Structural Analysis of Boolean Equation Systems

JEROEN J. A. KEIREN and MICHEL A. RENIERS and TIM A.C. WILLEMSE
Eindhoven University of Technology

We analyse the problem of solving Boolean equation systems through the use of *structure graphs*. The latter are obtained through an elegant set of Plotkin-style deduction rules. Our main contribution is that we show that equation systems with bisimilar structure graphs have the same solution. We show that our work conservatively extends earlier work, conducted by Keiren and Willemse, in which *dependency graphs* were used to analyse a subclass of Boolean equation systems, *viz.*, equation systems in *standard recursive form*. We illustrate our approach by a small example, demonstrating the effect of simplifying an equation system through minimisation of its structure graph.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic; D.2.4 [**Software Engineering**]: Software/Program Verification; D.2.4 [**Software Engineering**]: Software/Program Verification; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

General Terms: Theory, Verification

Additional Key Words and Phrases: Boolean Equation Systems, Structure Graphs, Bisimilarity

1. INTRODUCTION

A *Boolean equation system* [Larsen 1993; Mader 1997] — equation system for short — is a sequence of fixed-point equations, in which all equations range over the Boolean lattice. The interest in equation systems has both practical and theoretical origins.

Equation systems have been used as a uniform framework for solving traditional verification problems such as the celebrated *model checking* problem [Mader 1997] and a variety of *behavioural equivalence checking* problems, see [Mateescu 2003; 2006; Chen et al. 2007]; this has led to effective tooling, see *e.g.* [Garavel et al. 2007; Groote et al. 2009]. The size of the resulting equation system is dependent on the input and the verification problem: for instance, the global μ-calculus model checking problem $L \models \phi$, where L is a state space and ϕ a formula can be made to yield equation systems $\mathsf{E}^L(\phi)$ of size $\mathcal{O}(|L| \cdot |\phi|)$, where $|L|$ is the size of the state space and $|\phi|$ the size of the modal formula. As a result, the encoding to equation systems suffers from a phenomenon akin to the state space explosion problem.

From a theoretical stance, the problem of solving an equation system is intriguing: it is in NP $\cap$ co-NP, see, *e.g.* [Mader 1997]. In fact, the problem of solving an equation system is equivalent to the problem of computing the winner in a *Parity*

Game [Zielonka 1998]. The latter has been shown to be in UP ∩ co-UP, see [Jurdziński 1998]. This makes the problem of solving an equation system a favourable candidate for finding a polynomial time algorithm, if it exists. Currently, the algorithm with the best worst-case time complexity for solving Parity Games, and thereby equation systems, is the *bigstep* algorithm [Schewe 2007]. This algorithm has running-time complexity $\mathcal{O}(n \cdot m^{d/3})$, where n corresponds to the number of vertices, m to the number of edges and d to the number of priorities in the Parity Game (or equivalently, the number of equations, the cumulative size of the right-hand sides and the number of fixed-point sign alternations in an equation system, respectively).

The running-time complexity of the algorithms for solving equation systems provides a practical motivation for investigating methods for efficiently reducing the size of equation systems. In the absence of notions such as a behaviour of an equation system, an unorthodox strategy in this setting is the use of bisimulation minimisation techniques. Nevertheless, recent work [Keiren and Willemse 2009] demonstrates that such minimisations are practically cost-effective: they yield massive reductions of the size of equation systems, they do not come with memory penalties, and the time required for solving the original equation system significantly exceeds the time required for minimisation and subsequent solving of the minimised equation system.

In *ibid.*, the minimisations are only obtained for a strict subclass of equation systems, *viz.*, equation systems in *standard recursive form (SRF)*. The minimisation technique relies on a bisimulation minimisation for a variation of *dependency graphs* [Mader 1997; Keinänen 2006] underlying the equation systems in SRF. Such graphs basically reflect the (possibly mutual) dependencies of the equations in an equation system in SRF. It is noteworthy that the transformation of an equation system into SRF (henceforth referred to as the process of *normalising*) is a linear-time process.

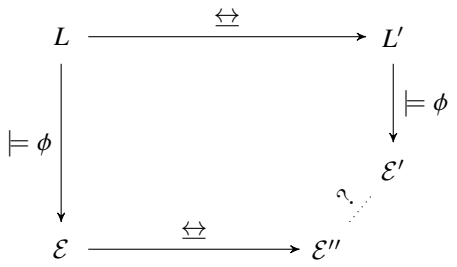
From a practical viewpoint, the class of equation systems in SRF does not pose any limitations to the applicability of the method: normalising an equation system does not change the solution to the proposition variables of the original equation system, and the transformation comes at the cost of only a linear blow-up in size. Its effects on the minimising capabilities of bisimulation, however, are unknown, leading to the first question:

1. What is the effect of normalising an equation system on the minimising capabilities of bisimulation? In other words: how does the size of the *bisimulation quotient* of an equation system compare to the size of the bisimulation quotient of its normalised counterpart?

We answer this question in favour of the process of normalisation: the size of the quotient of the normalised equation system will be at most the size of the quotient of the original equation system (see Theorem 4.6). In addition, we provide an example (see Example 4.7) in which the quotient is strictly smaller in size.

It is well-known that the modal μ-calculus is preserved under bisimulation minimisation of the behavioural state space. As the size of the BES encoding a model checking problem is proportional to the size of the state space, minimising the

state space prior to verification (by whatever global method) can be a useful pre-computation step, provided that the state space is available (in some methodologies, BESs are generated from symbolic state spaces, see *e.g.* [Groote and Willemse 2005]). However, it is unknown whether state space minimisation and minimisation of equation systems encoding a model checking problem are comparable, see also the picture below.



This naturally leads to the second question:

2. Do bisimilar states in a state space give rise to bisimilar equations in the equation systems encoding model checking problems?

The answer to this question is stated by Proposition 6.2, confirming that pairs of bisimilar states in some state space L induce equations in $\mathcal{E}$ that can also be related through an appropriate bisimulation relation underlying the equation system encoding the model checking problem $L \models \phi$. This result remains valid when considering 'safe' abstractions on the original state space. This is shown in Theorem 6.6. We moreover provide an example, see Example 6.7, in which we show that the bisimulation reduction of equation systems can be arbitrarily larger than the reduction of state spaces, even in the presence of safe abstractions.

The main problem in obtaining our results, and answering the above questions, is that it is hard to elegantly capture the structure of an equation system, without resulting in a parse-tree of the equation system. As a matter of fact, bisimilarity is required to reflect associativity and commutativity of Boolean operators such as $\wedge$ and $\vee$ in order to obtain our aforementioned second result; this cannot be achieved using parse-trees. In addition, the arbitrary nesting levels of Boolean operators in equation systems complicate a straightforward definition of bisimilarity for such general equation systems. We solve these issues by using a set of deduction rules in Plotkin style [Plotkin 2004] to map the equation systems onto *structure graphs*. The latter generalise the aforementioned dependency graphs by dropping the requirement that each vertex necessarily represents a proposition variable occurring at the left-hand side of some equation and adding facilities for reasoning about Boolean constants true and false, and unbound variables.

Related Work. Various types of graphs for equation systems have appeared in the literature. We review some of the more relevant types of graphs below.

Boolean Graphs are introduced in [Andersen 1994], in an attempt to use graphs for representing the (implicit) equation systems (in simple form), underlying model

checking problems obtained by verifying μ-calculus formulae on state spaces. Equations are represented by vertices, and dependencies on variables are represented by the edges. In addition, each vertex is labelled with either $\vee$ or $\wedge$, representing the fact that the right-hand side of the equation is disjunctive or conjunctive, respectively. On the basis of the graph representation, Andersen describes the first on-the-fly model checking algorithm for alternation-free equation systems.

The on-the-fly techniques by Andersen are generalised to the full modal μ-calculus in [Liu et al. 1998]. The graphs underlying the latter approach, called *Partitioned Dependency Graphs*, generalise Andersen's Boolean Graphs, by considering *hyper-edges* from vertices to sets of vertices. [Liu and Smolka 1998] proposes an improvement over the latter approach for the special case of alternation-free equation systems, using *dependency graphs*. The latter simplify the Partitioned Dependency Graphs, and, at the same time, generalise the Boolean Graphs of Andersen, giving rise to simpler equation system resolution algorithms. In addition, Liu and Smolka show that their dependency graphs are useful for solving Horn clauses.

Keinänen extends the Boolean Graphs of Andersen by decorating each vertex, in addition to the labelling with $\wedge$ or $\vee$, with a natural number that abstractly represents the fixed-point sign of the equation, see [Keinänen 2006]. Also these graphs are referred to as *dependency graphs*. In [Keiren and Willemse 2009], the latter type of graphs is used to investigate two notions of bisimulation, *viz.*, *strong bisimulation*, and a weakened variation thereof, called *idempotence-identifying bisimulation*, and their theoretical and practical use for minimising equation systems in SRF.

The dependency graphs of Keinänen are closely related to *Parity Games* (see *e.g.* [Zielonka 1998]) and the games proposed by Stirling (see *e.g.* [Stirling 1997; Stevens and Stirling 1998]), in which players aim to win an infinite game. It has been shown on several occasions that the latter problem is equivalent to solving an equation system. Stirling's game graphs were implemented in various tools, most notably in the Concurrency Workbench.

Simulation relations for Parity Games have been studied in, among others [Fritz and Wilke 2006]. Finally, we mention the framework of *Switching Graphs* [Groote and Ploeger 2009], which have two kinds of edges: ordinary edges and *switches*, which can be set to one of two destinations. Switching Graphs are more general than the dependency graphs of [Keinänen 2006], but are still inadequate for directly capturing the structure of the entire class of equation systems. Note that in the Switching Graph setting, the *v-parity loop problem* is equivalent to the problem of solving Boolean equation systems.

The current paper extends and improves upon preliminary work presented in [Reniers and Willemse 2010]. The structure graphs, introduced in *ibid.*, and further studied in this paper, generalise the discussed graphs by being capable of representing arbitrary nestings of $\wedge$ and $\vee$ in the right-hand sides of the equations, and providing the facilities to reason about equation systems with free variables and constants. The main sources of inspiration for structure graphs are the Stirling and Parity Games, and the dependency graphs of Keinänen.

Outline. For completeness, in Section 2, we briefly describe the formal settings, illustrating the model checking problem and how this problem can be translated to the problem of solving an equation system. Section 3 subsequently introduces

structure graphs and the deduction rules for generating these from an equation system. Our main results are presented in Sections 4–6. An application of our theory can be found in Section 7. Section 8 summarises our results and outlines future work.

2. PRELIMINARIES

Henceforth we assume the existence of two sufficiently large, disjoint, countable sets of proposition variables $\mathcal{X}$ and $\tilde{\mathcal{X}}$.

2.1 The Modal μ-Calculus

Labelled transition systems provide a formal, semantical model for the behaviour of a reactive system. While, in this paper, we are mostly concerned with Boolean equation systems, our work is motivated by the model checking problem, *i.e.*, the problem of deciding whether a given behavioural specification satisfies a temporal or modal formula. For this reason, we first repeat some basic results from the latter setting and illustrate its connection to the problem of solving Boolean equation systems.

Definition 2.1. A *labelled transition system* is a three-tuple $L = \langle S, Act, \rightarrow \rangle$, consisting of a finite, non-empty set of states S, a finite, non-empty set of actions Act and a transition relation $\rightarrow \subseteq S \times Act \times S$.

We visualise labelled transition systems by directed, edge-labelled graphs. In line with this graphical notation, we write $s \xrightarrow{a} s'$ if and only if $(s, a, s') \in \rightarrow$. The *de facto* behavioural equivalence relation for labelled transition systems is *strong bisimilarity*, see [Park 1981].

Definition 2.2. Let $L = \langle S, Act, \rightarrow \rangle$ be a labelled transition system. A symmetric relation $R \subseteq S \times S$ is a *strong bisimulation* if for all $(s, s') \in R$

$$\forall a \in Act, t \in S : \; s \xrightarrow{a} t \implies \exists t' \in S : \; s' \xrightarrow{a} t' \wedge (t, t') \in R$$

States $s, s' \in S$ are *bisimilar* if and only if there is a bisimulation relation R that relates states s and s'.

The *propositional modal μ-calculus*, see [Kozen 1983] is a highly-expressive language for analysing behaviours that are defined through a labelled transition system. We refrain from going into details, but solely present its grammar and semantics below. For an accessible treatment of the modal μ-calculus, we refer to [Bradfield and Stirling 2001].

Definition 2.3. Let Act be a finite set of actions. The set of modal μ-calculus formulae is defined through the following grammar, which is given directly in positive form:

$$\phi, \psi ::= \mathsf{true} \mid \mathsf{false} \mid \tilde{X} \mid \phi \wedge \psi \mid \phi \vee \psi \mid [A]\phi \mid \langle A \rangle \phi \mid \nu \tilde{X}.\phi \mid \mu \tilde{X}.\phi$$

where $\tilde{X} \in \tilde{\mathcal{X}}$ is a proposition variable; $A \subseteq Act$ is a set of actions; μ is a least fixed point sign and ν is a greatest fixed point sign. Throughout this paper we write σ to denote an arbitrary fixed point sign μ or ν.

Note that our use of generalised modal operators $[A]\phi$ and $\langle A\rangle\phi$ is merely for reasons of convenience, and has no implications for the presented theory in this paper. Henceforth, we write $[a]\phi$ instead of $[\{a\}]\phi$ and $[\overline{a}]\phi$ instead of $[Act \setminus \{a\}]\phi$.

In a formula $\sigma\tilde{X}.\phi$, each occurrence of the variable $\tilde{X}$ is *bound*. A variable $\tilde{X}$ is bound in a formula ϕ if all its occurrences are bound. The set of bound proposition variables in ϕ is denoted $\mathsf{bnd}(\phi)$; the set of proposition variables that syntactically occur in ϕ is denoted $\mathsf{occ}(\phi)$. Formula ϕ is said to be *closed* if and only if $\mathsf{occ}(\phi) \subseteq \mathsf{bnd}(\phi)$. We only consider μ-calculus formulae ϕ that are *well-formed*, *i.e.*:

(1) there are no two distinct subformulae of ϕ that bind the same proposition variable;
(2) for every free proposition variable $\tilde{X} \in \mathsf{occ}(\phi) \setminus \mathsf{bnd}(\phi)$, no subformula $\sigma\tilde{X}.\psi$ (binding $\tilde{X}$ locally), occurs in ϕ.

The well-formedness requirement is a technicality and does not incur a loss of generality of the theory.

Modal μ-calculus formulae ϕ are *interpreted* in the context of a labelled transition system and an *environment* $\theta : \tilde{\mathcal{X}} \to 2^S$ that assigns sets of states to proposition variables. We write $\theta[\tilde{X} := S']$ to represent the environment in which $\tilde{X}$ receives the value S', and all other proposition variables have values that coincide with those given by θ.

Definition 2.4. Let $L = \langle S, Act, \to\rangle$ be a labelled transition system and let $\theta : \tilde{\mathcal{X}} \to 2^S$ be a proposition environment. The semantics of a μ-calculus formula ϕ is defined inductively as follows:

$$
\begin{array}{lcl}
[\![\mathsf{true}]\!]\theta & = & S \\
[\![\mathsf{false}]\!]\theta & = & \emptyset \\
[\![\tilde{X}]\!]\theta & = & \theta(\tilde{X}) \\
[\![\phi \wedge \psi]\!]\theta & = & [\![\phi]\!]\theta \cap [\![\psi]\!]\theta \\
[\![\phi \vee \psi]\!]\theta & = & [\![\phi]\!]\theta \cup [\![\psi]\!]\theta \\
[\![[A]\phi]\!]\theta & = & \{s \in S \mid \forall s' \in S : \forall a \in A : \ s \xrightarrow{a} s' \implies s' \in [\![\phi]\!]\theta\} \\
[\![\langle A\rangle\phi]\!]\theta & = & \{s \in S \mid \exists s' \in S : \exists a \in A : \ s \xrightarrow{a} s' \wedge s' \in [\![\phi]\!]\theta\} \\
[\![\nu\tilde{X}.\phi]\!]\theta & = & \bigcup\{S' \subseteq S \mid S' \subseteq [\![\phi]\!]\theta[\tilde{X} := S']\} \\
[\![\mu\tilde{X}.\phi]\!]\theta & = & \bigcap\{S' \subseteq S \mid [\![\phi]\!]\theta[\tilde{X} := S'] \subseteq S'\}
\end{array}
$$

The *global* model checking problem, denoted $L, \theta \models \phi$, is defined as the question whether for all states $s \in S$ of a given labelled transition system $L = \langle S, Act, \to\rangle$, we have $s \in [\![\phi]\!]\theta$, for given formula ϕ and environment θ. The *local* model checking problem, denoted $L, s, \theta \models \phi$, is the problem whether $s \in [\![\phi]\!]\theta$ for a given state $s \in S$. Often, one is only interested in *closed* formulae. Small examples of typical model checking problems can be found in the remainder of this paper.

2.2 Boolean Equation Systems

A Boolean equation system is a finite sequence of least and greatest fixed point equations, where each right-hand side of an equation is a proposition formula. For

an excellent, in-depth account on Boolean equation systems, we refer to [Mader 1997].

Definition 2.5. A *Boolean equation system (BES)* $\mathcal{E}$ is defined by the following grammar:

$$\begin{array}{lll} \mathcal{E} & ::= & \epsilon \mid (\nu X = f)\ \mathcal{E} \mid (\mu X = f)\ \mathcal{E} \\ f, g & ::= & \mathsf{true} \mid \mathsf{false} \mid X \mid f \wedge g \mid f \vee g \end{array}$$

where ϵ is the empty BES; $X \in \mathcal{X}$ is a proposition variable; and f, g are proposition formulae.

We only consider equation systems that are *well-formed*, *i.e.*, equation systems $\mathcal{E}$, in which a proposition variable X occurs at the left-hand side in at most a single equation in $\mathcal{E}$.

In line with the notions of bound and occurring proposition variables for μ-calculus formulae, we introduce analogue notions for equation systems. Let $\mathcal{E}$ be an arbitrary equation system. The set of *bound* proposition variables of $\mathcal{E}$, denoted $\mathsf{bnd}(\mathcal{E})$, is the set of variables occurring at the left-hand side of the equations in $\mathcal{E}$. The set of *occurring* proposition variables, denoted $\mathsf{occ}(\mathcal{E})$, is the set of variables occurring at the right-hand side of some equation in $\mathcal{E}$.

