTSs).

Let \mathcal{M} be a closed MES, and let $X \in \text{bnd}(\mathcal{M})$. Given a GTS $G = \langle S, s_0, R^+, R^-, L \rangle$, we have, for any $s \in S$:

$$s \in \llbracket \mathcal{M} \rrbracket_G^\square(X) \text{ iff } \llbracket \mathbf{E}_G^\square(\mathcal{M}) \rrbracket(X)(s) = \top$$

We proceed to show that mixed simulation between GTSs induces a consistent consequence between variable instances of the related states. We will first need to introduce some notation.

Observe that if $\mathcal{E} = \mathbf{E}(G)(\mathcal{M})$, then any environment $\theta : \text{bnd}(\mathcal{E}) \rightarrow \mathbb{D} \rightarrow \mathbb{B}$ induces a corresponding environment $\eta(\theta) : \mathcal{X} \rightarrow \mathcal{P}(S)$, namely

$$\eta(\theta)(X) \triangleq \{s : \theta(X_G^\square)(s) = \top\}$$

When comparing right-hand sides of encodings of the model checking problem in the concrete and abstract case, we will need the following straightforward observation, which we state without proof.

Lemma 3. For any GTS G , any of its states s , environment θ and modal formula φ without binding operators (one in the form that appears on the right-hand side of a MES), we have the following equivalence:

$$\llbracket \mathbf{RHS}_G^\square(\varphi)[s/d] \rrbracket \theta = \top \Leftrightarrow s \in \llbracket \varphi \rrbracket_G^\square \eta(\theta)$$

Now we can prove the key lemma that allows us to establish consistent consequence between encoding of a model checking problem on a GTS and Kripke structure that it abstracts.

Lemma 4. Assume two GTSs $G_C = (S_C, s_0^C, R, L_C)$ and $G_A = (S_A, s_0^A, R^+, R^-, L_A)$. Take any $s_C \in S_C$ and $s_A \in S_A$ such that there is a mixed simulation from s_C to s_A ($s_C \leq s_A$). For all $\varphi \in L_\mu$, and for any environment θ , consistent with \leq (i.e. if $q \leq s$, then $\theta(X_{G_A}^\square)(s) \Rightarrow \theta(X_{G_C}^\square)(q)$) we have:

$$\llbracket \mathbf{RHS}_{G_A}^\square(\varphi)d[s_A/d] \rrbracket \theta \Rightarrow \llbracket \mathbf{RHS}_{G_C}^\square(\varphi)d[s_C/d] \rrbracket \theta$$

Proof. Take any environment θ , consistent with \leq , i.e. if $s \leq q$, then $\theta(X)(s) \Rightarrow \theta(X)(q)$. The proof now proceeds with straightforward structural induction on φ . We will only present the proof of the most involved part of the inductive step.

– $\Diamond\varphi$: we have

$$\llbracket \mathbf{RHS}_{G_A}^\square(\Diamond\varphi)d[s_A/d] \rrbracket \theta = \top$$

(def. of RHS operator)

$$\Leftrightarrow \llbracket \exists B.s_A R^+ B \wedge \forall d'.d' \in B \Rightarrow \mathbf{RHS}_{G_A}^\square(\varphi)q_A[d'/d] \rrbracket \theta = \top$$

(unfolding semantics)

$$\Leftrightarrow \bigvee_{B:s_A R^+ B} \bigwedge_{q_A \in B} \llbracket \mathbf{RHS}_{G_A}^\square(\varphi)q_A[q_A/d] \rrbracket \theta = \top$$

Take B_0 that makes the above disjunction true. From Lemma 3 we know that, for all $q_A \in B_0$, $q_A \in \llbracket \varphi \rrbracket_{G_A}^\square \eta(\theta)$. From the fact that $s_C \leq s_A$, there exists q_C such that $s_C R q_C$ and $\forall q_A \in B_0, q_C \leq q_A$. Because mixed simulation ‘backward-preserves’ necessary semantics, we have $q_C \in \llbracket \varphi \rrbracket_{G_C}^\square \eta(\theta)$. By using Lemma 3 again, we have $\llbracket \mathbf{RHS}_{G_C}^\square(\varphi)q_C[q_C/d] \rrbracket \theta = \top$.