An equation system $\mathcal{E}$ is said to be *closed* whenever $\mathsf{occ}(\mathcal{E}) \subseteq \mathsf{bnd}(\mathcal{E})$. Intuitively, a (closed) equation system uniquely assigns truth values to its bound proposition variables. An equation system is said to be in *simple form* [Arnold and Crubille 1988] if none of the right-hand sides of the equations that occur in the equation system contain both $\wedge$- and $\vee$-operators. If such an equation system, in addition, has no occurrences of the constants true and false in its right-hand sides, it is said to be in *standard recursive form* [Keiren and Willemse 2009].

Proposition variables occurring in a proposition formula f are collected in the set $\mathsf{occ}(f)$. The *rank* of a proposition variable $X \in \mathsf{bnd}(\mathcal{E})$, notation $\mathsf{rank}_{\mathcal{E}}(X)$, is defined as follows:

$$\mathsf{rank}_{(\sigma Y = f)\mathcal{E}}(X) = \begin{cases} \mathsf{rank}_{\mathcal{E}}(X) & \text{if } X \neq Y \\ \mathsf{block}_{\sigma}(\mathcal{E}) & \text{otherwise} \end{cases}$$

where $\mathsf{block}_{\sigma}(\mathcal{E})$ is defined as:

$$\mathsf{block}_{\sigma}(\epsilon) = \begin{cases} 0 & \text{if } \sigma = \nu \\ 1 & \text{otherwise} \end{cases} \qquad \mathsf{block}_{\sigma}((\sigma' Y = f)\mathcal{E}) = \begin{cases} \mathsf{block}_{\sigma}(\mathcal{E}) & \text{if } \sigma = \sigma' \\ 1 + \mathsf{block}_{\sigma'}(\mathcal{E}) & \text{if } \sigma \neq \sigma' \end{cases}$$

Informally, the rank of a variable X is the i-th block of like-signed equations, containing X's defining equation, counting from right-to-left and starting at 0 if the last equation is a greatest fixed point sign, and 1 otherwise.

Formally, proposition formulae are interpreted in the context of an *environment* $\eta{:}\mathcal{X} \to \mathbb{B}$. For an arbitrary environment η, we write $\eta[X := b]$ for the environment η in which the proposition variable X has Boolean value b and all other proposition variables X' have value $\eta(X')$. The ordering $\sqsubseteq$ on environments is defined as $\eta \sqsubseteq \eta'$ if and only if $\eta(X)$ implies $\eta'(X)$ for all X. For reading ease, we do not formally distinguish between a semantic Boolean value and its representation by true and false; likewise, for the operands $\wedge$ and $\vee$.

Definition 2.6. Let $\eta{:}\mathcal{X} \to \mathbb{B}$ be an environment. The *interpretation* $[\![f]\!]\eta$ maps a proposition formula f to true or false:

$$\begin{aligned} [\![X]\!]\eta &= \eta(X) \\ [\![\mathsf{true}]\!]\eta &= \mathsf{true} & [\![f \wedge g]\!]\eta &= [\![f]\!]\eta \wedge [\![g]\!]\eta \\ [\![\mathsf{false}]\!]\eta &= \mathsf{false} & [\![f \vee g]\!]\eta &= [\![f]\!]\eta \vee [\![g]\!]\eta \end{aligned}$$

The *solution of a BES*, given an environment η, is inductively defined as follows:

$$\begin{aligned} [\![\epsilon]\!]\eta &= \eta \\ [\![(\sigma X = f)\ \mathcal{E}]\!]\eta &= \begin{cases} [\![\mathcal{E}]\!](\eta[X := [\![f]\!]([\![\mathcal{E}]\!]\eta[X := \mathsf{false}])]) & \text{if } \sigma = \mu \\ [\![\mathcal{E}]\!](\eta[X := [\![f]\!]([\![\mathcal{E}]\!]\eta[X := \mathsf{true}])]) & \text{if } \sigma = \nu \end{cases} \end{aligned}$$

We refer to [Mader 1997, Section 3.2] for an explanation of the nature of this definition. A solution to an equation system verifies every equation, in the sense that the value at the left-hand side is logically equivalent to the value at the right-hand side of the equation. At the same time, the fixed-point signs of left-most equations *outweigh* the fixed-point signs of those equations that follow, *i.e.*, the fixed-point signs of left-most equations are more important. The latter phenomenon is a result of the nested recursion for evaluating the proposition f of the left-most equation $(\sigma X = f)$, assuming an extremal value for X. As a consequence, the solution is order-sensitive: the solution to $(\mu X = Y)\ (\nu Y = X)$, yielding all false, differs from the solution to $(\nu Y = X)\ (\mu X = Y)$, yielding all true. It is exactly this tree-like recursive definition of a solution that makes it intricately complex.

Closed equation systems enjoy the property that the solution to the equation system is independent of the environment in which it is defined, *i.e.*, for all environments η, η', we have $[\![\mathcal{E}]\!]\eta(X) = [\![\mathcal{E}]\!]\eta'(X)$ for all $X \in \mathsf{bnd}(\mathcal{E})$. For this reason, we henceforth refrain from writing the environment explicitly in all our considerations dealing with closed equation systems, *i.e.*, we write $[\![\mathcal{E}]\!]$, and $[\![\mathcal{E}]\!](X)$ instead of the more verbose $[\![\mathcal{E}]\!]\eta$ and $[\![\mathcal{E}]\!]\eta(X)$.

The following lemma relates the semantics for open equation systems to that of closed equation systems. We write $\mathcal{E}[X := b]$, where $X \notin \mathsf{bnd}(\mathcal{E})$ and $b \in \{\mathsf{true}, \mathsf{false}\}$ is a constant, to denote the equation system in which each syntactic occurrence of X is replaced by b.

LEMMA 2.7. *Let $\mathcal{E}$ be an equation system, and let η be an arbitrary environment. Assume $X \notin \mathsf{bnd}(\mathcal{E})$ is a proposition variable, and let b be such that $\eta(X) = [\![b]\!]$. Then $[\![\mathcal{E}]\!]\eta = [\![\mathcal{E}[X := b]]\!]\eta$.*

PROOF. We show this by induction on the size of $\mathcal{E}$. The base case for $\mathcal{E} = \epsilon$ follows immediately. As our induction hypothesis, we take

$$\forall \eta, b, X \notin \mathsf{bnd}(\mathcal{E}) :\ [\![b]\!] = \eta(X) \implies [\![\mathcal{E}]\!]\eta = [\![\mathcal{E}[X := b]]\!]\eta \qquad \text{(IH)}$$

Assume our induction hypothesis holds for $\mathcal{E}$, and let η and b be such that $[\![b]\!] = \eta(X)$. Consider the equation system $(\nu Y = f)\ \mathcal{E}$, and assume $X \notin \mathsf{bnd}((\nu Y = f)\ \mathcal{E})$.

Using the semantics of equation systems, we reason as follows:

$$\begin{array}{ll} & [\![(\nu Y = f)\mathcal{E}]\!]\eta \\ = & [\![\mathcal{E}]\!]\eta[Y := [\![f]\!]([\![\mathcal{E}]\!]\eta[Y := \mathsf{true}])] \\ =^{2\times IH} & [\![\mathcal{E}[X := b]]\!]\eta[Y := [\![f]\!]([\![\mathcal{E}[X := b]]\!]\eta[Y := \mathsf{true}])] \\ =^{\ddagger} & [\![\mathcal{E}[X := b]]\!]\eta[Y := [\![f[X := b]]\!]([\![\mathcal{E}[X := b]]\!]\eta[Y := \mathsf{true}])] \\ = & [\![((\nu Y = f)\ \mathcal{E})\ [X := b]]\!]\eta \end{array}$$

where at ‡, we have used that $[\![f]\!]\eta = [\![f[X := b]]\!]\eta$ for $[\![b]\!] = \eta(X)$. The case for $(\mu Y = f)\ \mathcal{E}$ follows the exact same line of reasoning and is therefore omitted. □

Finally, we introduce some generic shorthand notation. The operators $\bigsqcap$ and $\bigsqcup$ are used as shorthands for nested applications of $\wedge$ and $\vee$. Formally, these are defined as follows. Let $\lessdot$ be a total order on $\mathcal{X} \cup \{\mathsf{true}, \mathsf{false}\}$. Assuming that $\lessdot$ is lifted to a total ordering on formulae, we define for formula f $\lessdot$-smaller than all formulae in a finite, non-empty set F ($f \notin F$):

$$\bigsqcap \emptyset = \mathsf{true} \qquad \bigsqcap\{f\} = f \wedge f \qquad \bigsqcap(\{f\} \cup F) = f \wedge \left(\bigsqcap F\right)$$

$$\bigsqcup \emptyset = \mathsf{false} \qquad \bigsqcup\{f\} = f \vee f \qquad \bigsqcup(\{f\} \cup F) = f \vee \left(\bigsqcup F\right)$$

Note that the duplication introduced by this definition does not have any semantic influence.

In a similar fashion, we define an equation system obtained from a set of equations. Let $X = f$ be an equation, where f is a proposition formula and X is a proposition variable. Assuming that X is $\lessdot$-smaller than all left-hand side variables in the equations in a finite set of equations E, we define:

$$\sigma\{X = f\} = (\sigma X = f) \qquad \sigma(\{X = f\} \cup E) = (\sigma X = f)\sigma E$$

2.3 Boolean Equation Systems for Model Checking

An obvious strategy for solving a typical model checking problem is through the use of Tarski's approximation schemes for computing the solution to the fixed points of monotone operators in a complete lattice, see *e.g.* [Tarski 1955]. More advanced techniques employ intermediate formalisms such as Boolean equation systems for solving the verification problem.

Below, we provide the translation of the model checking problem to the problem of solving a Boolean equation system. The transformer E reduces the global model checking problem $L, \eta \models \phi$ to the problem of solving an equation system.

Definition 2.8. Assume $L = \langle S, Act, \rightarrow \rangle$ is a labelled transition system. Let ϕ be an arbitrary modal μ-calculus formula over Act. Suppose that for every proposition variable $\tilde{X} \in \mathsf{occ}(\phi) \cup \mathsf{bnd}(\phi)$, we have a set of fresh proposition variables $\{X_s \mid s \in$

$S\} \subseteq \mathcal{X}$.

$$
\begin{array}{lcl}
\mathsf{E}^L(b) & = & \epsilon \\
\mathsf{E}^L(\tilde{X}) & = & \epsilon \\
\mathsf{E}^L(f \wedge g) & = & \mathsf{E}^L(f)\ \mathsf{E}^L(g) \\
\mathsf{E}^L(f \vee g) & = & \mathsf{E}^L(f)\ \mathsf{E}^L(g) \\
\mathsf{E}^L([A]f) & = & \mathsf{E}^L(f) \\
\mathsf{E}^L(\langle A\rangle f) & = & \mathsf{E}^L(f) \\
\mathsf{E}^L(\sigma\tilde{X}.\ f) & = & (\sigma\{(X_s = \mathsf{RHS}_s(f)) \mid s \in S\})\ \mathsf{E}^L(f)
\end{array}
$$

$$
\begin{array}{lcl}
\mathsf{RHS}_s(b) & = & b \\
\mathsf{RHS}_s(\tilde{X}) & = & X_s \\
\mathsf{RHS}_s(f \wedge g) & = & \mathsf{RHS}_s(f) \wedge \mathsf{RHS}_s(g) \\
\mathsf{RHS}_s(f \vee g) & = & \mathsf{RHS}_s(f) \vee \mathsf{RHS}_s(g) \\
\mathsf{RHS}_s([A]f) & = & \bigsqcap\{\mathsf{RHS}_t(f) \mid a \in A, s \xrightarrow{a} t\} \\
\mathsf{RHS}_s(\langle A\rangle f) & = & \bigsqcup\{\mathsf{RHS}_t(f) \mid a \in A, s \xrightarrow{a} t\} \\
\mathsf{RHS}_s(\sigma\tilde{X}.\ f) & = & X_s
\end{array}
$$

Observe that the definition of E provided here coincides (semantically) with the definition given in [Mader 1997] for modal μ-calculus formulae ϕ; the only deviation is a syntactic one, ensuring that the $[\_]$ and $\langle\_\rangle$ modalities are mapped onto proposition formulae with $\wedge$, and $\vee$ as their main logical connectives in case there is a non-empty set of emanating transitions.

The relation between the original local model checking problem and the problem of solving a Boolean equation system is stated by the theorem below.

THEOREM 2.9 [MADER 1997]. *Assume $L = \langle S, Act, \rightarrow\rangle$ is a labelled transition system. Let $\sigma\tilde{X}.f$ be an arbitrary modal μ-calculus formula, and let θ be an arbitrary environment. Then:*

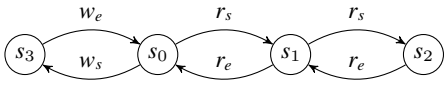
$$L, s, \theta \models \sigma\tilde{X}.f \text{ if and only if } (\llbracket \mathsf{E}^L(\sigma\tilde{X}.f)\rrbracket\eta)(X_s) = \mathsf{true}$$

where for all proposition variables $Y_t \in \{Z_s \mid s \in S \wedge \tilde{Z} \in \mathsf{occ}(\sigma\tilde{X}.f) \cup \mathsf{bnd}(\sigma\tilde{X}.f)\}$, we set $\eta(Y_t) = \mathsf{true}$ if and only if $t \in \theta(\tilde{Y})$, and false for all other proposition variables.

Informally, the theorem expresses that a state s satisfies a modal μ-calculus formula $\sigma\tilde{X}.f$ if, and only if the associated proposition variable X_s in the equation system $\mathsf{E}^L(\sigma\tilde{X}.f)$ has true as its solution. The environment η ensures that free proposition variables are correctly dealt with. The correspondence between the global model checking problem and the solution to an equation system then follows immediately from the latter's correspondence to the local model checking problem.

The example below illustrates the above translation and theorem.

Example 2.10. Consider the labelled transition system (depicted below), modelling mutual exclusion between two readers and a single writer.



Reading is started using an action r_s and action r_e indicates its termination. Likewise for writing. The verification problem $\nu\tilde{X}.\mu\tilde{Y}.\ \langle r_s\rangle\tilde{X} \vee \langle \overline{r_s}\rangle\tilde{Y}$, modelling that on

some path, a reader can infinitely often start reading, translates to the following equation system using the translation E:

$$
\begin{array}{l}
(\nu X_{s_0} = Y_{s_0})\ (\nu X_{s_1} = Y_{s_1})\ (\nu X_{s_2} = Y_{s_2})\ (\nu X_{s_3} = Y_{s_3}) \\
(\mu Y_{s_0} = (X_{s_1} \vee X_{s_1}) \vee (Y_{s_3} \vee Y_{s_3})) \\
(\mu Y_{s_1} = (X_{s_2} \vee X_{s_2}) \vee (Y_{s_0} \vee Y_{s_0})) \\
(\mu Y_{s_2} = \mathsf{false} \vee (Y_{s_1} \vee Y_{s_1})) \\
(\mu Y_{s_3} = \mathsf{false} \vee (Y_{s_0} \vee Y_{s_0}))
\end{array}
$$

Observe that, like the original μ-calculus formula, which has mutual dependencies between $\tilde{X}$ and $\tilde{Y}$, the resulting equation system has mutual dependencies between the indexed X and Y variables. Solving the resulting equation system leads to true for all bound variables; $X_{s_i} = \mathsf{true}$, for arbitrary state s_i, implies that the property holds in state s_i. Furthermore, note that the right-hand sides of the resulting equation system can be rewritten using standard rules of logic, removing, *e.g.*, all occurrences of false. It is not hard to check that this does not affect the solution to the equation system.

3. STRUCTURE GRAPHS FOR BOOLEAN EQUATION SYSTEMS

A large part of the complexity of equation systems is attributed to the mutual dependencies between the equations. These intricate dependencies are neatly captured by *structure graph*s. In Section 3.1, we define how a structure graph can be obtained from a formula in the context of an equation system. In Section 3.2, we define how an equation system can be associated with a structure graph assuming that it satisfies some well-formedness constraints.

Definition 3.1. Given a set of proposition variables $\mathcal{X}$. A structure graph over $\mathcal{X}$ is a vertex-labelled graph $\mathcal{G} = \langle T, t, \rightarrow, d, r, \nearrow \rangle$, where:

—T is a finite set of vertices;
—$t \in T$ is the initial vertex;
—$\rightarrow \subseteq T \times T$ is a dependency relation;
—$d{:}T \mapsto \{\blacktriangle, \blacktriangledown, \top, \bot\}$ is a vertex decoration mapping;
—$r{:}T \mapsto \mathbb{N}$ is a vertex ranking mapping;
—$\nearrow : T \mapsto \mathcal{X}$ is a free variable mapping.

A structure graph can be used to capture the dependencies between bound variables and (sub)formulae occurring in the equations of such bound variables. Intuitively, the decoration mapping d reflects whether the top symbol of a proposition formula is true (represented by $\top$), false (represented by $\bot$), a conjunction (represented by $\blacktriangle$), or a disjunction (represented by $\blacktriangledown$). The vertex ranking mapping r indicates the rank of a vertex. The free variable mapping indicates whether a vertex represents a free variable. Note that each vertex can have at most one rank, at most one decoration $\star \in \{\blacktriangle, \blacktriangledown, \top, \bot\}$, and at most one free variable $\nearrow_X$. We sometimes write t to refer to a structure graph $\langle T, t, \rightarrow, d, r, \nearrow \rangle$, where t is in fact the initial vertex of the structure graph.

We define the size of a structure graph $\mathcal{G} = \langle T, t, \rightarrow, d, r, \nearrow \rangle$ as $|\mathcal{G}| = |T|$, *i.e.*, the size of a structure graph is the number of vertices in the graph.

One can easily define bisimilarity on structure graphs.

Definition 3.2. Let $\mathcal{G} = \langle T, t, \rightarrow, d, r, \nearrow \rangle$ and $\mathcal{G}' = \langle T', t', \rightarrow', d', r', \nearrow' \rangle$ be structure graphs. A relation $R \subseteq T \times T'$ is a bisimulation relation if for all $(u, u') \in R$

—$d(u) = d'(u')$, $r(u) = r'(u')$, and $\nearrow(u) = \nearrow'(u')$;
—for all $v \in T$, if $u \rightarrow v$, then $u' \rightarrow' v'$ for some $v' \in T'$ such that $(v, v') \in R$;
—for all $v' \in T'$, if $u' \rightarrow' v'$, then $u \rightarrow v$ for some $v \in T$ such that $(v, v') \in R$.

Two vertices u and u' are bisimilar, notation $u \leftrightarrows u'$ if there exists a bisimulation relation R such that $(u, u') \in R$.

Using this notion of bisimilarity, we also define the *bisimulation quotient* of a structure graph.

Definition 3.3. Let $\mathcal{G} = \langle T, t, \rightarrow, d, r, \nearrow \rangle$ be a structure graph. The bisimulation quotient $\mathcal{G}_{/\leftrightarrows} = \langle T', t', \rightarrow', d', r', \nearrow' \rangle$ of $\mathcal{G}$ is defined as follows:

—$T' = T_{/\leftrightarrows} = \{[t_i]_{/\leftrightarrows} \mid t_i \in T\}$ with $[t_i]_{/\leftrightarrows} = \{t_j \in T \mid t_i \leftrightarrows t_j\}$;
—$t' = [t]_{/\leftrightarrows}$;
—$\rightarrow'$ is defined by: $\dfrac{t_i \rightarrow t_j}{[t_i]_{/\leftrightarrows} \rightarrow' [t_j]_{/\leftrightarrows}}$
—$d'([t_i]_{/\leftrightarrows}) = d(t_i)$, if $t_i \in \mathsf{dom}(d)$, and undefined otherwise;
—$r'([t_i]_{/\leftrightarrows}) = r(t_i)$, if $t_i \in \mathsf{dom}(r)$, and undefined otherwise;
—$\nearrow'([t_i]_{/\leftrightarrows}) = \nearrow(t_i)$, if $t_i \in \mathsf{dom}(\nearrow)$, and undefined otherwise.

3.1 Structured Operational Semantics for equation systems

Next, we define structure graphs for arbitrary equation systems $\mathcal{E}$ and proposition formulae t. We use Plotkin-style Structural Operational Semantics [Plotkin 2004] to associate a structure graph with a formula f in the context of a equation system $\mathcal{E}$, notation $\langle f, \mathcal{E} \rangle$. The deduction rules define a relation $\_ \rightarrow \_$ and predicates $\_ \Pi n$ (for $n \in \mathbb{N}$), $\_ \nearrow_X$ (for $X \in \mathcal{X}$), $\_\top$, $\_\bot$, $\_\blacktriangle$, and $\_\blacktriangledown$. In the deduction rules also negative premises are used, see [Mousavi et al. 2005] for an overview.

The notations used in the deduction rules are slightly different from those used in the structure graphs. The predicate $t \nearrow_X$ represents $\nearrow(t) = X$, the predicate $t \Pi n$ represents $r(t) = n$, for $\star \in \{\blacktriangle, \blacktriangledown, \top, \bot\}$, $t\star$ represents $d(t) = \star$. The notation $t \not\Pi$ represents $\neg(t \Pi n)$ for all $n \in \mathbb{N}$.

First, as we are dealing with possibly open equation systems, free variables are labelled as such:

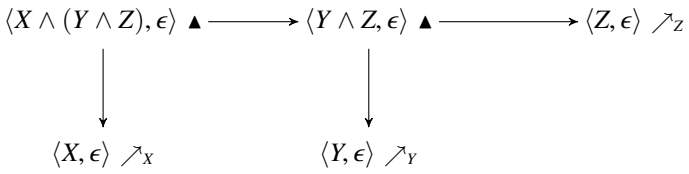
$$(1)\ \frac{X \in \mathsf{occ}(\mathcal{E}) \setminus \mathsf{bnd}(\mathcal{E})}{\langle X, \mathcal{E} \rangle \nearrow_X}$$

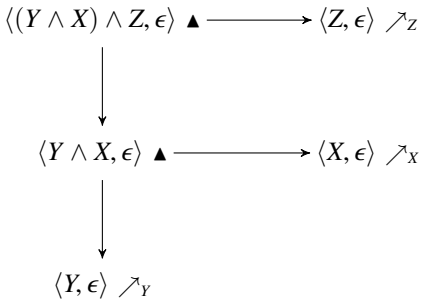
In addition, vertices representing bound proposition variables are labelled by a natural number representing the rank of the variable in the equation system:

$$(2)\ \frac{X \in \mathsf{bnd}(\mathcal{E})}{\langle X, \mathcal{E} \rangle \Pi \mathsf{rank}_{\mathcal{E}}(X)}$$

In Boolean equation systems, the right-hand sides are built up of binary conjunctions and disjunctions. A question that needs to be answered is 'How to capture

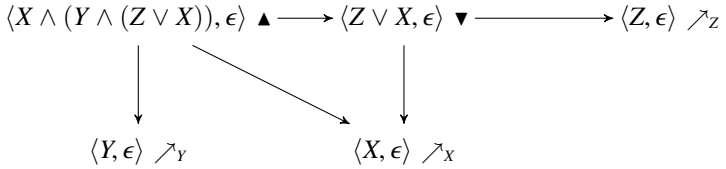
this structure in the structure graph?' One way of doing so would be to precisely reflect the structure of the proposition formula. For a formula of the form $X \wedge (Y \wedge Z)$ in the context of an empty equation system this results in the first structure graph depicted below:





A drawback of this solution is that, in general, the logical equivalence between $X \wedge (Y \wedge Z)$ and $(Y \wedge X) \wedge Z$ (see the second structure graph above) is not reflected by bisimilarity. Retaining this logical equivalence (and hence associativity and commutativity) of both conjunction and disjunction is desirable.

The logical connectives for conjunction ($\wedge$) and disjunction ($\vee$) may occur nested in a formula. This is solved by reflecting a change in leading operator in the structure graph. So the anticipated structure of the structure graph for $X \wedge (Y \wedge (Z \vee X))$, where we assume that the equation system contains no equations, is:



This can be elegantly achieved by means of the following deduction rules for the decorations and the dependency transition relation $\rightarrow$:

$$(3)\ \frac{}{\langle \mathsf{true}, \mathcal{E}\rangle \top} \qquad (4)\ \frac{}{\langle \mathsf{false}, \mathcal{E}\rangle \bot} \qquad (5)\ \frac{}{\langle f \wedge f', \mathcal{E}\rangle \blacktriangle} \qquad (6)\ \frac{}{\langle f \vee f', \mathcal{E}\rangle \blacktriangledown}$$

$$(7)\ \frac{\langle f, \mathcal{E}\rangle \blacktriangle \quad \langle f, \mathcal{E}\rangle \not\sigma \quad \langle f, \mathcal{E}\rangle \rightarrow \langle g, \mathcal{E}\rangle}{\langle f \wedge f', \mathcal{E}\rangle \rightarrow \langle g, \mathcal{E}\rangle} \qquad (8)\ \frac{\langle f', \mathcal{E}\rangle \blacktriangle \quad \langle f', \mathcal{E}\rangle \not\sigma \quad \langle f', \mathcal{E}\rangle \rightarrow \langle g', \mathcal{E}\rangle}{\langle f \wedge f', \mathcal{E}\rangle \rightarrow \langle g', \mathcal{E}\rangle}$$

$$(9)\ \frac{\langle f,\mathcal{E}\rangle\blacktriangledown \quad \langle f,\mathcal{E}\rangle \not\Cap \quad \langle f,\mathcal{E}\rangle \to \langle g,\mathcal{E}\rangle}{\langle f \vee f',\mathcal{E}\rangle \to \langle g,\mathcal{E}\rangle} \qquad (10)\ \frac{\langle f',\mathcal{E}\rangle\blacktriangledown \quad \langle f',\mathcal{E}\rangle \not\Cap \quad \langle f',\mathcal{E}\rangle \to \langle g',\mathcal{E}\rangle}{\langle f \vee f',\mathcal{E}\rangle \to \langle g',\mathcal{E}\rangle}$$

$$(11)\ \frac{\neg\langle f,\mathcal{E}\rangle\blacktriangle}{\langle f \wedge f',\mathcal{E}\rangle \to \langle f,\mathcal{E}\rangle} \qquad (12)\ \frac{\neg\langle f',\mathcal{E}\rangle\blacktriangle}{\langle f \wedge f',\mathcal{E}\rangle \to \langle f',\mathcal{E}\rangle}$$

$$(13)\ \frac{\neg\langle f,\mathcal{E}\rangle\blacktriangledown}{\langle f \vee f',\mathcal{E}\rangle \to \langle f,\mathcal{E}\rangle} \qquad (14)\ \frac{\neg\langle f',\mathcal{E}\rangle\blacktriangledown}{\langle f \vee f',\mathcal{E}\rangle \to \langle f',\mathcal{E}\rangle}$$

$$(15)\ \frac{\langle f,\mathcal{E}\rangle \Cap n}{\langle f \wedge f',\mathcal{E}\rangle \to \langle f,\mathcal{E}\rangle} \qquad (16)\ \frac{\langle f',\mathcal{E}\rangle \Cap n}{\langle f \wedge f',\mathcal{E}\rangle \to \langle f',\mathcal{E}\rangle}$$

$$(17)\ \frac{\langle f,\mathcal{E}\rangle \Cap n}{\langle f \vee f',\mathcal{E}\rangle \to \langle f,\mathcal{E}\rangle} \qquad (18)\ \frac{\langle f',\mathcal{E}\rangle \Cap n}{\langle f \vee f',\mathcal{E}\rangle \to \langle f',\mathcal{E}\rangle}$$

Rules (3-6) describe the axioms for decoration. The first four deduction rules (7-10) for $\to$ are introduced to flatten the nesting hierarchy of the same connective. They can be used to deduce that $X \wedge (Y \wedge Z) \to Y$. Deduction rules 7-10 work for the situation that the subformula has a $\blacktriangle$ or $\blacktriangledown$ but that this is not caused by a recursion variable (see the second premise of the deduction rules in combination with deduction rules 19 and 20). Deduction rules 11-18 describe the dependencies in case there is no flattening possible anymore (by absence of structure). The deduction rules 11-14 deal with the case that a subformula has no $\blacktriangle$ or $\blacktriangledown$. The deduction rules 15-18 deal with the case that the subformula represents a bound variable.

Finally, we present deduction rules that describe how the structure of a vertex representing a variable is derived from the right-hand side of the corresponding equation. Observe that the deduction rules only have to deal with the case that a defining equation for the recursion variable X has been found in the Boolean equation system. Deduction rules 19 and 20 define the predicates $\blacktriangle$ and $\blacktriangledown$ for the case that the right-hand side is a variable or a constant. Deduction rules 21 and 22 define the dependency relation $\to$ for the case that the right-hand side is a variable or a constant. Deduction rules 23 and 24 do this for the cases in which the right-hand side is a proposition formula that is neither a variable nor a constant.

$$(19)\ \frac{\sigma X = f \in \mathcal{E} \quad \langle f,\mathcal{E}\rangle\blacktriangle \quad \langle f,\mathcal{E}\rangle \not\Cap}{\langle X,\mathcal{E}\rangle\blacktriangle} \qquad (20)\ \frac{\sigma X = f \in \mathcal{E} \quad \langle f,\mathcal{E}\rangle\blacktriangledown \quad \langle f,\mathcal{E}\rangle \not\Cap}{\langle X,\mathcal{E}\rangle\blacktriangledown}$$

$$(21)\ \frac{\sigma X = f \in \mathcal{E} \quad \neg\langle f,\mathcal{E}\rangle\blacktriangledown \quad \neg\langle f,\mathcal{E}\rangle\blacktriangle}{\langle X,\mathcal{E}\rangle \to \langle f,\mathcal{E}\rangle} \qquad (22)\ \frac{\sigma X = f \in \mathcal{E} \quad \langle f,\mathcal{E}\rangle \Cap n}{\langle X,\mathcal{E}\rangle \to \langle f,\mathcal{E}\rangle}$$

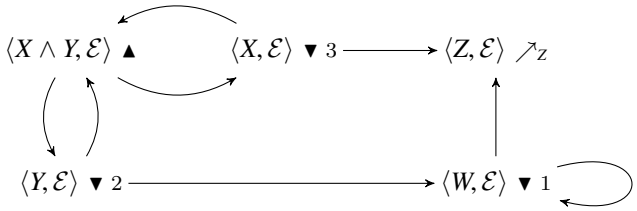
$$(23)\ \frac{\sigma X = f \in \mathcal{E} \quad \langle f,\mathcal{E}\rangle \to \langle g,\mathcal{E}\rangle \quad \langle f,\mathcal{E}\rangle\blacktriangle \quad \langle f,\mathcal{E}\rangle \not\Cap}{\langle X,\mathcal{E}\rangle \to \langle g,\mathcal{E}\rangle}$$

$$(24)\ \frac{\sigma X = f \in \mathcal{E} \quad \langle f, \mathcal{E}\rangle \to \langle g, \mathcal{E}\rangle \quad \langle f, \mathcal{E}\rangle \blacktriangledown \quad \langle f, \mathcal{E}\rangle \not\Uparrow}{\langle X, \mathcal{E}\rangle \to \langle g, \mathcal{E}\rangle}$$

An SOS that is defined using negative premises is not necessarily well-defined [Groote 1993]. In case one can provide a stratification, i.e., a mapping from transitions and predicates to ordinals such that for any closed instance of every deduction rule the positive premises are not larger than the conclusion and (the positive instances of) the negative premises are strictly smaller than the conclusion, the SOS defines a collection of transition relations and predicates uniquely. In this case, providing such a stratification is easy. As long as all transitions are larger than all predicates and the predicates $\blacktriangle$ and $\blacktriangledown$ are larger than $\Uparrow$ predicates, the SOS is stratified.

Example 3.4. An equation system $\mathcal{E}$ (see left) and its associated structure graph (see right). Observe that the term $X \wedge Y$ is shared by the equations for X and Y, and appears only once in the structure graph as an unranked vertex. There is no equation for Z; this is represented by term Z, decorated only by the label $\nearrow_Z$. The subterm $Z \vee W$ in the equation for W does not appear as a separate vertex in the structure graph, since the disjunctive subterm occurs within the scope of another disjunction.

$$\begin{array}{lcl} \mu X & = & (X \wedge Y) \vee Z \\ \nu Y & = & W \vee (X \wedge Y) \\ \mu W & = & Z \vee (Z \vee W) \end{array}$$



Given a formula f and an equation system $\mathcal{E}$, $\langle f, \mathcal{E}\rangle$ denotes the part of the structure graph generated by the deduction rules that is reachable from the vertex $\langle f, \mathcal{E}\rangle$.

LEMMA 3.5. *Let $\mathcal{E}$ be an equation system. Let f, f', g and g' be arbitrary proposition formulae such that $\langle f, \mathcal{E}\rangle \leftrightarrow \langle f', \mathcal{E}\rangle$ and $\langle g, \mathcal{E}\rangle \leftrightarrow \langle g', \mathcal{E}\rangle$. Then the following hold:*

$$\langle f \wedge g, \mathcal{E}\rangle \leftrightarrow \langle f' \wedge g', \mathcal{E}\rangle, \quad \langle f \vee g, \mathcal{E}\rangle \leftrightarrow \langle f' \vee g', \mathcal{E}\rangle$$

PROOF. Suppose that bisimilarity of $\langle f, \mathcal{E}\rangle$ and $\langle f', \mathcal{E}\rangle$ is witnessed by R and the bisimilarity of $\langle g, \mathcal{E}\rangle$ and $\langle g', \mathcal{E}\rangle$ is witnessed by S. The relation $\{(\langle f \wedge g, \mathcal{E}\rangle, \langle f' \wedge g', \mathcal{E}\rangle)\} \cup R \cup S$ is a bisimulation relation that proves bisimilarity of $\langle f \wedge g, \mathcal{E}\rangle$ and $\langle f' \wedge g', \mathcal{E}\rangle$. Similarly, $\{(\langle f \vee g, \mathcal{E}\rangle, \langle f' \vee g', \mathcal{E}\rangle)\} \cup R \cup S$ is a bisimulation relation that proves bisimilarity of $\langle f \vee g, \mathcal{E}\rangle$ and $\langle f' \vee g', \mathcal{E}\rangle$. □

The following lemma indicates that bisimilarity on structure graphs respects logical equivalences such as commutativity, associativity and a weak form of idempotence for the $\wedge$ and $\vee$ operators.

LEMMA 3.6. *Let $\mathcal{E}$ be an equation system. Let f, f', and f'' be arbitrary propo-*

sition formulae. Then the following hold:

$$\begin{array}{rcl}
\langle (f \wedge f') \wedge f'', \mathcal{E}\rangle & \leftrightarrows & \langle f \wedge (f' \wedge f''), \mathcal{E}\rangle, \\
\langle (f \vee f') \vee f'', \mathcal{E}\rangle & \leftrightarrows & \langle f \vee (f' \vee f''), \mathcal{E}\rangle, \\
\langle f \wedge f', \mathcal{E}\rangle & \leftrightarrows & \langle f' \wedge f, \mathcal{E}\rangle, \\
\langle f \vee f', \mathcal{E}\rangle & \leftrightarrows & \langle f' \vee f, \mathcal{E}\rangle, \\
\langle (f \wedge f) \wedge f', \mathcal{E}\rangle & \leftrightarrows & \langle f \wedge f', \mathcal{E}\rangle, \\
\langle (f \vee f) \vee f', \mathcal{E}\rangle & \leftrightarrows & \langle f \vee f', \mathcal{E}\rangle
\end{array}$$

PROOF. The proofs are easy. For example, the bisimulation relation that witnesses bisimilarity of $\langle (f \wedge f') \wedge f'', \mathcal{E}\rangle$ and $\langle f \wedge (f' \wedge f''), \mathcal{E}\rangle$ is the relation that relates all formulae of the form $\langle (g \wedge g') \wedge g'', \mathcal{E}\rangle$ and $\langle g \wedge (g' \wedge g''), \mathcal{E}\rangle$ and additionally contains the identity relation on structure graphs. Proofs of the 'transfer conditions' are easy as well. As an example, suppose that $\langle (g \wedge g') \wedge g'', \mathcal{E}\rangle \to \langle h, \mathcal{E}\rangle$ for some formula h. In case this transition is due to $\langle g \wedge g', \mathcal{E}\rangle\blacktriangle$ and $\langle g \wedge g', \mathcal{E}\rangle \to \langle h, \mathcal{E}\rangle$, one of the cases that occurs for $\langle g \wedge g', \mathcal{E}\rangle \to \langle h, \mathcal{E}\rangle$ is that $\langle g, \mathcal{E}\rangle\blacktriangle$ and $\langle g, \mathcal{E}\rangle \to \langle h, \mathcal{E}\rangle$. We obtain $\langle g \wedge (g' \wedge g''), \mathcal{E}\rangle \to \langle h, \mathcal{E}\rangle$. Since $\langle h, \mathcal{E}\rangle$ and $\langle h, \mathcal{E}\rangle$ are related, this finishes the proof of the transfer condition in this case. All other cases are similar or at least equally easy. □

COROLLARY 3.7. *Let $\mathcal{E}$ be an equation system. Let F and G be arbitrary finite sets of proposition formulae such that (1) for all $f \in F$ there exists $g \in G$ with $\langle f, \mathcal{E}\rangle \leftrightarrows \langle g, \mathcal{E}\rangle$, and, vice versa, (2) for all $g \in G$ there exists $f \in F$ with $\langle g, \mathcal{E}\rangle \leftrightarrows \langle f, \mathcal{E}\rangle$. Then, $\langle \bigsqcap F, \mathcal{E}\rangle \leftrightarrows \langle \bigsqcap G, \mathcal{E}\rangle$ and $\langle \bigsqcup F, \mathcal{E}\rangle \leftrightarrows \langle \bigsqcup G, \mathcal{E}\rangle$.*

PROOF. The corollary follows immediately from the congruence of $\wedge$ and $\vee$ (Lemma 3.5) and commutativity and associativity of those (Lemma 3.6). □

Idempotence of $\wedge$ and $\vee$, and more involved logical equivalences such as distribution and absorption are not captured by isomorphism or even bisimilarity on the structure graphs. The reason is that, for an arbitrary equation system $\mathcal{E}$ and variable X, the vertex associated with $\langle X \wedge X, \mathcal{E}\rangle$ will be decorated by ▲, in contrast to the vertex associated with $\langle X, \mathcal{E}\rangle$!