From the above observation, substituting a singleton set $\{q_C\}$ for B yields the following:

$$\bigvee_{B:s_C R^+ B} \bigwedge_{q_C \in B} \llbracket \mathbf{RHS}_{G_C}^\square(\varphi)q_C[q_C/d] \rrbracket \theta = \top$$

(folding semantics)

$$\Leftrightarrow \llbracket \exists B.s_C R^+ B \wedge \forall d'.d' \in B \Rightarrow \mathbf{RHS}_{G_C}^\square(\varphi)q_C[d'/d] \rrbracket \theta = \top$$

(def. of RHS operator)

$$\Leftrightarrow \llbracket \mathbf{RHS}_{G_C}^\square(\Diamond\varphi)d[s_C/d] \rrbracket \theta = \top$$

□

Proposition 4. Assume that $G_C = (S_C, s_0^C, R, L_C)$ is a concrete Kripke Structure and $G_A = (S_A, s_0^A, R^+, R^-, L_A)$ a GTS. Take any $s_C \in S_C$ and $s_A \in S_A$ such that there is a mixed simulation from s_C to s_A ($s_C \leq s_A$). We have:

$$X_{G_A}^\square(s_A) \leq X_{G_C}^\square(s_C)$$

As a result of the above proposition and the soundness of the MES model checking problem, we have proven one part of Theorem 2. For the other direction, we proceed as follows.

We want to prove that if encoding of a model checking problem can be approximated (using consistent consequence) with a finite equation system, then there exists a finite GTS that abstracts the original Kripke structure and the desired formula can be proved on this abstraction.

Observe first that in order to prove this fact, it suffices to show that from the existence of finite approximation in the form of equation system it follows that there is a certain least fixpoint-free formula (or MES, in our setting), which semantically implies the original formula and is satisfied by the original model. The existence of a proper finite GTS is then a result of completeness of the GTS framework for greatest-fixpoint only fragment of the modal μ -calculus.

A natural candidate for such such greatest fixpoint-only “witnesses” are the μ approximants, defined as below.

Definition 21. Assume that \mathcal{M} has m μ variables. A μ -approximant of a MES \mathcal{M} , denoted as $(\mathcal{M})_\mu(K)$ with respect to K , contains, for each propositional variable X of the original MES \mathcal{M} , K^m copies of X , parameterised with a vector α , formally defining propositional variables are contained in the set

$$\{X(\alpha) \mid 0 \leq \alpha_i \leq K \forall i \in \{1, \dots, m\}\}$$

The right-hand sides are defined as follows.

$$\begin{aligned} \mu X(\alpha) &= \perp && \text{if } \alpha_i = 0 \\ \mu X(\alpha) &= f_X[Y := Y(\alpha[\alpha_i := \alpha_i - 1, \alpha_j := K \forall j > i])] && \text{if } \alpha_i > 0 \\ \nu X(\alpha) &= f_X[Y := Y(\alpha[\alpha_j := K \forall j > i])] \end{aligned}$$

where i is the consecutive number in which X appears in \mathcal{E} .

The α vector specifies, the amount of unfoldings of each μ -variable that are still permitted to be taken, without visiting (looking from the corresponding parity game perspective) a more important rank.

Proposition 5. For every closed MES \mathcal{M} with m least fixpoint equations and $\alpha \in \mathbb{N}^m$ we have:

1. $(\mathcal{M})_\mu(K)$ semantically implies \mathcal{M} , i.e. for every GTS G , $\llbracket (\mathcal{M})_\mu(K) \rrbracket_G^\square \subseteq \llbracket \mathcal{M} \rrbracket_G^\square$
2. there is a least fixpoint-free MES \mathcal{M}' such that $\mathcal{M}' \equiv (\mathcal{M})_\mu(K)$, i.e. for every GTS G , $\llbracket \mathcal{M}' \rrbracket_G^\square = \llbracket (\mathcal{M})_\mu(K) \rrbracket_G^\square$.

The first of the above facts is straightforward to observe. Intuitively, $(\mathcal{M})_\mu(K)$ makes right-hand sides stronger by replacing variables with the constant \perp , or a copy of the same variable, which cannot have a larger potential to be true. The second observation follows from the fact that, if we look at the MES solution for any model from a game perspective, player “Odd” or “False” can win the play only if it is finite, since every odd priority can be visited only finitely many times without a more important even priority occurring in between.

Suppose we have a possibly infinite GTS G (for our purposes it suffices that G is a Kripke structure, but this assumption is irrelevant for the proof). The next step is to establish the sufficient condition on the PBES encoding $\mathbf{E}_G^\square(\mathcal{M})$ such that there exists some $K \in \mathbb{N}$ for which $\mathbf{E}_G^\square((\mathcal{M})_\mu(K))$ has solution true. This condition can be established by switching to the game-theoretical framework.

In order to avoid elaborating too much on yet another formalism, we will explain our line of reasoning in an informal way. Deciding solution of a PBES can be seen as finding a winner of the following game, which very much resembles a parity game. It is played by two players: “True” (or “Disjunctive”) and “False” (or “Conjunctive”). The arena consists of subformulae of right-hand sides of a PBES, labelled with the defining variable and data value instance (so if there is an equation $(\sigma X(d : D) = \phi_X)$, then for every possible value v that d can take and every subformula ψ of $\phi_X[v/d]$, a pair $((X(v), \psi)$ is a valid node in solution game. The edges connect formulae with their direct subformulae, formulae of the

form $((X(v), Y(w)))$ are linked with $(Y(w), \phi_Y[w/d])$ and priorities are simply ranks of defining predicate variables. Ownership is defined in a natural way for boolean connectives and quantifiers.

We give a more formal definition below. Let $Sub(\varphi)$ denote the set of all subformulae of φ . As a convention, we denote the predicate formula occurring at the right-hand side of X in \mathcal{E} as ϕ_X .

Definition 22. For a PBES \mathcal{E} we define the corresponding solution game as a tuple $(V, V_\vee, V_\wedge, E, \Omega)$, where:

- $V = \bigcup_{X \in bnd(\mathcal{E})} \text{sig}(X) \times Sub(\phi_X)$
- V_\vee are those nodes $((X, v), \phi)$ in which ϕ is of the form $\exists d : D.\psi$ or $\psi_1 \vee \psi_2$
- V_\wedge are those nodes $((X, v), \phi)$ in which ϕ is of the form $\forall d : D.\psi$ or $\psi_1 \wedge \psi_2$
- the rest of the nodes can be assigned an arbitrary owner (they have at most one outgoing edge)
- outgoing edges of $s = ((X, v), \phi)$ are defined as follows (sE denotes successors of s):
 - if $\phi = b$, then $sE = \emptyset$
 - if $\phi = \psi_1 \oplus \psi_2$, then $sE = \{((X, v), \psi_1), ((X, v), \psi_2)\}$
 - if $\phi = Qd : D.\psi$ for $Q \in \{\exists, \forall\}$, then $sE = \{((X, v), \psi[d = v]) \mid v \in \mathbb{D}\}$
 - if $\phi = Y(e)$, then $sE = \{((Y, e), \phi_Y)\}$
- $\Omega(((X, v), \phi)) \triangleq \text{rank}_{\mathcal{E}}(X)$

A play π is a possibly infinite path in a SG. A play is winning for player True, if either the play is finite and the last state is labelled with a boolean value \top , or if it is infinite and the lowest value occurring infinitely often in π is even. A strategy σ of player P , given a history of play, assigns to each node owned by P a choice of an outgoing edge. All plays consistent with a strategy σ are denoted with Π_σ .

The key property that suffices for the existence of a least fixpoint-free “witness” MES is the finiteness of odd stretches.

Definition 23. An odd stretch (μ -stretch) in a play π is a sequence $i_1 < i_2 < \dots < i_k$ such that $\Omega(\pi_{i_1}) = \Omega(\pi_{i_2}) = \dots = \Omega(\pi_{i_k}) = \omega$ for some odd ω and there is no $j : i_1 \leq j \leq i_k$ such that $\Omega(\pi_j) < \omega$ and $\Omega(\pi_j)$ is even.

We first observe that \mathcal{E} be a BES and \mathcal{E}' a PBES such that $X \triangleleft Y(v)$ for $X \in bnd(\mathcal{E}), Y \in bnd(\mathcal{E}')$, then in the game corresponding to \mathcal{E}' the player True can force a winning strategy in which every odd stretch is bounded by some global natural number K .

To see that it is indeed the case, we can show that for every strategy σ of player True on left-hand side \mathcal{E} there exists a strategy σ' of player True on \mathcal{E}' such that there is a play in Π_σ on the left-hand side in which occur the same “stripes” of the same consecutive priorities (they may differ in length, but the order is the same) and there is a global bound on the length of a “stripe” of consecutive subformulae of the same right-hand side (this is because the tree of subformulae of a predicate formula has a bounded depth). An important property

of consistent consequence used here is that it requires the same priorities on the related variables.

Secondly, if the above situation (global bound on odd stretches) holds for a certain encoding of a model checking problem $\mathbf{E}_G^\square(\mathcal{M})$, then there is a value $L \in \mathbb{N}$, such that $\llbracket \mathbf{E}_G^\square((\mathcal{M})_\mu(L))(X(L, \dots, L))(s) \rrbracket = \top$. This is because PBESs for full formula and approximant have the same structure, so we can adopt strategy from the encoding of the approximant to the full formula PBES; now if False wins the L -th approximant, then from its construction it can visit the same variable L times in the full PBES without a variable with a lower (more significant) priority.

The above observations yield the following two lemmata, which formalise the basic reasons for correctness of Theorem 2.