3.2 Translating Structure Graphs to Equation Systems

Next, we show how, under some mild conditions, a formula and equation system can be obtained from a structure graph. Later in the paper this transformation will be used and proved correct.

A structure graph $\mathcal{G} = \langle T, t, \to, d, r, \nearrow\rangle$ is called *BESsy* if it satisfies the following constraints:

—a vertex t decorated by $\top, \bot$ or $\nearrow_X$ for some X has no successor w.r.t. $\to$.
—a vertex is decorated by ▲ or ▼ or a rank if and only if it has a successor w.r.t. $\to$.
—a vertex with multiple successors w.r.t. $\to$, is decorated with ▲ or ▼.
—every cycle contains a vertex with a rank.

Observe that BESsyness is preserved under bisimilarity:

LEMMA 3.8. *Let $\mathcal{G}$ and $\mathcal{G}'$ be bisimilar structure graphs. Then, $\mathcal{G}$ is BESsy if, and only if, $\mathcal{G}'$ is BESsy.*

PROOF. This follows immediately from the transfer conditions of bisimilarity. □

The following lemma states that any structure graph obtained from a formula and an equation system is BESsy.

LEMMA 3.9. *For any formula f and equation system $\mathcal{E}$, the structure graph $\langle f, \mathcal{E}\rangle$ is BESsy.*

PROOF. We have to establish that the structure graph $\langle f, \mathcal{E}\rangle$ is BESsy. Thereto it has to be shown that the four requirements of the definition of BESsyness are satisfied.

The first one trivially follows by considering all the possibilities for generating a vertex labelled by either $\top$, $\bot$, or $\nearrow_X$. In each case it turns out that f is of a form that does not allow the derivation of a $\rightarrow$-transition.

The proof of the second requirement requires induction on the depth of the proof of $\langle f, \mathcal{E}\rangle\blacktriangle$, $\langle f, \mathcal{E}\rangle\blacktriangledown$, or $\langle f, \mathcal{E}\rangle$ $\Mapsto$, respectively. Inside this induction there is a case distinction on the deduction rule that has been applied last in the proof.

For the proof of the third requirement it suffices to consider all possibilities for generating multiple successors and it follows easily that in these cases the vertex is also labelled by $\blacktriangle$ or $\blacktriangledown$.

The last requirement follows trivially from the observation that a cycle of successor relations can never be generated without using a bound variable along the cycle. This would inevitably introduce a rank for that vertex. □

For a BESsy structure graph $\mathcal{G} = \langle T, t, \rightarrow, d, r, \nearrow\rangle$ the function φ is defined as follows: for $u \in T$

$$\varphi(u) = \begin{cases} \bigsqcap\{\varphi(u') \mid u \rightarrow u'\} & \text{if } d(u) = \blacktriangle \text{ and } u \notin \mathrm{dom}(r), \\ \bigsqcup\{\varphi(u') \mid u \rightarrow u'\} & \text{if } d(u) = \blacktriangledown \text{ and } u \notin \mathrm{dom}(r), \\ \mathsf{true} & \text{if } d(u) = \top, \\ \mathsf{false} & \text{if } d(u) = \bot, \\ X & \text{if } \nearrow(u) = X, \\ X_u & \text{otherwise.} \end{cases}$$

The function φ introduces variables for those vertices that are in the domain of the vertex rank mapping or the free variable mapping. In the second case, the associated variable name is used. In the former case, a fresh variable name is introduced to represent the vertex. For other vertices the structure that is offered via vertex decoration mapping d is used to obtain a formula representing such a structure.

Definition 3.10. Let $\mathcal{G} = \langle T, t, \rightarrow, d, r, \nearrow\rangle$ be a BESsy structure graph. The equation system associated to $\mathcal{G}$, denoted $\beta(\mathcal{G})$, is defined below.

To each vertex $u \in T$ such that $u \in \mathrm{dom}(r)$, we associate an equation of the form:

$$\sigma X_u = \mathsf{rhs}(u)$$

Here σ is μ in case the rank associated to the vertex is odd, and ν otherwise. $\mathsf{rhs}(u)$

is defined as follows:

$$\mathsf{rhs}(u) = \begin{cases} \bigsqcap\{\varphi(u') \mid u \to u'\} & \text{if } d(u) = \blacktriangle \\ \bigsqcup\{\varphi(u') \mid u \to u'\} & \text{if } d(u) = \blacktriangledown \\ \varphi(u') & \text{if } d(u) \neq \blacktriangle, d(u) \neq \blacktriangledown, \text{ and } u \to u' \end{cases}$$

The equation system $\beta(\mathcal{G})$ is obtained by ordering the equations from left-to-right ensuring the ranks of the vertices associated to the equations are descending.

We next show the correspondence between a BES and the BES obtained from its structure graph. First, given a BES $\mathcal{E}$ we show the correspondence between the right hand side of an equation in $\mathcal{E}$, and the right hand side obtained from the structure graph of $\mathcal{E}$.

PROPOSITION 3.11. *Let $\mathcal{E}$ be a BES such that $\sigma Y = f \in \mathcal{E}$. Then for all environments η for which $\eta(Z) = \eta(X_{\langle Z,\mathcal{E}\rangle})$ for all $Z \in \mathsf{bnd}(\mathcal{E})$, we have $[\![f]\!]\eta = [\![\mathsf{rhs}(\langle Y, \mathcal{E}\rangle)]\!]\eta$.*

PROOF. We prove this using a distinction on the cases of $\mathsf{rhs}(\langle Y, \mathcal{E}\rangle)$. The proof involves a number of lemmata expressing distribution laws of φ over Boolean connectives $\wedge$ and $\vee$, as well as the relation between f and $\varphi(\langle f, \mathcal{E}\rangle)$ for arbitrary formulae f. These lemmata in turn require proofs involving case distinctions on the SOS rules, and induction on formulae. The required lemmata, as well as a detailed proof of this proposition (rephrased as Proposition A.4) can be found in the appendix. □

Next we show that evaluating a formula f in a BES $\mathcal{E}$, and evaluating the formula $\varphi(\langle f, \mathcal{E}\rangle)$ in the BES $\beta(\langle f, \mathcal{E}\rangle)$ are equivalent.

THEOREM 3.12. *Let $\mathcal{E}$ be a BES and η an environment. Then for all formulae f it holds that $[\![f]\!][\![\mathcal{E}]\!]\eta = [\![\varphi(\langle f, \mathcal{E}\rangle)]\!][\![\beta(\langle f, \mathcal{E}\rangle)]\!]\eta$.*

PROOF. Let $\mathcal{F}$ abbreviate the equation system $\beta(\langle f, \mathcal{E}\rangle)$, and abbreviate the formula $\varphi(\langle f, \mathcal{E}\rangle)$ by g. Note that by construction, $\mathcal{F}$ consists of equations of the form $\sigma X_{\langle Z,\mathcal{E}\rangle} = \mathsf{rhs}(\langle Z, \mathcal{E}\rangle)$, for $Z \in \mathsf{bnd}(\mathcal{E})$.

Denote the free-variable closure of $\mathcal{E}$, $\mathcal{F}$, f and g, using η by $\mathcal{E}_c$, $\mathcal{F}_c$, f_c and g_c, respectively. According to Lemma 2.7, we have $[\![\mathcal{E}_c]\!] = [\![\mathcal{E}]\!]\eta$, and $[\![\mathcal{F}_c]\!] = [\![\mathcal{F}]\!]\eta$, and, likewise, $[\![f]\!][\![\mathcal{E}]\!]\eta = [\![f_c]\!][\![\mathcal{E}_c]\!]$ and $[\![g]\!][\![\mathcal{F}]\!]\eta = [\![g_c]\!][\![\mathcal{F}_c]\!]$. Let $\mathcal{E}'$ be the equation system obtained by merging all equations of $\mathcal{E}_c$ and $\mathcal{F}_c$, such that:

(1) $\mathsf{rank}_{\mathcal{E}'}(X) = \mathsf{rank}_{\mathcal{E}}(X)$ for all $X \in \mathsf{bnd}(\mathcal{E}_c)$;
(2) $\mathsf{rank}_{\mathcal{E}'}(X_{\langle Z,\mathcal{E}\rangle}) = \mathsf{rank}_{\mathcal{E}}(Z)$ for all $X_{\langle Z,\mathcal{E}\rangle} \in \mathsf{bnd}(\mathcal{F}_c)$.

Observe that the resulting $\mathcal{E}'$ is well-formed, since we have

$$\mathsf{bnd}(\mathcal{E}_c) \cap \mathsf{bnd}(\mathcal{F}_c) = \emptyset$$

Moreover, since

$$\mathsf{bnd}(\mathcal{E}_c) \cap \mathsf{occ}(\mathcal{F}_c) = \mathsf{bnd}(\mathcal{F}_c) \cap \mathsf{occ}(\mathcal{E}_c) = \emptyset$$

also $[\![\mathcal{E}']\!] = [\![\mathcal{E}_c]\!][\![\mathcal{F}_c]\!]$, *i.e.*, we can find the solution to $\mathcal{E}_c$ and $\mathcal{F}_c$ by solving $\mathcal{E}'$. We rely on [Willemse 2010, Theorem 2] for proving $[\![\mathcal{E}']\!](Z) = [\![\mathcal{E}']\!](X_{\langle Z,\mathcal{E}\rangle})$. For this, we must construct a relation $R \subseteq \mathsf{bnd}(\mathcal{E}') \times \mathsf{bnd}(\mathcal{E}')$, such that $X\ R\ Y$ implies

—$\mathsf{rank}_{\mathcal{E}'}(X) = \mathsf{rank}_{\mathcal{E}'}(Y)$;
—for all $\theta \in \{\eta \mid U\ R\ V \implies \eta(U) = \eta(V)\}$, we have $[\![f_X]\!]\theta = [\![f_Y]\!]\theta$, where f_X and f_Y are the right-hand sides of the equations for X and Y.

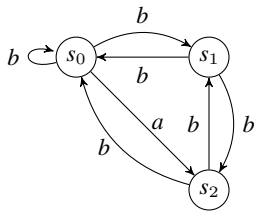
Given such a relation R, we can conclude that $[\![\mathcal{E}']\!](X) = [\![\mathcal{E}']\!](Y)$ for all $X\ R\ Y$. Using Proposition 3.11, it is not hard to check that the relation R, defined as

$$R = \{(Z, X_{\langle Z,\mathcal{E}\rangle}), (X_{\langle Z,\mathcal{E}\rangle}, Z) \mid X_{\langle Z,\mathcal{E}\rangle} \in \mathcal{F}_c\}$$

is indeed such a relation. We therefore find that for all $X_{\langle Z,\mathcal{E}\rangle} \in \mathsf{bnd}(\mathcal{F}_c)$, we have $[\![Z]\!][\![\mathcal{E}']\!] = [\![X_{\langle Z,\mathcal{E}\rangle}]\!][\![\mathcal{E}']\!]$. More specifically we find that $[\![Z]\!][\![\mathcal{E}_c]\!] = [\![X_{\langle Z,\mathcal{E}\rangle}]\!][\![\mathcal{F}_c]\!]$. It is not hard to show that then also $[\![f_c]\!][\![\mathcal{E}_c]\!] = [\![g_c]\!][\![\mathcal{F}_c]\!]$. Hence our claim follows. □

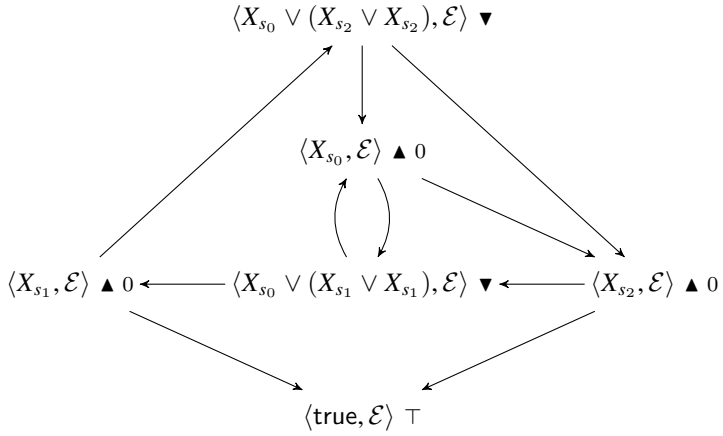
We illustrate the various translations described in this section through the following example.

Example 3.13. Consider the labelled transition system L given below.

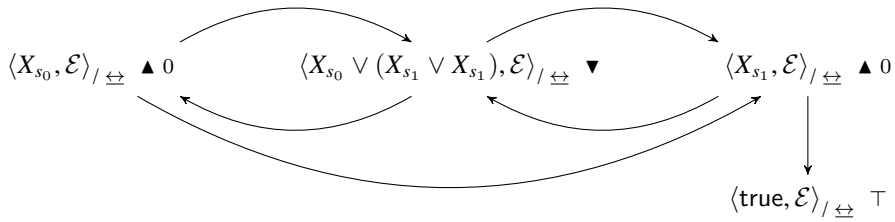


Let $\phi = \nu X.[a]X \wedge \langle b\rangle X$. Consider the following equation system $\mathcal{E} = \mathsf{E}^L(\phi)$, together with the structure graph $\langle X_{s_0}, \mathcal{E}\rangle$:

$$\begin{array}{lcl}(\nu X_{s_0} & = & (X_{s_2} \wedge X_{s_2}) \wedge (X_{s_0} \vee (X_{s_1} \vee X_{s_1})))\\ (\nu X_{s_1} & = & \mathsf{true} \wedge (X_{s_0} \vee (X_{s_2} \vee X_{s_2})))\\ (\nu X_{s_2} & = & \mathsf{true} \wedge (X_{s_0} \vee (X_{s_1} \vee X_{s_1})))\end{array}$$



Observe that the above structure graph can be minimised with respect to bisimilarity, identifying vertices $\langle X_{s_1}, \mathcal{E}\rangle$ and $\langle X_{s_2}, \mathcal{E}\rangle$, as well as $\langle X_{s_0} \vee (X_{s_1} \vee X_{s_1}), \mathcal{E}\rangle$ and $\langle X_{s_0} \vee (X_{s_2} \vee X_{s_2}), \mathcal{E}\rangle$. This leads to the following bisimilar, minimal structure graph:



The above structure graph induces the following equation system, using the translation of Definition 3.10.

$$(\nu X_{\langle X_{s_0},\mathcal{E}\rangle_{/\leftrightarrow}} = (X_{\langle X_{s_0},\mathcal{E}\rangle_{/\leftrightarrow}} \vee (X_{\langle X_{s_1},\mathcal{E}\rangle_{/\leftrightarrow}} \vee X_{\langle X_{s_1},\mathcal{E}\rangle_{/\leftrightarrow}})) \wedge (X_{\langle X_{s_1},\mathcal{E}\rangle_{/\leftrightarrow}} \wedge X_{\langle X_{s_1},\mathcal{E}\rangle_{/\leftrightarrow}}))$$
$$(\nu X_{\langle X_{s_1},\mathcal{E}\rangle_{/\leftrightarrow}} = (X_{\langle X_{s_0},\mathcal{E}\rangle_{/\leftrightarrow}} \vee (X_{\langle X_{s_1},\mathcal{E}\rangle_{/\leftrightarrow}} \vee X_{\langle X_{s_1},\mathcal{E}\rangle_{/\leftrightarrow}})) \wedge (\mathsf{true} \wedge \mathsf{true}))$$

The size of the original structure graph is 6. By comparison, the size of the minimal structure graph is 4. As will become clear in Section 5, solving the above equation system enables one to deduce the solution to the original equation system.

4. NORMALISATION OF STRUCTURE GRAPHS

In BESsy structure graphs, a vertex that is decorated by a rank typically represents a proposition variable that occurs at the left-hand side of some equation in the associated equation system, whereas the non-ranked vertices can occur as subterms in right-hand sides of equations with mixed occurrences of $\wedge$ and $\vee$. Normalisation of a structure graph assigns a rank to each non-ranked vertex that has successors. The net effect of this operation is that the structure graph obtained this way induces an equation system in simple form. In choosing the rank, one has some degree of freedom; an effective and sound strategy is to ensure that all equations in the associated equation system end up in the very last block. This is typically achieved by assigning 0 as a rank.

$$(25)\frac{t▲}{\mathsf{norm}(t)▲} \qquad (26)\frac{t▼}{\mathsf{norm}(t)▼} \qquad (27)\frac{t \to t'}{\mathsf{norm}(t) \to \mathsf{norm}(t')}$$

$$(28)\frac{t⊤}{\mathsf{norm}(t)⊤} \qquad (29)\frac{t\,\bot}{\mathsf{norm}(t)\,\bot} \qquad (30)\frac{t \nearrow_X}{\mathsf{norm}(t) \nearrow_X}$$

$$(31)\frac{t \pitchfork n}{\mathsf{norm}(t) \pitchfork n} \qquad (32)\frac{t \not\pitchfork \quad t \to t'}{\mathsf{norm}(t) \pitchfork 0}$$

The last deduction rule expresses that in case a vertex t does not have a rank, rank 0 is associated to the normalised version of t, provided, of course, that the vertex has a successor. Observe that normalisation preserves BESsyness of the structure graph, *i.e.*, any BESsy structure graph that is normalised again yields a BESsy structure graph.