Lemma 5. *Let \mathcal{E} be a BES and \mathcal{E}' a PBES such that $X \prec Y(v)$ for $X \in \text{bnd}(\mathcal{E})$, $Y \in \text{bnd}(\mathcal{E}')$. Then in the PBES solution game corresponding to \mathcal{E}' there is a strategy σ , starting in $Y(v)$ and winning for True, and a value $K \in \mathbb{N}$, such that every odd stretch in any play $\pi \in \Pi_\sigma$ has length $\leq K$.*

Lemma 6. *Let G be a GTS and \mathcal{M} a MES such that in the solution game corresponding to $\mathbf{E}_G^\square(\mathcal{M})$ there is a strategy σ , starting in $X(s)$ and winning for True, and a value $K \in \mathbb{N}$, such that every odd stretch in any play $\pi \in \Pi_\sigma$ has length $\leq K$. Then there is a certain bound $L \in \mathbb{N}$, such that $\llbracket \mathbf{E}_G^\square((\mathcal{M})_\mu(L))(X(K, \dots, K))(s) \rrbracket = \top$.*

Using the results we obtained above, we can now formally prove Theorem 2:

Theorem 2. *Let $M_1 = \langle S_1, R_1^+, R_1^-, L_1 \rangle$ be a Kripke structure and f an arbitrary closed μ -calculus formula such that $M, s_1 \models f$ for some $s_1 \in S_1$. Then there is a finite GTS $M_2 = \langle S_2, R_2^+, R_2^-, L_2 \rangle$ such that $\langle M_1, s_1 \rangle \leq \langle M_2, s_2 \rangle$ and $M_2, s_2 \models f$, iff there is a closed Boolean equation system $\bar{\mathcal{E}}$ such that $\bar{X} \prec V_{M_1, f}(s_1)$ and $\llbracket \bar{\mathcal{E}} \rrbracket(\bar{X}) = \top$.*

Proof. For the implication from left to right, stating that any formula that can be proved using a finite abstraction in the GTS framework can also be proved using a consistent consequence in the equation system setting follows from lemmas 3 and 4.

The implication in the other direction follows from combining lemmas 5, 6, Proposition 5 and the completeness of GTS framework for least fixpoint-free fragment of the μ -calculus. \square

A.3 Proofs of Section 5

In the following part we assume an isomorphism between the basic elements of the syntactic domain (D) and the semantic domain \mathbb{D} , and we permit ourselves to use them interchangeably for the sake of readability.

We will call a term or formula *concrete* [resp. *abstract*] if all the syntactic objects occurring therein (constants, variables, operators) are concrete [abstract]. First, we will introduce the following auxiliary lemma.

Lemma 7. *For all concrete predicate formulae ϕ , and for all predicate environments θ such that for all $X \in \text{bnd}(\mathcal{E})$, $v \in D_X$:*

$$(*) \quad \theta(\widehat{X}_\sqcap)(h(v)) \Rightarrow \theta(X)(v) \Rightarrow \theta(\widehat{X}_\sqcup)(h(v))$$

and for all data environments δ we have:

$$\llbracket \mathcal{A}_\sqcap(\phi) \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}] \Rightarrow \llbracket \phi \rrbracket \theta \delta[v/\mathbf{d}] \Rightarrow \llbracket \mathcal{A}_\sqcup(\phi) \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}]$$

Proof. We will prove the above property for the underapproximation transformation, the overapproximation case is similar. We assume that an environment θ is such that $(*)$ holds; we will show that

$$\llbracket \mathcal{A}_\sqcap(\phi) \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}] \Rightarrow \llbracket \phi \rrbracket \theta \delta[v/\mathbf{d}]$$

We proceed with structural induction on ϕ . The two base cases are the key cases.

– $\phi = b(\mathbf{d})$:

Suppose that $\llbracket \mathcal{A}_\sqcap(b(\mathbf{d})) \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}] = \top$. We will show that $\llbracket b(\mathbf{d}) \rrbracket \theta \delta[v/\mathbf{d}] = \top$ (which is equivalent to $\llbracket b(v) \rrbracket \delta = \top$).

We have:

$$\begin{aligned} & \llbracket \mathcal{A}_\sqcap(b(\mathbf{d})) \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}] = \top \\ & \text{(no predicate variable occurs in } b(\mathbf{d})) \\ & \Leftrightarrow \llbracket \mathcal{A}_\sqcap(b(\mathbf{d})) \rrbracket \delta[h(v)/\overline{\mathbf{d}}] = \top \\ & \text{(def. of } \mathcal{A}_\sqcap()) \\ & \Leftrightarrow \llbracket \forall \mathbf{v}_b : \text{Bool. } \mathbf{v}_b \in \widehat{b}(\overline{\mathbf{d}}) \Rightarrow \mathbf{v} \rrbracket \delta[h(v)/\overline{\mathbf{d}}] = \top \\ & \text{(unfolding semantics \& simplification)} \\ & \Leftrightarrow \llbracket \perp \notin \widehat{b}(\overline{\mathbf{d}}) \rrbracket \delta[h(v)/\overline{\mathbf{d}}] = \top \\ & \text{(applying substitution)} \\ & \Leftrightarrow \perp \notin \llbracket \widehat{b}(h(v)) \rrbracket \delta \end{aligned}$$

From the construction of lifted functions it follows that $\llbracket \widehat{b}(h(v)) \rrbracket \delta \neq \emptyset$, so $\llbracket \widehat{b}(h(v)) \rrbracket \delta$ must be equal to $\{\top\}$. From the safety condition we know that $\llbracket h(b(v)) \rrbracket \delta \in \llbracket \widehat{b}(h(v)) \rrbracket \delta$, but since the abstraction function for booleans is an identity, we obtain: $\llbracket b(v) \rrbracket \delta \in \llbracket \widehat{b}(h(v)) \rrbracket \delta = \{\top\}$, hence $\llbracket b(v) \rrbracket \delta = \top$.

– $\phi = X(e(\mathbf{d}))$ for some expression $e : D \rightarrow E$

Assume that $\llbracket \mathcal{A}_\sqcap(X(e(\mathbf{d}))) \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}] = \top$. We will show that

$$\llbracket X(e(\mathbf{d})) \rrbracket \theta \delta[v/\mathbf{d}] = \top, \text{ or equivalently } \llbracket X(e(v)) \rrbracket \theta \delta = \top.$$

We have:

$$\begin{aligned} & \llbracket \mathcal{A}_\sqcap(X(e(\mathbf{d}))) \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}] = \top \\ & \text{(def. of } \mathcal{A}_\sqcap()) \\ & \Leftrightarrow \llbracket \forall \mathbf{v}' \in E. \mathbf{v}' \in \widehat{e}(\overline{\mathbf{d}}) \Rightarrow \widehat{X}_\sqcap(\mathbf{v}') \rrbracket \theta \delta[h(v)/\overline{\mathbf{d}}] = \top \\ & \text{(unfolding semantics)} \\ & \Leftrightarrow (\forall \mathbf{v}' \in E. \llbracket \mathbf{v}' \in \widehat{e}(h(v)) \rrbracket \theta \delta \Rightarrow \llbracket \widehat{X}_\sqcap(\mathbf{v}') \rrbracket \theta \delta = \top) \end{aligned}$$

From the safety condition we have $h(e(v)) \in \llbracket \widehat{e}(\{h(v)\}) \rrbracket \theta \delta$, so from the above equality we obtain that $\llbracket \widehat{X}_\sqcap(h(e(v))) \rrbracket \theta \delta = \top$. From the assumption about the environment θ $(*)$ we finally obtain that $\llbracket X(e(v)) \rrbracket \theta \delta = \top$.

We now proceed with a (very straightforward) inductive step; we assume that the property holds for all subformulae of ϕ .