PROPERTY 4.1. *Let t be an arbitrary BESsy structure graph.*

(1) $\varphi(\mathsf{norm}(t)) \in \mathcal{X} \cup \{\mathsf{true}, \mathsf{false}\}$*;*

(2) $\beta(\mathsf{norm}(t))$ *is in simple form;*

(3) $\mathsf{norm}(\mathsf{norm}(t)) \leftrightarrows \mathsf{norm}(t)$.

The well-definedness of the extended SOS is obtained by adapting the stratification from the previous SOS by requiring that $t \Cap n$ is larger than $u \Cap m$ in all cases where the number of occurrences of norm in t is larger than in u.

The lemmata below formalise that the solution to an equation system that is induced by a BESsy structure graph, is preserved and reflected by the equation system associated to the normalised counterpart of that structure graph.

LEMMA 4.2. *Let t be a BESsy structure graph. Then, there is a total injective mapping $h : \mathsf{bnd}(\beta(t)) \to \mathsf{bnd}(\beta(\mathsf{norm}(t)))$, such that for all η:*

$$\forall X \in \mathsf{bnd}(\beta(t)) : [\![\beta(t)]\!]\eta(X) = [\![\beta(\mathsf{norm}(t))]\!]\eta(h(X))$$

PROOF. Observe that for each ranked vertex u in t, vertex $\mathsf{norm}(u)$ has the same rank in $\mathsf{norm}(t)$. Following Definition 3.10, these vertices both induce equations in the equation systems that appear in the same block of identical fixed point equations. All unranked vertices u' in t that are ranked in $\mathsf{norm}(t)$, induce ν-equations at the end of the equation system induced by $\mathsf{norm}(t)$. References to these latter equations can be eliminated, following [Mader 1997, Lemma 6.3]. □

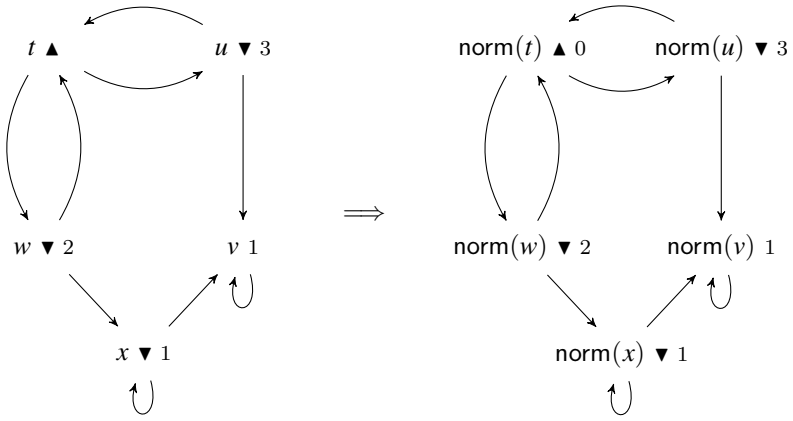
LEMMA 4.3. *Let t be a BESsy structure graph. Then, for all η:*

$$[\![\varphi(t)]\!][\![\beta(t)]\!]\eta = [\![\varphi(\mathsf{norm}(t))]\!][\![\beta(\mathsf{norm}(t))]\!]\eta$$

PROOF. Follows from Lemma 4.2. □

The example below illustrates an application of normalisation, and it provides a demonstration of the above lemmata and its implications.

Example 4.4. The BESsy structure graph depicted at the left contains a single vertex that is not decorated with a rank. Normalisation of this structure graph yields the structure graph depicted at the right.



Assuming that vertex t is the initial vertex, $\beta(t)$ is as follows:

$$\begin{array}{l}(\mu X_u = (X_u \wedge (X_w \wedge X_w)) \vee (X_v \vee X_v))\\ (\nu X_w = (X_u \wedge (X_w \wedge X_w)) \vee (X_x \vee X_x))\\ (\mu X_v = X_v)\\ (\mu X_x = X_v \vee (X_x \vee X_x))\end{array}$$

$\beta(\mathsf{norm}(t))$ has similar top-level logical operands as $\beta(t)$, but contains an extra greatest fixed point equation trailing the other four, and references to this equation:

$$\begin{array}{l}(\mu X_{\mathsf{norm}(u)} = X_{\mathsf{norm}(t)} \vee (X_{\mathsf{norm}(v)} \vee X_{\mathsf{norm}(v)}))\\ (\nu X_{\mathsf{norm}(w)} = X_{\mathsf{norm}(t)} \vee (X_{\mathsf{norm}(x)} \vee X_{\mathsf{norm}(x)}))\\ (\mu X_{\mathsf{norm}(v)} = X_{\mathsf{norm}(v)})\\ (\mu X_{\mathsf{norm}(x)} = X_{\mathsf{norm}(v)} \vee (X_{\mathsf{norm}(x)} \vee X_{\mathsf{norm}(x)}))\\ (\nu X_{\mathsf{norm}(t)} = X_{\mathsf{norm}(u)} \wedge (X_{\mathsf{norm}(w)} \wedge X_{\mathsf{norm}(w)}))\end{array}$$

According to Lemma 4.2, there is an injection $h : \mathsf{bnd}(\beta(t)) \to \mathsf{bnd}(\beta(\mathsf{norm}(t)))$, such that for all $X \in \mathsf{bnd}(\beta(t))$, we have $[\![\beta(t)]\!](X) = [\![\beta(\mathsf{norm}(t))]\!](h(X))$; $h(X_z) = X_{\mathsf{norm}(z)}$ for $z \in \{u, v, w, x\}$ is such an injection. Following Lemma 4.3, we furthermore find $[\![\varphi(t)]\!][\![\beta(t)]\!] = [\![X_u \wedge (X_w \wedge X_w)]\!][\![\beta(t)]\!] = [\![X_{\mathsf{norm}(t)}]\!][\![\beta(\mathsf{norm}(t))]\!] = [\![\varphi(\mathsf{norm}(t))]\!][\![\beta(\mathsf{norm}(t))]\!]$.

The below proposition states that bisimilarity on structure graphs is a congruence for normalisation.

PROPOSITION 4.5. *Let t, t' be arbitrary, but bisimilar structure graphs. Then also* $\mathsf{norm}(t) \leftrightarrows \mathsf{norm}(t')$.

PROOF. Let R be a bisimulation relation witnessing $t \leftrightarrows t'$. We define the relation R_n as $\{(\mathsf{norm}(u), \mathsf{norm}(u')) \mid (u, u') \in R\}$. Then R_n is a bisimulation relation witnessing $\mathsf{norm}(t) \leftrightarrows \mathsf{norm}(t')$. □

Ultimately, the above proposition implies that the simple form is not harmful from a bisimulation perspective: normalisation does not lead to larger quotients of structure graphs. This addresses the hitherto open question concerning the effect of normalisation on the bisimulation reductions of [Keiren and Willemse 2009]. Formally, we have:

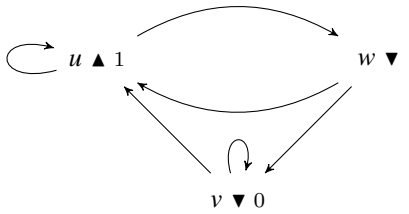
THEOREM 4.6. *Let t be an arbitrary structure graph. Then $t_{/\leftrightarrows}$ is at least as large as* $\mathsf{norm}(t)_{/\leftrightarrows}$.

PROOF. The theorem follows immediately from the fact that $\mathsf{norm}(t)$ and t are equal in size, and Proposition 4.5. □

The example below illustrates that normalisation can in fact sometimes be beneficial for the minimising capabilities of bisimulation.

Example 4.7. Consider the following equation system $\mathcal{E}$, with the associated structure graph $\langle X, \mathcal{E}\rangle$ (represented by vertex u) depicted next to it.

$(\mu X = X \wedge (Y \vee X))$
$(\nu Y = Y \vee X)$



Clearly, the structure graph is already minimal. Normalisation *upcasts* vertex w to a ranked vertex, assigning rank 0 to it. It is then easy to check that vertices $\mathsf{norm}(w)$ and $\mathsf{norm}(v)$ are bisimilar. Hence, the quotient of $\mathsf{norm}(u)$ has size 2, compared to size 3 for u.

5. BISIMILARITY IMPLIES SOLUTION EQUIVALENCE

In this section we state one of our main results, proving that equation systems corresponding to bisimilar BESsy structure graphs essentially have the same solution. This allows one to safely use bisimulation minimisation of the structure graph, and solve the equation system induced by the minimal structure graph instead. Before we give our main theorem, we first lift some known results for equation systems, see *e.g.* [Mader 1997; Keinänen 2006; Keiren and Willemse 2009], to structure graphs.

Definition 5.1. Let $\langle T, t, \rightarrow, d, r, \nearrow \rangle$ be a structure graph. A partial function $\gamma{:}T \mapsto T$ is a $\bullet$-*choice function*, with $\bullet \in \{\blacktriangle, \blacktriangledown\}$, when both:

—$\mathsf{dom}(\gamma) = \{u \in T \mid d(u) = \bullet \wedge u \rightarrow\}$;
—$u \rightarrow \gamma(u)$ for all $u \in \mathsf{dom}(\gamma)$.

Given a $\bullet$-choice function γ, with $\bullet \in \{\blacktriangle, \blacktriangledown\}$, for a structure graph, we can obtain a new structure graph by choosing one successor among the successors for vertices decorated with a $\bullet$, *viz.*, the one prescribed by γ. This is formalised next.

Definition 5.2. Let $\mathcal{G} = \langle T, t, \rightarrow, d, r, \nearrow \rangle$ be an arbitrary structure graph. Let $\bullet \in \{\blacktriangle, \blacktriangledown\}$, and γ a $\bullet$-choice function. The structure graph $\mathcal{G}_\gamma$, obtained by applying the $\bullet$-choice function γ on $\mathcal{G}$, is defined as the six-tuple $\langle T, t, \rightarrow_\gamma, d_\gamma, r, \nearrow \rangle$, where:

—for all $u \notin \mathsf{dom}(\gamma)$, $u \rightarrow_\gamma u'$ if and only if $u \rightarrow u'$;
—for all $u \in \mathsf{dom}(\gamma)$, only $u \rightarrow_\gamma \gamma(u)$;
—$d_\gamma(t) = d(t)$ and $\mathsf{dom}(d_\gamma) = \{u \mid d(u) \neq \bullet\}$

Observe that a structure graph obtained by applying a $\blacktriangle$-choice function entails a structure graph in which no vertex is labelled with $\blacktriangle$. Similarly, applying a $\blacktriangledown$-choice function yields a structure graph without $\blacktriangledown$ labelled vertices.

PROPERTY 5.3. *Let t be an arbitrary BESsy structure graph. Assume an arbitrary $\bullet$-choice function γ on t. Then $\mathsf{norm}(t)_\gamma$ is again BESsy.*

The effect that applying, *e.g.*, a $\blacktriangle$-choice function has on the solution to the equation system associated to the structure graph to which it is applied, is characterised by the proposition below. This result is well-known in the setting of equation systems, see *e.g.* [Mader 1997].

PROPOSITION 5.4. *Let t be a normalised, BESsy structure graph, with no vertex labelled $\nearrow$.*

(1) For all $\blacktriangle$-choice functions γ applied to t, we have $[\![\beta(t)]\!] \sqsubseteq [\![\beta(t_\gamma)]\!]$;
(2) There exists a $\blacktriangle$-choice function γ, such that $[\![\beta(t)]\!] = [\![\beta(t_\gamma)]\!]$.
(3) For all $\blacktriangledown$-choice functions γ applied to t, we have $[\![\beta(t)]\!] \sqsupseteq [\![\beta(t_\gamma)]\!]$;
(4) There exists a $\blacktriangledown$-choice function γ, such that $[\![\beta(t)]\!] = [\![\beta(t_\gamma)]\!]$.

PROOF. Follows immediately from [Mader 1997, Proposition 3.36], and the correspondence between structure graphs and Boolean Equation Systems. □

In some cases, *viz.*, when a structure graph is void of any vertices labelled $\blacktriangledown$ or void of vertices labelled $\blacktriangle$, the solution of an equation system associated to a structure graph can be characterised by the structure of the graph. While one could consider these to be degenerate cases, they are essential in our proof of the main theorem in this section. A key concept used in characterising the solution of equation systems in the degenerate cases is that of a ν-dominated lasso, and its dual, μ-dominated lasso.

Definition 5.5. Let t be a BESsy structure graph. A lasso starting in t is a finite sequence $t_0, t_1, \ldots, t_n$, satisfying $t_0 = t$, $t_n = t_j$ for some $j \leq n$, and for each $1 \leq i \leq n$, $t_{i-1} \to t_i$. A lasso is said to be ν-dominated if $\max\{r(t_i) \mid j \leq i \leq n\}$ is even; otherwise it is μ-dominated.

The following lemma is loosely based on lemmata taken from Keinänen (see Lemmata 40 and 41 in [Keinänen 2006]).

LEMMA 5.6. *Let t be a normalised, BESsy structure graph in which no vertex is labelled with $\nearrow$. Then:*

(1) if no vertex in t is labelled with $\blacktriangle$ then $[\![\varphi(t)]\!][\![\beta(t)]\!] = \mathsf{true}$ if and only if some lasso starting in t is ν-dominated, or some maximal, finite path starting in t terminates in a vertex labelled with $\top$;
(2) if no vertex in t is labelled with $\blacktriangledown$ then $[\![\varphi(t)]\!][\![\beta(t)]\!] = \mathsf{false}$ if and only if some lasso starting in t is μ-dominated, or some maximal, finite path starting in t terminates in a vertex labelled with $\bot$

PROOF. We only consider the first statement; the proof of the second statement is dual. Observe that since no vertex in t is labelled with $\blacktriangle$, $\varphi(u) \neq \bigsqcap\{u_1, \ldots, u_n\}$ for all u. We distinguish two cases:

(1) Assume there is a ν-dominated lasso $t_0, t_1, \ldots, t_n$, starting in t. BESsyness of t implies that there is a ranked vertex t_i on the cycle of the lasso. Without loss of generality assume that t_i has the highest rank on the cycle of the ν-dominated lasso. By definition, this highest rank is even. This means that it induces an equation $\nu X_{t_i} = g_i$ in $\beta(t)$, that precedes all other equations $\sigma X_{t_k} = g_k$ induced by the other vertices on the cycle. Consider the path snippet starting in t_i, leading to t_i again: $t_i, t_{i+1}, \ldots, t_{n-1}, t_j, t_{j+1}, t_{i-1}$. *Gauß elimination* [Mader 1997] allows one to substitute g_{i+1} for $X_{t_{i+1}}$ in the equation for X_{t_i}, yielding $\nu X_{t_i} = g_i[X_{t_{i+1}} := g_{i+1}]$. Repeatedly applying Gauß elimination on the path snippet ultimately allows one to rewrite $\nu X_{t_i} = g_i$ to $\nu X_{t_i} = g_i' \vee X_{t_i}$, since $X_{t_{i-1}}$

depends on X_{t_i} again, and none of the formulae is conjunctive. The solution to $\nu X_{t_i} = g_i' \vee X_{t_i}$ is easily seen to be $X_{t_i} = \mathsf{true}$. This solution ultimately propagates through the entire lasso, and back to t, leading to $\varphi(t) = X_t = \mathsf{true}$.

(2) Suppose there is a finite path $t_0, t_1, \ldots, t_n$ starting in t, where t_n is labelled with $\top$. This means that there is an equation $\sigma X_{t_n} = \mathsf{true}$ on which X_t depends. As the equation $\sigma X_{t_n} = \mathsf{true}$ is solved, we may immediately substitute the solution in all other formulae on the path. As none of the formulae is conjunctive, we find $\varphi(t) = \mathsf{true}$.

Conversely, observe that due to Proposition 5.4, there is a structure graph $t_\blacktriangledown$, void of any vertices labelled $\blacktriangledown$, that has an equation system associated to it with solution equivalent to that of the equation system associated to t. This means that $t_\blacktriangledown$ has no branching structure, but is necessarily a set of lassoes and maximal, finite paths. In case the initial vertex of t is on a lasso, $[\![\varphi(t)]\!][\![\beta(t)]\!] = \mathsf{true}$ is because the cycle on the lasso has an even highest rank. In the other case, $[\![\varphi(t)]\!][\![\beta(t)]\!] = \mathsf{true}$ can only be the case because ultimately $t_\blacktriangledown$ leads to a vertex labelled true. □

Using the structure graph characterisation of solution, we prove that, for BESsy structure graphs that do not have vertices labelled with $\nearrow$, bisimulation minimisation of the structure graph preserves the solution of the associated BES.