- $\phi = (\chi \oplus \psi)(\mathbf{d})$
 - For $\oplus = \vee$: assume that $\llbracket \mathcal{A}_\sqcap((\chi \vee \psi)(\mathbf{d})) \rrbracket \theta\delta[h(v)/\bar{\mathbf{d}}] = \top$; we have
 (def. of $\mathcal{A}_\sqcap()$)
 $\Leftrightarrow \llbracket \mathcal{A}_\sqcap(\chi(\mathbf{d})) \vee \mathcal{A}_\sqcap(\psi(\mathbf{d})) \rrbracket \theta\delta[h(v)/\bar{\mathbf{d}}] = \top$
 $\Leftrightarrow (\llbracket \mathcal{A}_\sqcap(\chi(\mathbf{d})) \rrbracket \theta\delta[h(v)/\bar{\mathbf{d}}] = \top) \vee (\llbracket \mathcal{A}_\sqcap(\psi(\mathbf{d})) \rrbracket \theta\delta[h(v)/\bar{\mathbf{d}}] = \top)$
 (inductive hypothesis)
 $\Rightarrow (\llbracket \chi(\mathbf{d}) \rrbracket \theta\delta[v/\mathbf{d}] = \top) \vee (\llbracket \psi(\mathbf{d}) \rrbracket \theta\delta[v/\mathbf{d}] = \top)$
 $\Leftrightarrow \llbracket (\chi \vee \psi)(\mathbf{d}) \rrbracket \theta\delta[v/\mathbf{d}] = \top$
 - For $\oplus = \wedge$: very similar
- $\phi = (\mathbf{Q} \, d':D. \psi)(\mathbf{d})$
 - $\mathbf{Q} = \exists$: assume that $\mathcal{A}_\sqcap(\exists d':D. \psi)(\mathbf{d}) \theta\delta[h(v)/\bar{\mathbf{d}}] = \top$; we have
 (def. of $\mathcal{A}_\sqcap()$)
 $\llbracket \exists \hat{\mathbf{d}}':\hat{D}. \mathcal{A}_\sqcap(\psi(d', \mathbf{d})) \rrbracket \theta\delta[h(v)/\bar{\mathbf{d}}] = \top$
 (unfolding semantics)
 $\Leftrightarrow \exists \hat{v}':\hat{\mathbb{D}}. \llbracket \mathcal{A}_\sqcap(\psi(d', \mathbf{d})) \rrbracket \theta\delta[(\hat{v}', h(v))/(\bar{d}', \bar{\mathbf{d}})] = \top$
 (h is a surjection)
 $\Leftrightarrow \exists v':\mathbb{D}. \llbracket \mathcal{A}_\sqcap(\psi(d', \mathbf{d})) \rrbracket \theta\delta[(h(v'), h(v))/(\bar{d}', \bar{\mathbf{d}})] = \top$
 (h defined pointwise for vectors & d' does not occur in $\mathcal{A}_\sqcap(\psi)$)
 $\Leftrightarrow \exists v':\mathbb{D}. \llbracket \mathcal{A}_\sqcap(\psi(d', \mathbf{d})) \rrbracket \theta\delta[h(v', v)/(\bar{d}', \bar{\mathbf{d}})] = \top$
 (inductive hypothesis)
 $\Rightarrow \exists v':\mathbb{D}. \llbracket \psi(d', \mathbf{d}) \rrbracket \theta\delta[v/\mathbf{d}, v'/d'] = \top$
 (h defined pointwise for vectors)
 $\Leftrightarrow \exists v':\mathbb{D}. \llbracket \psi(d', \mathbf{d}) \rrbracket \theta\delta[v/\mathbf{d}, v'/d'] = \top$
 (def. of semantics)
 $\Leftrightarrow \llbracket \exists d':D. \psi(d', \mathbf{d}) \rrbracket \theta\delta[v/\mathbf{d}] = \top$
 - $\mathbf{Q} = \forall$: similar

Theorem 6. *Let \mathcal{E} be an arbitrary closed equation system. Then for all $X \in \text{bnd}(\mathcal{E})$ and all closed terms $v:D_X$, we have:*

$$\llbracket \mathcal{A}_\sqcap(\mathcal{E}) \rrbracket(\hat{X}_\sqcap)(\llbracket h(v) \rrbracket) \Rightarrow \llbracket \mathcal{E} \rrbracket(X)(\llbracket v \rrbracket) \Rightarrow \llbracket \mathcal{A}_\sqcup(\mathcal{E}) \rrbracket(\hat{X}_\sqcup)(\llbracket h(v) \rrbracket)$$

Proof. As a consequence of Lemma 7, it also follows that we have $\hat{X}_\sqcap(\llbracket h(v) \rrbracket) \leq X(\llbracket v \rrbracket) \leq \hat{X}_\sqcup(\llbracket h(v) \rrbracket)$. By Theorem 1, we then have the desired result. \square

If you want to receive reports, send an email to: wsinsan@tue.nl (we cannot guarantee the availability of the requested reports).

In this series appeared (from 2009):