LEMMA 5.7. *Let t, t' be normalised BESsy structure graphs in which no vertex is labelled with $\nearrow$. Assume t is minimal w.r.t strong bisimilarity. Then $t \leftrightarrows t'$ implies $[\![\varphi(t)]\!][\![\beta(t)]\!] = [\![\varphi(t')]\!][\![\beta(t')]\!]$.*

PROOF. The case where the initial vertex of t is decorated with a $\top$ or $\bot$ is trivial and therefore omitted. Assume that the initial vertex of t is not decorated with $\top$ nor $\bot$. Suppose that $[\![\varphi(t)]\!][\![\beta(t)]\!] = \mathsf{true}$. By Proposition 5.4 we know that there is a $\blacktriangledown$-choice function γ such that $[\![\beta(t_\gamma)]\!] = [\![\beta(t)]\!]$. We next construct a $\blacktriangledown$-choice function γ' for t' that satisfies the following condition:

$$\forall u \in \mathsf{dom}(\gamma), u' \in \mathsf{dom}(\gamma') : \; u \leftrightarrows u' \implies \gamma(u) \leftrightarrows \gamma'(u')$$

Note that the minimality of t is such that γ satisfies $\gamma(w) \leftrightarrows \gamma(w')$ for all $w \leftrightarrows w'$ with $w, w' \in \mathsf{dom}(\gamma)$. We then have $t_\gamma \leftrightarrows t_{\gamma'}$, as the choice for successors chosen in previously bisimilar $\blacktriangledown$-labelled vertices is synchronised by the $\blacktriangledown$-choice function. Because of this bisimilarity and the finiteness of t', any ν-dominated lasso starting in a vertex u reachable in t implies the existence of a similar ν-dominated lasso starting in vertices u' reachable in t' that are bisimilar to u, and, of course, also *vice versa*. Likewise for maximal finite paths. Suppose the initial vertex of t_γ has only ν-dominated lassoes and finite maximal paths ending in $\top$-labelled vertices. Then, by construction, so has $t'_{\gamma'}$. This means that

$$[\![\varphi(t)]\!][\![\beta(t)]\!] = [\![\varphi(t_\gamma)]\!][\![\beta(t_\gamma)]\!] =^\dagger \mathsf{true} = [\![\varphi(t'_{\gamma'})]\!][\![\beta(t'_{\gamma'})]\!]$$

where at †, Lemma 5.6 is used. Using Proposition 5.4, we find:

$$[\![\varphi(t'_{\gamma'})]\!][\![\beta(t'_{\gamma'})]\!] \Longrightarrow [\![\varphi(t')]\!][\![\beta(t')]\!]$$

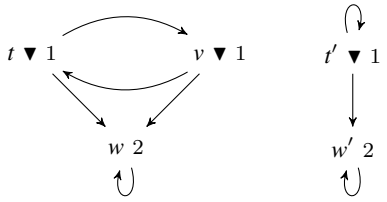
Combining the above, we can conclude that we have:

$$[\![\varphi(t')]\!][\![\beta(t')]\!] = \mathsf{true}$$

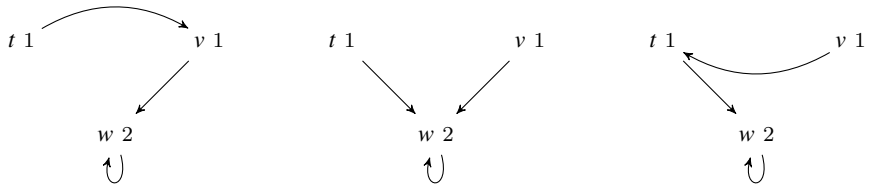
The case where $[\![\varphi(t)]\!][\![\beta(t)]\!] = \mathsf{false}$ follows the same line of reasoning, constructing a structure graph with a ▲-choice function γ, resulting in a structure graph containing no vertices labelled ▲. □

We set out to prove that bisimilar structure graphs t and t' always give rise to equation systems and formulae with the same truth value. The above lemma may seem like a roundabout way in proving this property. In particular, the assumption in Lemma 5.7 that t is minimal with respect to bisimilarity may seem odd. The reason for using the quotient is due to our appeal to the non-constructive Proposition 5.4, as we illustrate through the following example.

Example 5.8. Consider the two bisimilar BESsy structure graphs t and t' below:



Following Lemma 5.6, we know that all vertices will be associated to proposition variables with solution true, as both structure graphs are normalised and contain no ▲-labelled vertices. Appealing to Proposition 5.4, we know that there is a structure graph $t_▼$ that gives rise to an equation system with the same solution as the one that can be associated to t. In fact, there are three choices for $t_▼$:



Note that all three structure graphs are associated to equation systems with the same solution as the equation system for t. However, while the middle structure graph would allow us to construct a ▼-choice function that resolves the choice for successors for vertex t', the other two structure graphs do not allow us to do so, simply because they have bisimilar vertices whose only successor leads to different equivalence classes. Such conflicts do not arise when assuming that t is already minimal, in which case each vertex represents a unique class.

Regardless of the above example, we can still derive the desired result. Based on the previous lemma, the fact that bisimilarity is an equivalence relation on structure graphs and the fact that quotienting is well-behaved, we find the following theorem, which holds for arbitrary BESsy structure graphs.

THEOREM 5.9. *Let t, t' be arbitrary bisimilar BESsy structure graphs. Then for all environments η, $[\![\varphi(t)]\!][\![\beta(t)]\!]\eta = [\![\varphi(t')]\!][\![\beta(t')]\!]\eta$.*

PROOF. Let η be an arbitrary environment. Let $\bar{t}$ and $\bar{t}'$ be the structure graphs obtained from t and t' by replacing all decorations of the form $\nearrow_X$ of all vertices

with $\top$ if $\eta(X) = \mathsf{true}$, and $\bot$ otherwise. Note that we have $\bar{t} \leftrightarrows \bar{t}'$. Based on Lemma 2.7 and Definition 3.10, we find:

$$[\![\varphi(t)]\!][\![\beta(t)]\!]\eta = [\![\varphi(\bar{t})]\!][\![\beta(\bar{t})]\!]$$

Likewise, we can derive such an equivalence for $\bar{t}'$ and t'. By Lemma 4.3, we find:

$$[\![\varphi(\bar{t})]\!][\![\beta(\bar{t})]\!] = [\![\varphi(\mathsf{norm}(\bar{t}))]\!][\![\beta(\mathsf{norm}(\bar{t}))]\!]$$

Again, a similar equivalence can be derived for $\bar{t}'$ and $\mathsf{norm}(\bar{t}')$. Observe that by Proposition 4.5, we find that $\bar{t} \leftrightarrows \bar{t}'$ implies $\mathsf{norm}(\bar{t}) \leftrightarrows \mathsf{norm}(\bar{t}')$. Observe that $\mathsf{norm}(\bar{t}) \leftrightarrows \mathsf{norm}(\bar{t})_{/\leftrightarrows} \leftrightarrows \mathsf{norm}(\bar{t}')$. Finally, since all three are still BESsy structure graphs, that furthermore do not contain vertices labelled with $\nearrow$, we can apply Lemma 5.7 twice to find:

$$\begin{array}{rl} & [\![\varphi(\mathsf{norm}(\bar{t}))]\!][\![\beta(\mathsf{norm}(\bar{t}))]\!] \\ = & [\![\varphi(\mathsf{norm}(\bar{t})_{/\leftrightarrows})]\!][\![\beta(\mathsf{norm}(\bar{t})_{/\leftrightarrows})]\!] \\ = & [\![\varphi(\mathsf{norm}(\bar{t}'))]\!][\![\beta(\mathsf{norm}(\bar{t}'))]\!] \end{array}$$

But this necessitates our desired conclusion:

$$[\![\varphi(t)]\!][\![\beta(t)]\!] = [\![\varphi(t')]\!][\![\beta(t')]\!]$$

□

6. BISIMILARITY ON PROCESSES VS BISIMILARITY ON STRUCTURE GRAPHS

The μ-calculus and bisimilarity of labelled transition systems are intimately related: two states in a transition system are bisimilar if and only if the states satisfy the same set of μ-calculus formulae. As a result, one can rely on bisimulation minimisation techniques for reducing the complexity of the labelled transition system, prior to analysing whether a given μ-calculus formula holds for that system. Unfortunately, in practice, bisimulation reductions are often disappointing, and have to be combined with abstractions that are safe with respect to the formula in order to be worthwhile.

We show that minimising an equation system that encodes a model checking problem is, size-wise, always at least as effective as first applying a safe abstraction to the labelled transition system, subsequently minimising the latter and only then encoding the model checking problem in an equation system. An additional example illustrates that bisimulation minimisation for equation systems can in fact be more effective.

LEMMA 6.1. *Assume $L = \langle S, Act, \rightarrow \rangle$ is an arbitrary labelled transition system. Let ϕ be an arbitrary formula. Then, for arbitrary equation system $\mathcal{E}$, we have:*

$$\begin{array}{rl} \textit{if} & \forall s, s' \in S : s \leftrightarrows s' \implies \forall \tilde{X} \in \mathsf{bnd}(\phi) \cup \mathsf{occ}(\phi) : \langle X_s, \mathcal{E} \rangle \leftrightarrows \langle X_{s'}, \mathcal{E} \rangle \\ \textit{then} & \forall s, s' \in S : s \leftrightarrows s' \implies \langle \mathsf{RHS}_s(\phi), \mathcal{E} \rangle \leftrightarrows \langle \mathsf{RHS}_{s'}(\phi), \mathcal{E} \rangle \end{array}$$

PROOF. Assume a given equation system $\mathcal{E}$. We proceed by means of an induction on the structure of ϕ.

—*Base cases.* Assume that for all $s, s' \in S$, satisfying $s \leftrightarrows s'$, and all $\tilde{X} \in \mathsf{bnd}(\phi) \cup \mathsf{occ}(\phi)$, we have $\langle X_s, \mathcal{E} \rangle \leftrightarrows \langle X_{s'}, \mathcal{E} \rangle$. Assume that $t, t' \in S$ are arbitrary states satisfying $t \leftrightarrows t'$.

—ad $\phi \equiv b$, where $b \in \{\mathsf{true}, \mathsf{false}\}$. Clearly, $\langle \mathsf{RHS}_t(\phi), \mathcal{E}\rangle = \langle b, \mathcal{E}\rangle = \langle \mathsf{RHS}_{t'}(\phi), \mathcal{E}\rangle$, so bisimilarity is guaranteed by unicity of the term, regardless of the states t and t';

—ad $\phi \equiv \tilde{X}$. Clearly, $\tilde{X} \in \mathsf{occ}(\phi)$, so, the required conclusion follows immediately from the fact that $\langle \mathsf{RHS}_t(\phi), \mathcal{E}\rangle = \langle X_t, \mathcal{E}\rangle \leftrightarrows \langle X_{t'}, \mathcal{E}\rangle = \langle \mathsf{RHS}_{t'}(\phi), \mathcal{E}\rangle$;

—Inductive cases: we assume the following induction hypothesis:

$$\begin{array}{rl} \text{if} & \forall s, s' \in S : s \leftrightarrows s' \implies \forall \tilde{X} \in \mathsf{bnd}(f_i) \cup \mathsf{occ}(f_i) : \langle X_s, \mathcal{E}\rangle \leftrightarrows \langle X_{s'}, \mathcal{E}\rangle \\ \text{then} & \forall s, s' \in S : s \leftrightarrows s' \implies \langle \mathsf{RHS}_s(f_i), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{s'}(f_i), \mathcal{E}\rangle \end{array} \tag{IH}$$

From hereon, assume that we have a pair of bisimilar states $t, t' \in S$.

—ad $\phi \equiv f_1 \wedge f_2$. Assume that for any pair of bisimilar states $s, s' \in S$, and for all $\tilde{X} \in \mathsf{bnd}(f_1 \wedge f_2) \cup \mathsf{occ}(f_1 \wedge f_2) = (\mathsf{bnd}(f_1) \cup \mathsf{occ}(f_1)) \cup (\mathsf{bnd}(f_2) \cup \mathsf{occ}(f_2))$, we have $\langle X_s, \mathcal{E}\rangle \leftrightarrows \langle X_{s'}, \mathcal{E}\rangle$. By our induction hypothesis, we have $\langle \mathsf{RHS}_t(f_1), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{t'}(f_1), \mathcal{E}\rangle$ and $\langle \mathsf{RHS}_t(f_2), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{t'}(f_2), \mathcal{E}\rangle$. Lemma 3.5 immediately leads to $\langle \mathsf{RHS}_t(f_1) \wedge \mathsf{RHS}_t(f_2), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{t'}(f_1) \wedge \mathsf{RHS}_{t'}(f_2), \mathcal{E}\rangle$. By definition of RHS, we have the required $\langle \mathsf{RHS}_t(f_1 \wedge f_2), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{t'}(f_1 \wedge f_2), \mathcal{E}\rangle$.

—ad $\phi \equiv f_1 \vee f_2$. Follows the same line of reasoning as the previous case.

—ad $\phi \equiv [A]f_1$. Assume that for all pairs of bisimilar states $s, s' \in S$, and all $\tilde{X} \in \mathsf{bnd}([A]f_1) \cup \mathsf{occ}([A]f_1) = \mathsf{bnd}(f_1) \cup \mathsf{occ}(f_1)$, we have $\langle X_s, \mathcal{E}\rangle \leftrightarrows \langle X_{s'}, \mathcal{E}\rangle$. By induction, we find that $\langle \mathsf{RHS}_s(f_1), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{s'}(f_1), \mathcal{E}\rangle$ holds for all pairs of bisimilar states $s, s' \in S$. This includes states t and t'. Since t and t' are bisimilar, we have $t \xrightarrow{a}$ if and only if $t' \xrightarrow{a}$ for all $a \in A$. We distinguish two cases:

(1) Case $t \not\xrightarrow{a}$ for any $a \in A$. Then also $t' \not\xrightarrow{a}$ for any $a \in A$. Hence, $\mathsf{RHS}_t([A]f_1) = \mathsf{true} = \mathsf{RHS}_{t'}([A]f_1)$. We thus immediately have the required $\langle \mathsf{RHS}_t([A]f_1), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{t'}([A]f_1), \mathcal{E}\rangle$;

(2) Case $t \xrightarrow{a}$ for some $a \in A$. Assume that $t \xrightarrow{a} u$. Since $t \leftrightarrows t'$, we have $t' \xrightarrow{a} u'$ for some $u' \in S$ satisfying $u \leftrightarrows u'$ (and *vice versa*). Because of our induction hypothesis, we then also have $\langle \mathsf{RHS}_u(f_1), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{u'}(f_1), \mathcal{E}\rangle$ (and *vice versa*). We thus find that for every term in the non-empty set $\{\langle \mathsf{RHS}_u(f_1), \mathcal{E}\rangle \mid a \in A, t \xrightarrow{a} u\}$, we can find a bisimilar term in the set $\{\langle \mathsf{RHS}_{u'}(f_1), \mathcal{E}\rangle \mid a \in A, t' \xrightarrow{a} u'\}$ and *vice versa*. Then, by Corollary 3.7, also $\langle \bigsqcap\{\mathsf{RHS}_u(f_1) \mid a \in A, t \xrightarrow{a} u\}, \mathcal{E}\rangle \leftrightarrows \langle \bigsqcap\{\mathsf{RHS}_{u'}(f_1) \mid a \in A, t' \xrightarrow{a} u'\}, \mathcal{E}\rangle$. This leads to $\langle \mathsf{RHS}_t([A]f_1), \mathcal{E}\rangle \leftrightarrows \langle \mathsf{RHS}_{t'}([A]f_1), \mathcal{E}\rangle$.

Clearly, both cases lead to the required conclusion.

—ad $\phi \equiv \langle A\rangle f_1$. Follows the same line of reasoning as the previous case.

—ad $\phi \equiv \sigma\tilde{X}.\ f_1$. Since $\tilde{X} \in \mathsf{bnd}(\phi)$, this case follows immediately from the assumption on $\tilde{X}$ and the definition of RHS.

□

The above lemma is at the basis of the following proposition:

PROPOSITION 6.2. *Let $L = \langle S, Act, \rightarrow\rangle$ be a labelled transition system. Let ϕ be an arbitrary closed μ-calculus formula. Let $s, s' \in S$ be an arbitrary pair of bisimilar states. We then have:*

$$\forall \tilde{X} \in \mathsf{bnd}(\phi) : \langle X_s, \mathsf{E}^L(\phi)\rangle \leftrightarrows \langle X_{s'}, \mathsf{E}^L(\phi)\rangle$$

PROOF. Let ϕ be an arbitrary closed formula, *i.e.*, $\mathsf{occ}(\phi) \subseteq \mathsf{bnd}(\phi)$; since ϕ is a closed formula, $\mathsf{E}^L(\phi)$ will be a closed equation system. In case $\mathsf{bnd}(\phi) = \emptyset$, the statement holds vacuously. Assume $\mathsf{bnd}(\phi) = \{\tilde{X}^1, \ldots, \tilde{X}^n\}$, for some $n \geqslant 1$. Clearly, for each variable $\tilde{X}^i \in \mathsf{bnd}(\phi)$, we obtain equations of the form $\sigma_i X^i_s = \mathsf{RHS}_s(f^i)$ in $\mathsf{E}^L(\phi)$. Let I be the relation on vertices, defined as follows:

$$I = \{(\langle X^i_s, \mathsf{E}^L(\phi)\rangle, \langle X^i_{s'}, \mathsf{E}^L(\phi)\rangle) \mid s, s' \in S, \tilde{X}^i \in \mathsf{bnd}(\phi), s \leftrightarrows s'\}$$

According to Lemma 6.1, I underlies the bisimilarity between $\langle \mathsf{RHS}_s(f^i), \mathsf{E}^L(\phi)\rangle$ and $\langle \mathsf{RHS}_{s'}(f^i), \mathsf{E}^L(\phi)\rangle$ for pairs of bisimilar states $s, s' \in S$. Assume R_{f^i} is the bisimulation relation underlying said equivalence. Let R be defined as follows:

$$R = I \cup \bigcup_{f^i} R_{f^i}$$

R is again a bisimulation relation, as can be checked using the SOS rules for equations and Lemma 6.1. Clearly, R relates $\langle X_s, \mathsf{E}^L(\phi)\rangle$ and $\langle X_{s'}, \mathsf{E}^L(\phi)\rangle$ for arbitrary $\tilde{X} \in \mathsf{bnd}(\phi)$ and bisimilar states $s, s' \in S$. □

As a result of the above proposition one can argue that bisimulation on processes is less powerful than bisimulation on equation systems. However, one may be inclined to believe that combined with abstraction, bisimilarity on processes can lead to greater reductions. Below, we show that even in the presence of *safe* abstractions, bisimilarity on equation systems still surpasses bisimilarity on processes.

We first formalise the notion of safe abstraction for processes. Assume τ is a constant, not present in any set of actions *Act*.

Definition 6.3. An *abstraction* of a labelled transition system $L = \langle S, Act, \rightarrow\rangle$ with respect to a set of actions $A \subseteq Act$, is the labelled transition system $L_A = \langle S, Act \cup \{\tau\}, \rightarrow_A\rangle$, where:

—for all actions $a \notin A$, $s \xrightarrow{a}_A s'$ if and only if $s \xrightarrow{a} s'$;
—$s \xrightarrow{\tau}_A s'$ if and only if $s \xrightarrow{a} s'$ for some $a \in A$;

In effect, an abstraction relabels an action that decorates a transition to τ only if that action appears in the set A. Clearly, if $s \leftrightarrows s'$ holds in L, then also $s \leftrightarrows s'$ in L_A, but the converse does not hold necessarily.

Definition 6.4. An abstraction L_A of L is said to be *safe* with respect to a closed modal μ-calculus formula ϕ if and only if for each subformula $[A']\psi$ and $\langle A'\rangle\psi$ of ϕ, $A' \cap A = \emptyset$.

It follows from the semantics of the modal μ-calculus that all actions of some L, disjoint with the actions found inside the modalities in ϕ can be renamed to τ without affecting the validity of the model checking problem.

PROPOSITION 6.5. *Let $L = \langle S, Act, \rightarrow\rangle$ be a labelled transition system. Let ϕ be a closed modal μ-calculus formula, and assume L_A is a safe abstraction of L. Then for each state $s \in S$, we have $L, s \models \phi$ if and only if $L_A, s \models \phi$.*

The below theorem strengthens the result we obtained in Proposition 6.2, by stating that even in the presence of safe abstractions, bisimilarity for equation systems are as powerful as bisimilarity taking abstractions into account.

THEOREM 6.6. *Let $L = \langle S, Act, \rightarrow \rangle$ be an arbitrary labelled transition system. Let ϕ be an arbitrary closed modal μ-calculus formula over Act. Then for every safe abstraction L_A of L, we have for every pair of bisimilar states $s, s' \in S$ in L_A:*

$$\forall X \in \mathsf{bnd}(\phi) : \langle X_s, \mathsf{E}^L(\phi) \rangle \leftrightarrows \langle X_{s'}, \mathsf{E}^L(\phi) \rangle$$

PROOF. The proof is similar to the proof of Proposition 6.2. In particular, it relies on the definition of a safe abstraction to ensure that $\langle \mathsf{RHS}_s([A']\psi), \mathcal{E} \rangle$ and $\langle \mathsf{RHS}_{s'}([A']\psi), \mathcal{E} \rangle$ for states s, s' that are bisimilar in L_A, but not in L, are mapped onto $\langle \mathsf{true}, \mathcal{E} \rangle$ for both LTSs. □

In fact, Theorem 6.6 positively answers the second question that was raised in the introduction. Bisimilar states in a state space indeed give rise to bisimilar equations in the equations systems encoding model checking problems, even when considering 'safe' abstractions on the original state space.

Lastly, we provide an example that demonstrates that bisimulation reduction of equation systems can lead to arbitrarily larger reductions compared to the reductions achievable through safe abstractions and minimisation of a given LTS.

Example 6.7. Let N be an arbitrary positive number. Consider the process described by the following set of recursive processes (using process algebra style notation):

$$\{P_1 = a \cdot Q_N, \quad P_{n+1} = a \cdot P_n, \quad Q_1 = b \cdot P_N, \quad Q_{n+1} = b \cdot Q_n \mid n < N\}$$

Process P_N induces an LTS L that performs a sequence of a actions of length N, followed by a sequence of b actions of length N, returning to process P_N. Observe that the process P_N cannot be reduced further modulo bisimulation. Let ϕ be the modal μ-calculus formula $\phi = \nu\tilde{X}.\ \langle\{a, b\}\rangle\tilde{X}$, asserting that there is an infinite sequence consisting of a's, b's, or a's and b's. Clearly, there is no safe abstraction of process P_N with respect to ϕ, other than process P_N itself. The equation system $\mathsf{E}^{P_N}(\phi)$ is as follows:

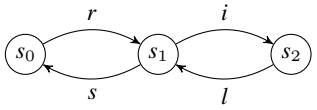
$$\begin{aligned}\nu\{&(X_{P_1} = X_{Q_N} \vee X_{Q_N}), (X_{P_{n+1}} = X_{P_n} \vee X_{P_n}),\\ &(X_{Q_1} = X_{P_N} \vee X_{P_N}), (X_{Q_{n+1}} = X_{Q_n} \vee X_{Q_n}) \mid n < N\}\end{aligned}$$

We find that $\langle X_{P_N}, \mathsf{E}^{P_N}(\phi) \rangle$ and $\langle Y, (\nu Y = Y \vee Y) \rangle$ are bisimilar, which demonstrates a reduction of a factor $2N$. As the labelled transition system can be scaled to arbitrary size, this demonstrates that bisimilarity for equation systems can be arbitrarily more effective.

7. APPLICATION

Equation systems that are not immediately in simple form can be obtained through the reduction of process equivalence checking problems such as the branching bisimulation problem, see *e.g.* [Chen et al. 2007], and the more involved model checking problems. As a slightly simplified example of the latter, we analyse an unreliable channel using μ-calculus model checking. The channel can read messages from its environment through the r action, and send or lose these next through the s action and the l action, respectively. Losing a message happens because of noise affecting the reliability of the channel; we model this using an internal action i preceding action l. In case the message is lost, subsequent attempts are made to send the

message until this finally succeeds. The labelled transition system, modelling this system is given below.



Suppose we wish to verify for which states it holds whether along all paths consisting of reading and sending actions, it is infinitely often possible to potentially never perform a send action. Intuitively, this should be the case in all states: from states s_0 and s_1, there is a finite path leading to state s_1, which can subsequently produce the infinite path $(s_1\ s_2)^\omega$, along which the send action does not occur. For state s_2, we observe that there is no path consisting of reading and sending actions, so the property holds vacuously in s_2. We formalise this problem as follows:[1]

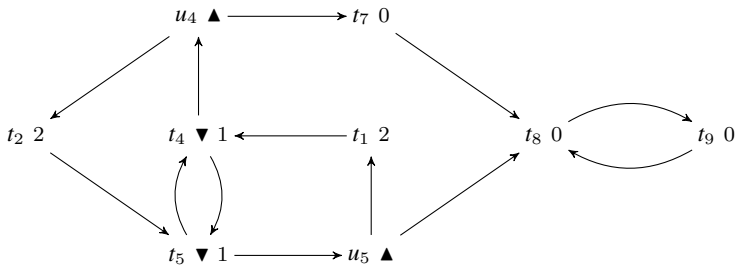
$$\phi \equiv \nu\tilde{X}.\ \mu\tilde{Y}.\ (([\{r,s\}]\tilde{X} \wedge (\nu\tilde{Z}.\ \langle\overline{s}\rangle\tilde{Z})) \vee [\{r,s\}]\tilde{Y})$$

Verifying which states in the labelled transition system satisfy ϕ is answered by solving the below equation system. Note that the equation system was obtained through Definition 2.8. The solution to X_{s_i} answers whether $s_i \models \phi$.

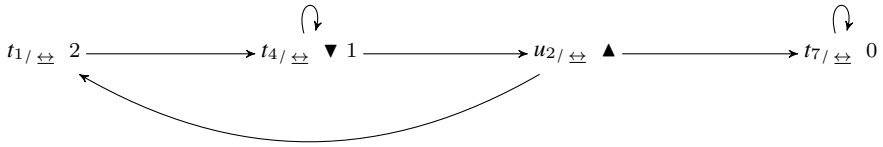
$$\begin{array}{l}
(\nu X_{s_0} = Y_{s_0}) \\
(\nu X_{s_1} = Y_{s_1}) \\
(\nu X_{s_2} = Y_{s_2}) \\
(\mu Y_{s_0} = ((X_{s_1} \wedge X_{s_1}) \wedge Z_{s_0}) \vee ((Y_{s_1} \wedge Y_{s_1}) \vee (Y_{s_1} \wedge Y_{s_1}))) \\
(\mu Y_{s_1} = ((X_{s_0} \wedge X_{s_0}) \wedge Z_{s_1}) \vee ((Y_{s_0} \wedge Y_{s_0}) \vee (Y_{s_0} \wedge Y_{s_0}))) \\
(\mu Y_{s_2} = (\mathsf{true} \wedge Z_{s_2}) \vee \mathsf{true}) \\
(\nu Z_{s_0} = Z_{s_1} \vee Z_{s_1}) \\
(\nu Z_{s_1} = Z_{s_2} \vee Z_{s_2}) \\
(\nu Z_{s_2} = Z_{s_1} \vee Z_{s_1})
\end{array}$$

An answer to the global model checking problem would be encoded by the structure graph $\langle X_{s_0} \wedge X_{s_1} \wedge X_{s_1}, \mathsf{E}^L(\phi)\rangle$. We here only depict the structure graph encoding the local model checking problem $s_0 \models \phi$, encoded by the structure graph $\langle X_{s_0}, \mathsf{E}^L(\phi)\rangle$, which has initial vertex t_1. Note that the ranked vertices t_i originate from the i-th equation in the equation system. Likewise, the unranked vertices u_i originate from the right-hand side of the i-th equation.

[1] Alternative phrasings are possible, but this one nicely projects onto an equation system with non-trivial right-hand sides, clearly illustrating the theory outlined in the previous sections in an example of manageable proportions.



Observe that we have $t_1 \leftrightarrow t_2$, $t_7 \leftrightarrow t_8 \leftrightarrow t_9$, $t_4 \leftrightarrow t_5$ and $u_4 \leftrightarrow u_5$. Minimising the above structure graph with respect to bisimulation leads to the structure graph depicted below:



Note that the structure graph is BESsy, and, hence, admits a translation back to an equation system. Using the translation provided in Definition 3.10 results in the following equation system:

$$\begin{array}{l}(\nu X_{t_{1/\leftrightarrow}} = X_{t_{4/\leftrightarrow}})\\ (\mu X_{t_{4/\leftrightarrow}} = (X_{t_{7/\leftrightarrow}} \wedge (X_{t_{1/\leftrightarrow}} \wedge X_{t_{1/\leftrightarrow}})) \vee (X_{t_{4/\leftrightarrow}} \vee X_{t_{4/\leftrightarrow}}))\\ (\nu X_{t_{7/\leftrightarrow}} = X_{t_{7/\leftrightarrow}})\end{array}$$

Answering the verification problem $s_0 \models \phi$ can thus be achieved by solving 3 equations rather than the original 9 equations. Using standard algorithms for solving equation systems, one quickly finds that all equations of the minimised equation system (and thereby all of the equations from the original equation system they represent) have true as their solutions. Note that the respective sizes of the structure graphs underlying the required equations in the original equation systems are 9 before minimisation and 4 after minimisation, which is almost a 55% gain. Such gains (and larger) appear to be typical in this setting (see also [Keiren and Willemse 2009]), and often surpass those in the setting of labelled transition systems. Similar gains are found for the global model checking problem. Observe, moreover, that the original labelled transition system already is minimal, demonstrating once more that the minimisation of an equation system can be more effective than minimising the original labelled transition system.

8. CLOSING REMARKS

Summary. We presented a set of deduction rules for deriving *structure graphs* from proposition formulae and Boolean equation systems, following the regime of [Plotkin 2004]. In defining these rules, we focused on simplicity. We carefully selected a small set of computationally cheap logical equivalences that we wished to be reflected by bisimilarity in our structure graphs, and subsequently showed that we met these goals.

Structure graphs generalise the *dependency graphs* of *e.g.* [Mader 1997; Keinänen 2006]. The latter formalism is incapable of capturing all the syntactic riches of Boolean equation systems, and is only suited for a subset of closed equation systems in simple form. A question, put forward in [Keiren and Willemse 2009], is how these restrictions affect the power of reduction of strong bisimulation. In Section 4, we showed that these restrictions are in fact beneficial to the identifying power of bisimilarity. This result follows immediately from the meta-theory for structured operational rules, see *e.g.* [Mousavi et al. 2005]. We furthermore proved that also in our richer setting, bisimulation minimisation of a structure graph, induced by an equation system, preserves and reflects the solution to the original equation system. This generalises [Keiren and Willemse 2009, Theorem 1] for dependency graphs.

Beyond the aforementioned results, we studied the connection between bisimilarity for labelled transition systems, the μ-calculus model checking problem and bisimilarity for structure graphs. In Section 6, we showed that bisimulation minimisation of a structure graph (associated to an equation system encoding an arbitrary model checking problem on an arbitrary labelled transition system) is at least as effective as bisimulation minimisation of the labelled transition system prior to the encoding. This relation even holds when bisimilarity is combined with safe abstractions for labelled transition systems. We moreover show that this relation is strict through an example formula ϕ and a labelled transition system L of $2N$ ($N \geqslant 1$) states that is already minimal (even when considering safe abstractions with respect to ϕ), whereas the structure graph induced by the equation system encoding the model checking problem can be reduced by a factor $2N$. These results provide the theoretical underpinning for the huge reductions observed in [Keiren and Willemse 2009]. Reducing a labelled transition system (if available explicitly), prior to encoding the verification problem as a Boolean equation system, can still be useful, as the encoding is proportional in the size of the labelled transition system.

Outlook. The structure graphs that we considered in this paper are of both theoretical and practical significance. They generalise various graph-based models, including the aforementioned dependency graphs, but also Parity Games [Zielonka 1998], and there are strong links between our structure graphs and Switching Graphs [Groote and Ploeger 2009]. Given these links, a *game-based* characterisation of the concept of solution for equation systems, stated in terms of our choice functions and structure graphs is open for investigation. In general, we consider studying equivalences weaker than bisimilarity for structure graphs to be worthwhile. For instance, it is not immediately clear whether the *idempotence-identifying bisimilarity* of [Keiren and Willemse 2009], which weakens some of the requirements of strong bisimilarity while preserving and reflecting the solution of the equation system, carries over to structure graphs without significant modifications. Furthermore, it would be very interesting to study variations of stuttering equivalence in this context, as it is one of the few equivalence relations that allow for good compression at favourable computational complexities.

REFERENCES

Andersen, H. R. 1994. Model checking and boolean graphs. *Theoretical Computer Science 126,* 1, 3–30.

Arnold, A. and Crubille, P. 1988. A linear algorithm to solve fixed-point equations on transition systems. *Information Processing Letters 29,* 2, 57–66.

Bradfield, J. C. and Stirling, C. 2001. Modal logics and mu-calculi: An introduction. In *Handbook of Process Algebra*, J. Bergstra, A. Ponse, and S. Smolka, Eds. Elsevier (North-Holland), Chapter 4, 293–330.

Chen, T., Ploeger, B., van de Pol, J., and Willemse, T. A. C. 2007. Equivalence checking for infinite systems using parameterized boolean equation systems. In *Proceedings of CONCUR 2007, Lisbon, Portugal*, L. Caires and V. T. Vasconcelos, Eds. Lecture Notes in Computer Science, vol. 4703. Springer, 120–135.

Fritz, C. and Wilke, T. 2006. Simulation relations for alternating parity automata and parity games. In *Proceedings of DLT'06, Santa Barbara, CA, USA*, O. H. Ibarra and Z. Dang, Eds. Lecture Notes in Computer Science, vol. 4036. Springer, 59–70.

Garavel, H., Mateescu, R., Lang, F., and Serwe, W. 2007. CADP 2006: A toolbox for the construction and analysis of distributed processes. In *Proceedings of CAV'07, Berlin, Germany*, W. Damm and H. Hermanns, Eds. Lecture Notes in Computer Science, vol. 4590. Springer, 158–163.

Groote, J. F. 1993. Transition system specifications with negative premises. *Theoretical Computer Science 118,* 2, 263–299.

Groote, J. F., Mathijssen, A. H. J., Reniers, M. A., Usenko, Y. S., and van Weerdenburg, M. J. 2009. Analysis of distributed systems with mCRL2. In *Process Algebra for Parallel and Distributed Processing*, M. Alexander and W. Gardner, Eds. Chapman & Hall, Chapter 4, 99–128.

Groote, J. F. and Ploeger, B. 2009. Switching graphs. *International Journal of Foundations of Computer Science 20,* 5, 869–886.

Groote, J. F. and Willemse, T. A. C. 2005. Model-checking processes with data. *Science of Computer Programming 56,* 3, 251–273.

Jurdziński, M. 1998. Deciding the winner in parity games is in UP ∩ co-UP. *Information Processing Letters 68,* 3, 119–124.

Keinänen, M. K. 2006. Techniques for solving boolean equation systems. Ph.D. thesis, Helsinki University of Technology.

Keiren, J. J. A. and Willemse, T. A. C. 2009. Bisimulation minimisations for Boolean equation systems. In *Proceedings HVC'09, Haifa, Israel*. Lecture Notes in Computer Science. To appear.

Kozen, D. 1983. Results on the propositional μ-calculus. *Theoretical Computer Science 27*, 333–354.

Larsen, K. G. 1993. Efficient local correctness checking. In *Proceedings of CAV '92, Montreal, Canada*, G. von Bochmann and D. K. Probst, Eds. Lecture Notes in Computer Science, vol. 663. Springer, 30–43.

Liu, X., Ramakrishnan, C., and Smolka, S. 1998. Fully local and efficient evaluation of alternating fixed points. In *Proceedings of TACAS'98, Lisbon, Portugal*, B. Steffen, Ed. Lecture Notes in Computer Science, vol. 1384. Springer, 5–19.

Liu, X. and Smolka, S. 1998. Simple linear-time algorithms for minimal fixed points. In *Proceedings of ICALP'98, Aalborg, Denmark*, K. G. Larsen, S. Skyum, and G. Winskel, Eds. Lecture Notes in Computer Science, vol. 1443. Springer, 53–66.

Mader, A. 1997. Verification of modal properties using boolean equation systems. Ph.D. thesis, Technische Universität München.

Mateescu, R. 2003. A generic on-the-fly solver for alternation-free Boolean equation systems. In *Proceedings of TACAS'03, Warsaw, Poland*, H. Garavel and J. Hatcliff, Eds. Lecture Notes in Computer Science, vol. 2619. Springer, 81–96.

Mateescu, R. 2006. CAESAR_SOLVE: A generic library for on-the-fly resolution of alternation-free boolean equation systems. *International Journal on Software Tools for Technology Transfer 8,* 1, 37–56.

Mousavi, M., Reniers, M. A., and Groote, J. F. 2005. Notions of bisimulation and congruence formats for SOS with data. *Information and Computation 200,* 1, 107–147.

Park, D. M. 1981. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI Conference*. Lecture Notes in Computer Science, vol. 104. Springer-Verlag, Berlin, Germany, 167–183.

Plotkin, G. D. 2004. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming 60–61*, 17–139.

Reniers, M. A. and Willemse, T. A. C. 2010. Analysis of boolean equation systems through structure graphs. In *Proceedings of SOS'09, Bologna, Italy*, B. Klin and P. Sobociński, Eds. Electronic Notes in Theoretical Computer Science, vol. 18. 92–107.