09/01	Wil M.P. van der Aalst, Kees M. van Hee, Peter Massuthe, Natalia Sidorova and Jan Martijn van der Werf	Compositional Service Trees
09/02	P.J.I. Cuijpers, F.A.J. Koenders, M.G.P. Pustjens, B.A.G. Senders, P.J.A. van Tilburg, P. Verduin	Queue merge: a Binary Operator for Modeling Queueing Behavior
09/03	Maarten G. Meulen, Frank P.M. Stappers and Tim A.C. Willemse	Breadth-Bounded Model Checking
09/04	Muhammad Atif and MohammadReza Mousavi	Formal Specification and Analysis of Accelerated Heartbeat Protocols
09/05	Michael Franssen	Placeholder Calculus for First-Order logic
09/06	Daniel Trivellato, Fred Spiessens, Nicola Zannone and Sandro Etalle	POLIPO: Policies & OntoLogies for the Interoperability, Portability, and autOnomy
09/07	Marco Zapletal, Wil M.P. van der Aalst, Nick Russell, Philipp Liegl and Hannes Werthner	Pattern-based Analysis of Windows Workflow
09/08	Mike Holenderski, Reinder J. Bril and Johan J. Lukkien	Swift mode changes in memory constrained real-time systems
09/09	Dragan Bošnački, Aad Mathijssen and Yaroslav S. Usenko	Behavioural analysis of an PC Linux Driver
09/10	Ugur Keskin	In-Vehicle Communication Networks: A Literature Survey
09/11	Bas Ploeger	Analysis of ACS using mCRL2
09/12	Wolfgang Boehmer, Christoph Brandt and Jan Friso Groote	Evaluation of a Business Continuity Plan using Process Algebra and Modal Logic
09/13	Luca Aceto, Anna Ingolfssdottir, MohammadReza Mousavi and Michel A. Reniers	A Rule Format for Unit Elements
09/14	Maja Pešić, Dragan Bošnački and Wil M.P. van der Aalst	Enacting Declarative Languages using LTL: Avoiding Errors and Improving Performance
09/15	MohammadReza Mousavi and Emil Sekerinski, Editors	Proceedings of Formal Methods 2009 Doctoral Symposium
09/16	Muhammad Atif	Formal Analysis of Consensus Protocols in Asynchronous Distributed Systems
09/17	Jeroen Keiren and Tim A.C. Willemse	Bisimulation Minimisations for Boolean Equation Systems
09/18	Kees van Hee, Jan Hidders, Geert-Jan Houben, Jan Paredaens, Philippe Thiran	On-the-fly Auditing of Business Processes
10/01	Ammar Osaiweran, Marcel Boosten, MohammadReza Mousavi	Analytical Software Design: Introduction and Industrial Experience Report
10/02	F.E.J. Kruseman Aretz	Design and correctness proof of an emulation of the floating-point operations of the Electrologica X8. A case study

10/03	Luca Aceto, Matteo Cimini, Anna Ingolfsdottir, MohammadReza Mousavi and Michel A. Reniers	On Rule Formats for Zero and Unit Elements
10/04	Hamid Reza Asaadi, Ramtin Khosravi, MohammadReza Mousavi, Neda Noroozi	Towards Model-Based Testing of Electronic Funds Transfer Systems
10/05	Reinder J. Bril, Uğur Keskin, Moris Behnam, Thomas Nolte	Schedulability analysis of synchronization protocols based on overrun without payback for hierarchical scheduling frameworks revisited
10/06	Zvezdan Protić	Locally unique labeling of model elements for state-based model differences
10/07	C.G.U. Okwudire and R.J. Bril	Converting existing analysis to the EDP resource model
10/08	Muhammed Atif, Sjoerd Cranen, MohammadReza Mousavi	Reconstruction and verification of group membership protocols
10/09	Sjoerd Cranen, Jan Friso Groote, Michel Reniers	A linear translation from LTL to the first-order modal μ -calculus
10/10	Mike Holenderski, Wim Cools Reinder J. Bril, Johan J. Lukkien	Extending an Open-source Real-time Operating System with Hierarchical Scheduling
10/11	Eric van Wyk and Steffen Zschaler	1 st Doctoral Symposium of the International Conference on Software Language Engineering (SLE)
10/12	Pre-Proceedings	3 rd International Software Language Engineering Conference
10/13	Faisal Kamiran, Toon Calders and Mykola Pechenizkiy	Discrimination Aware Decision Tree Learning
10/14	J.F. Groote, T.W.D.M. Kouters and A.A.H. Osaiweran	Specification Guidelines to avoid the State Space Explosion Problem
10/15	Daniel Trivellato, Nicola Zannone and Sandro Etalle	GEM: a Distributed Goal Evaluation Algorithm for Trust Management
10/16	L. Aceto, M. Cimini, A.Ingolfsdottir, M.R. Mousavi and M. A. Reniers	Rule Formats for Distributivity
10/17	L. Aceto, A. Birgisson, A. Ingolfsdottir, and M.R. Mousavi	Decompositional Reasoning about the History of Parallel Processes
10/18	P.D. Mosses, M.R. Mousavi and M.A. Reniers	Robustness os Behavioral Equivalence on Open Terms
10/19	Harsh Beohar and Pieter Cuijpers	Desynchronisability of (partial) closed loop systems
11/01	Kees M. van Hee, Natalia Sidorova and Jan Martijn van der Werf	Refinement of Synchronizable Places with Multi-workflow Nets - Weak termination preserved!
11/02	M.F. van Amstel, M.G.J. van den Brand and L.J.P. Engelen	Using a DSL and Fine-grained Model Transformations to Explore the boundaries of Model Verification
11/03	H.R. Mahrooghi and M.R. Mousavi	Reconciling Operational and Epistemic Approaches to the Formal Analysis of Crypto-Based Security Protocols
11/04	J.F. Groote, A.A.H. Osaiweran and J.H. Wesselius	Benefits of Applying Formal Methods to Industrial Control Software
11/05	Jan Friso Groote and Jan Lanik	Semantics, bisimulation and congruence results for a general stochastic process operator
11/06	P.J.L. Cuijpers	Moore-Smith theory for Uniform Spaces through Asymptotic Equivalence
11/07	F.P.M. Stappers, M.A. Reniers and S. Weber	Transforming SOS Specifications to Linear Processes
11/08	Debjoyti Bera, Kees M. van Hee, Michiel van Osch and Jan Martijn van der Werf	A Component Framework where Port Compatibility Implies Weak Termination
11/09	Tseesuren Batsuuri, Reinder J. Bril and Johan Lukkien	Model, analysis, and improvements for inter-vehicle communication using one-hop periodic broadcasting based on the 802.11p protocol