Schewe, S. 2007. Solving parity games in big steps. In *Proceedings of FSTTCS'07, New Delhi, India*, V. Arvind and S. Prasad, Eds. Lecture Notes in Computer Science, vol. 4855. Springer, 449–460.

Stevens, P. and Stirling, C. 1998. Practical model checking using games. In *Proceedings of TACAS'98, Lisbon, Portugal*, B. Steffen, Ed. Lecture Notes in Computer Science, vol. 1384. Springer, 85–101.

Stirling, C. 1997. Bisimulation, model checking and other games. Notes for Mathfit Instructional Meeting on Games and Computation. University of Edinburgh.

Tarski, A. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics 5,* 2, 285–309.

Willemse, T. A. C. 2010. Consistent correlations for parameterised boolean equation systems with applications in correctness proofs for manipulations. In *Proceedings of CONCUR 2010, Paris, France*, P. Gastin and F. Laroussinie, Eds. Lecture Notes in Computer Science, vol. 6269. Springer, 584–598.

Zielonka, W. 1998. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science 200,* 1-2, 135 – 183.

A. DETAILED PROOFS AND ADDITIONAL LEMMATA

LEMMA A.1. *Let f, g be formulae, $\mathcal{E}$ a BES, and η an arbitrary environment, then we have the following semantic equivalences:*

$$\begin{aligned}[\![\varphi(\langle f, \mathcal{E}\rangle) \wedge \varphi(\langle g, \mathcal{E}\rangle)]\!]\eta &= [\![\varphi(\langle f \wedge g, \mathcal{E}\rangle)]\!]\eta \\ [\![\varphi(\langle f, \mathcal{E}\rangle) \vee \varphi(\langle g, \mathcal{E}\rangle)]\!]\eta &= [\![\varphi(\langle f \vee g, \mathcal{E}\rangle)]\!]\eta\end{aligned}$$

PROOF. We prove the first statement. Proof of the second statement is completely symmetric.

We first prove the implication $[\![\varphi(\langle f \wedge g, \mathcal{E}\rangle)]\!]\eta \Rightarrow [\![\varphi(\langle f, \mathcal{E}\rangle) \wedge \varphi(\langle g, \mathcal{E}\rangle)]\!]\eta$. We use induction on the structure of $\varphi(\langle f \wedge g, \mathcal{E}\rangle)$:

—case $\varphi(\langle f \wedge g, \mathcal{E}\rangle) = \bigsqcap\{\varphi(u') \mid \langle f \wedge g, \mathcal{E}\rangle \to u'\}$. It follows that $d(\langle f \wedge g, \mathcal{E}\rangle) = \blacktriangle$ and $\langle f \wedge g, \mathcal{E}\rangle \notin \mathsf{dom}(r)$. As $d(\langle f \wedge g, \mathcal{E}\rangle) = \blacktriangle$ and $\langle f \wedge g, \mathcal{E}\rangle$ is BESsy, there must be at least one u' such that $\langle f \wedge g, \mathcal{E}\rangle \to u'$.
We need to show that for each conjunct $u' \in \{\varphi(u'') \mid \langle f \wedge g, \mathcal{E}\rangle \to u''\}$ either:
—$u' \in \{\varphi(u'') \mid \langle f, \mathcal{E}\rangle \to u''\}$, or
—$u' \in \{\varphi(u'') \mid \langle g, \mathcal{E}\rangle \to u''\}$, or
—$u' = \varphi(\langle f, \mathcal{E}\rangle)$, or
—$u' = \varphi(\langle g, \mathcal{E}\rangle)$.
Let $v = \varphi(u')$ be an arbitrary conjunct in $\{\varphi(u'') \mid \langle f \wedge g, \mathcal{E}\rangle \to u''\}$. So we know $\langle f \wedge g, \mathcal{E}\rangle \to u'$. We apply case distinction on the inference rules that can introduce this edge.
—$\langle f \wedge g, \mathcal{E}\rangle \to u'$ is introduced through rule (7). Then we may assume that $d(\langle f, \mathcal{E}\rangle) = \blacktriangle$, $\langle f, \mathcal{E}\rangle \notin \mathsf{dom}(r)$ and $\langle f, \mathcal{E}\rangle \to u'$. According to the definition of φ we find that $\varphi(\langle f, \mathcal{E}\rangle) = \bigsqcap\{\varphi(u'') \mid \langle f, \mathcal{E}\rangle \to u''\}$. Hence by induction we find that v is a conjunct of $\varphi(\langle f, \mathcal{E}\rangle)$. As $d(\langle f, \mathcal{E}\rangle) = \blacktriangle$, every conjunct of this formula is also a conjunct of $\varphi(\langle f \wedge g, \mathcal{E}\rangle)$.
—$\langle f \wedge g, \mathcal{E}\rangle \to u'$ is introduced through rule (8). This case is analogous to the previous case.
—$\langle f \wedge g, \mathcal{E}\rangle \to u'$ is introduced through rule (11). We may assume that $\neg\langle f, \mathcal{E}\rangle\blacktriangle$. Therefore, $u' = \langle f, \mathcal{E}\rangle$, and the corresponding formula is $\varphi(\langle f, \mathcal{E}\rangle)$.
—The cases where $\langle f \wedge g, \mathcal{E}\rangle \to u'$ is introduced through rules (12), (15) or (16) are analogous to the previous case.

—case $\varphi(\langle f \wedge g, \mathcal{E}\rangle) = \bigsqcup\{\varphi(u') \mid \langle f \wedge g, \mathcal{E}\rangle \to u'\}$. According to rule (5) it must be the case that $\langle f \wedge g, \mathcal{E}\rangle\blacktriangle$. According to BESsyness then $d(\langle f \wedge g, \mathcal{E}\rangle) \neq \blacktriangledown$, hence $\varphi(\langle f \wedge g, \mathcal{E}\rangle) \neq \bigsqcup\{\varphi(u') \mid \langle f \wedge g, \mathcal{E}\rangle \to u'\}$, hence this case cannot apply.

—the cases where $\varphi(\langle f \wedge g, \mathcal{E}\rangle) \in \{\mathsf{true}, \mathsf{false}, X\}$ are analogous to the previous case.

—case $\varphi(\langle f \wedge g, \mathcal{E}\rangle) = X_{\langle f \wedge g, \mathcal{E}\rangle}$. Appealing to rule (5) it must be the case that $\varphi(\langle f \wedge g, \mathcal{E}\rangle)\blacktriangle$. Furthermore we know $\langle f \wedge g, \mathcal{E}\rangle \in \mathsf{dom}(r)$. According to the operational rules all ranked terms are of the form $\langle Y, \mathcal{E}\rangle$, for some Y. This contradicts the assumption that the term we are considering is $\langle f \wedge g, \mathcal{E}\rangle$.

The reverse case, showing that $[\![\varphi(\langle f \wedge g, \mathcal{E}\rangle)]\!]\eta \Leftarrow [\![\varphi(\langle f, \mathcal{E}\rangle) \wedge \varphi(\langle g, \mathcal{E}\rangle)]\!]\eta$ commences by induction on the structure of $\varphi(\langle f, \mathcal{E}\rangle)$ and $\varphi(\langle g, \mathcal{E}\rangle)$. We show that each conjunct of $\varphi(\langle f, \mathcal{E}\rangle)$ is also a conjunct of $\varphi(\langle f \wedge g, \mathcal{E}\rangle)$. The case for $\varphi(\langle g, \mathcal{E}\rangle)$ is completely analogous.

—case $\varphi(\langle f, \mathcal{E}\rangle) = \bigsqcap\{\varphi(u') \mid \langle f, \mathcal{E}\rangle \to u'\}$. In this case we know that $d(\langle f, \mathcal{E}\rangle) = \blacktriangle$, and $\langle f, \mathcal{E}\rangle \notin \mathsf{dom}(r)$. Let $\langle f, \mathcal{E}\rangle \to u'$, so $\varphi(u')$ is a top level conjunct of $\varphi(\langle f, \mathcal{E}\rangle)$. From rule (7) it follows immediately that $\langle f \wedge g, \mathcal{E}\rangle \to u'$, and $d(\langle f \wedge g, \mathcal{E}\rangle) = \blacktriangle$ according to (5), hence $\varphi(\langle f \wedge g, \mathcal{E}\rangle) = \bigsqcap\{\varphi(u') \mid \langle f \wedge g, \mathcal{E}\rangle \to u'\}$, and $\varphi(u')$ is a conjunct of $\varphi(\langle f \wedge g, \mathcal{E}\rangle)$.

—$\varphi(\langle f, \mathcal{E}\rangle) = \bigsqcup\{\varphi(u') \mid \langle f, \mathcal{E}\rangle \to u'\}$. So we know that $d(\langle f, \mathcal{E}\rangle) = \blacktriangledown$ and $\langle f, \mathcal{E}\rangle \notin \mathsf{dom}(r)$. Observe that the only conjunct of $\varphi(\langle f, \mathcal{E}\rangle)$ is $\varphi(\langle f, \mathcal{E}\rangle)$ itself. We show that $\varphi(\langle f, \mathcal{E}\rangle)$ is a conjunct of $\varphi(\langle f \wedge g, \mathcal{E}\rangle)$. According to rule (11), $\langle f \wedge g, \mathcal{E}\rangle \to \langle f, \mathcal{E}\rangle$. Furthermore $d(\langle f \wedge g, \mathcal{E}\rangle) = \blacktriangle$ according to (5) and $\langle f \wedge g, \mathcal{E}\rangle \notin \mathsf{dom}(r)$ according to (2), hence $\varphi(\langle f \wedge g, \mathcal{E}\rangle) = \bigsqcap\{\varphi(u') \mid \langle f \wedge g, \mathcal{E}\rangle \to u'\}$, and $\varphi(\langle f, \mathcal{E}\rangle)$ is a conjunct of $\varphi(\langle f \wedge g, \mathcal{E}\rangle)$.

—cases $\varphi(\langle f, \mathcal{E}\rangle) \in \{\mathsf{true}, \mathsf{false}, X\}$ follow a similar line of reasoning as the previous case.

—$\varphi(\langle f, \mathcal{E}\rangle) = X_{\langle f, \mathcal{E}\rangle}$, where $\langle f, \mathcal{E}\rangle \in \mathsf{dom}(r)$. This again follows a similar line of reasoning. We use the observation that the only edge that is generated from $\langle f \wedge g, \mathcal{E}\rangle$ induced by $\langle f, \mathcal{E}\rangle$ is the edge $\langle f \wedge g, \mathcal{E}\rangle \to \langle f, \mathcal{E}\rangle$ because f is ranked, according to (15), and in case also $d(\langle f, \mathcal{E}\rangle) \notin \{\blacktriangle, \blacktriangledown\}$ the same edge is generated (according to rule (11)).

□

LEMMA A.2. *Let $\mathcal{E}$ be a BES, η an environment, such that $\eta(Y) = \eta(X_{\langle Y, \mathcal{E}\rangle})$ for all $Y \in \mathsf{bnd}(\mathcal{E})$. Let f be a formula, such that $\mathsf{occ}(f) \subseteq \{Y \mid X_{\langle Y, \mathcal{E}\rangle} \in \mathsf{bnd}(\beta(\langle f, \mathcal{E}\rangle)) \cup \mathsf{free}(\beta(\langle f, \mathcal{E}\rangle))\}$. Then it holds that $[\![f]\!]\eta = [\![\varphi(\langle f, \mathcal{E}\rangle)]\!]\eta$*

PROOF. Let $\mathcal{E}$ be this BES, and f a formula. Assume that $\mathsf{occ}(f) \subseteq \{Y \mid X_{\langle Y, \mathcal{E}\rangle} \in \mathsf{bnd}(\beta(\langle f, \mathcal{E}\rangle)) \cup \mathsf{free}(\beta(\langle f, \mathcal{E}\rangle))\}$. We show that $[\![f]\!]\eta = [\![\varphi(\langle f, \mathcal{E}\rangle)]\!]\eta$ by induction on the structure of f.

—$f = \mathsf{true}$. By definition of φ, $[\![\varphi(\langle \mathsf{true}, \mathcal{E}\rangle)]\!]\eta = [\![\mathsf{true}]\!]\eta$.

—$f = \mathsf{false}$. Analogous to the previous case.

—$f = Y$. We distinguish two cases, either Y is bound, or Y is free:

—Y is bound, *i.e.* $X_{\langle Y, \mathcal{E}\rangle} \in \mathsf{bnd}(\beta(\langle f, \mathcal{E}\rangle))$. We derive:

$$
\begin{aligned}
& [\![\varphi(\langle Y, \mathcal{E}\rangle)]\!]\eta \\
= \ & \{X_{\langle Y, \mathcal{E}\rangle} \in \beta(\langle f, \mathcal{E}\rangle), \text{ hence } \langle Y, \mathcal{E}\rangle \in \mathsf{dom}(r), \text{ use definition of } \varphi\} \\
& [\![X_{\langle Y, \mathcal{E}\rangle}]\!]\eta \\
= \ & \{\text{Semantics of BES}\} \\
& \eta(X_{\langle Y, \mathcal{E}\rangle}) \\
= \ & \{\text{Assumption } \eta(X_{\langle Y, \mathcal{E}\rangle}) = \eta(Y)\} \\
& \eta(Y) \\
= \ & \{\text{Semantics of BES}\} \\
& [\![Y]\!]\eta
\end{aligned}
$$

—$Y \in \mathsf{free}(\beta(\langle f, \mathcal{E}\rangle))$. This case is easy, as $Y \in \mathsf{free}(\beta(\langle f, \mathcal{E}\rangle))$, also $\nearrow_{\langle Y, \mathcal{E}\rangle} Y$, hence using the definition of φ we immediately find $[\![\varphi(\langle Y, \mathcal{E}\rangle)]\!]\eta = [\![Y]\!]\eta$.

—$f = g \wedge g'$. Based on the SOS we know that $d(\langle g \wedge g', \mathcal{E}\rangle) = \blacktriangle$. As induction hypothesis we assume that the lemma holds for all subformulae. We derive:

$$\begin{aligned}
& [\![\varphi(\langle g \wedge g', \mathcal{E}\rangle)]\!]\eta \\
= \ & \{\text{Lemma A.1}\} \\
& [\![\varphi(\langle g, \mathcal{E}\rangle) \wedge \varphi(\langle g', \mathcal{E}\rangle)]\!]\eta \\
= \ & \{\text{Semantics of BES}\} \\
& [\![\varphi(\langle g, \mathcal{E}\rangle)]\!]\eta \wedge [\![\varphi(\langle g', \mathcal{E}\rangle)]\!]\eta \ ll \\
= \ & \{\text{Induction hypothesis}\} \\
& [\![g]\!]\eta \wedge [\![g']\!]\eta \\
= \ & \{\text{Semantics of BES}\} \\
& [\![g \wedge g']\!]\eta
\end{aligned}$$

—$f = g \vee g'$. Analogous to the previous case.

□

Lemma A.3. *Let $\mathcal{E}$ be a BES, $(\sigma X = f) \in \mathcal{E}$. Then it holds that $\varphi(\langle f, \mathcal{E}\rangle) = \mathsf{rhs}(\langle X, \mathcal{E}\rangle)$.*

Proof. Assume that $(\sigma X = f) \in \mathcal{E}$. Observe that $\langle X, \mathcal{E}\rangle \in \mathsf{dom}(r)$. We show this lemma using case distinction on rules for rhs.

—$d(\langle X, \mathcal{E}\rangle) = \blacktriangle$. Then according to rule (19) also $d(\langle f, \mathcal{E}\rangle) = \blacktriangle$, and furthermore $\langle f, \mathcal{E}\rangle \notin \mathsf{dom}(r)$. We derive:

$$\begin{aligned}
& \mathsf{rhs}(\langle X, \mathcal{E}\rangle) \\
= \ & \{\text{Definition of } \mathsf{rhs}\} \\
& \textstyle\bigsqcap\{\varphi(u') \mid \langle X, \mathcal{E}\rangle \to u'\} \\
= \ & \{d(\langle f, \mathcal{E}\rangle) = \blacktriangle \text{ and } \langle X, \mathcal{E}\rangle \notin \mathsf{dom}(r), \text{ hence } \langle X, \mathcal{E}\rangle \to u' \text{ if and only if } \langle f, \mathcal{E}\rangle \to u' \text{ according to rule (23)}\} \\
& \textstyle\bigsqcap\{\varphi(u') \mid \langle f, \mathcal{E}\rangle \to u'\} \\
= \ & \{\text{Definition of } \varphi\} \\
& \varphi(\langle f, \mathcal{E}\rangle)
\end{aligned}$$

—$d(\langle X, \mathcal{E}\rangle) = \blacktriangledown$. Analogous to the previous case.

—$d(\langle X, \mathcal{E}\rangle) \neq \blacktriangle$ and $d(\langle X, \mathcal{E}\rangle) \neq \blacktriangledown$. We know that there is exactly one u' such that $\langle X, \mathcal{E}\rangle \to u'$, hence using rule (21) we find $\langle X, \mathcal{E}\rangle \to \langle f, \mathcal{E}\rangle$. By definition of rhs, $\mathsf{rhs}(\langle X, \mathcal{E}\rangle) = \varphi(\langle f, \mathcal{E}\rangle)$.

□

Proposition A.4 (Proposition 3.11 in the main text). *Let $\mathcal{E}$ be a BES such that $\sigma Y = f \in \mathcal{E}$. Then for all environments η for which $\eta(Y) = \eta(X_{\langle Y, \mathcal{E}\rangle})$, $[\![f]\!]\eta = [\![\mathsf{rhs}(\langle Y, \mathcal{E}\rangle)]\!]\eta$.*

Proof. We prove this using a distinction on the cases of $\mathsf{rhs}(\langle Y, \mathcal{E}\rangle)$.

—case $d(\langle Y, \mathcal{E}\rangle) = \blacktriangle$). We derive:

$$\begin{aligned}
& [\![\mathsf{rhs}(\langle Y, \mathcal{E}\rangle)]\!]\eta \\
= \ & \{\text{Lemma A.3, } \sigma Y = f \in \mathcal{E}\} \\
& [\![\varphi(\langle f, \mathcal{E}\rangle)]\!]\eta \\
= \ & \{\text{Lemma A.2}\} \\
& [\![f]\!]\eta
\end{aligned}$$

—The cases where $d(\langle Y, \mathcal{E} \rangle) = \blacktriangledown$ and $d(\langle Y, \mathcal{E} \rangle) \notin \{\blacktriangle, \blacktriangledown\}$ are completely analogous.

□

Received February 2010; accepted July 2010