11/10	Neda Noroozi, Ramtin Khosravi, MohammadReza Mousavi and Tim A.C. Willemse	Synchronizing Asynchronous Conformance Testing
11/11	Jeroen J.A. Keiren and Michel A. Reniers	Type checking mCRL2
11/12	Muhammad Atif, MohammadReza Mousavi and Ammar Osaiweran	Formal Verification of Unreliable Failure Detectors in Partially Synchronous Systems
11/13	J.F. Groote, A.A.H. Osaiweran and J.H. Wesseliuss	Experience report on developing the Front-end Client unit under the control of formal methods
11/14	J.F. Groote, A.A.H. Osaiweran and J.H. Wesseliuss	Analyzing a Controller of a Power Distribution Unit Using Formal Methods
11/15	John Businge, Alexander Serebrenik and Mark van den Brand	Eclipse API Usage: The Good and The Bad
11/16	J.F. Groote, A.A.H. Osaiweran, M.T.W. Schuts and J.H. Wesseliuss	Investigating the Effects of Designing Control Software using Push and Poll Strategies
11/17	M.F. van Amstel, A. Serebrenik And M.G.J. van den Brand	Visualizing Traceability in Model Transformation Compositions
11/18	F.P.M. Stappers, M.A. Reniers, J.F. Groote and S. Weber	Dogfooding the Structural Operational Semantics of mCRL2
12/01	S. Cranen	Model checking the FlexRay startup phase
12/02	U. Khadim and P.J.L. Cuijpers	Appendix C / G of the paper: Repairing Time-Determinism in the Process Algebra for Hybrid Systems ACP
12/03	M.M.H.P. van den Heuvel, P.J.L. Cuijpers, J.J. Lukkien and N.W. Fisher	Revised budget allocations for fixed-priority-scheduled periodic resources
12/04	Ammar Osaiweran, Tom Fransen, Jan Friso Groote and Bart van Rijnsoever	Experience Report on Designing and Developing Control Components using Formal Methods
12/05	Sjoerd Cranen, Jeroen J.A. Keiren and Tim A.C. Willemse	A cure for stuttering parity games
12/06	A.P. van der Meer	CIF MSOS type system
12/07	Dirk Fahland and Robert Prüfer	Data and Abstraction for Scenario-Based Modeling with Petri Nets
12/08	Luc Engelen and Anton Wijs	Checking Property Preservation of Refining Transformations for Model-Driven Development
12/09	M.M.H.P. van den Heuvel, M. Behnam, R.J. Bril, J.J. Lukkien and T. Nolte	Opaque analysis for resource-sharing components in hierarchical real-time systems - extended version -
12/10	Milosh Stolikj, Pieter J. L. Cuijpers and Johan J. Lukkien	Efficient reprogramming of sensor networks using incremental updates and data compression
12/11	John Businge, Alexander Serebrenik and Mark van den Brand	Survival of Eclipse Third-party Plug-ins
12/12	Jeroen J.A. Keiren and Martijn D. Klabbers	Modelling and verifying IEEE Std 11073-20601 session setup using mCRL2
12/13	Ammar Osaiweran, Jan Friso Groote, Mathijs Schuts, Jozef Hooman and Bart van Rijnsoever	Evaluating the Effect of Formal Techniques in Industry
12/14	Ammar Osaiweran, Mathijs Schuts, and Jozef Hooman	Incorporating Formal Techniques into Industrial Practice
13/01	S. Cranen, M.W. Gazda, J.W. Wesselink and T.A.C. Willemse	Abstraction in Parameterised Boolean Equation Systems