# Evidence for Fixpoint Logic

**Sjoerd Cranen, Bas Luttik, and Tim A. C. Willemse**

**Technische Universiteit Eindhoven**
**P.O. Box 513, 5600 MB Eindhoven, The Netherlands**
{s.cranen,s.p.luttik,t.a.c.willemse}@tue.nl

**Abstract**

For many modal logics, dedicated model checkers offer diagnostics (*e.g.*, counterexamples) that help the user understand the result provided by the solver. Fixpoint logic offers a unifying framework in which such problems can be expressed and solved, but a drawback of this framework is that it lacks comprehensive diagnostics generation. We extend the framework with a notion of evidence, which can be specialised to obtain diagnostics for various model checking problems, behavioural equivalence and refinement checking problems. We demonstrate this by showing how our notion of evidence can be used to obtain diagnostics for the problem of deciding stuttering bisimilarity. Moreover, we show that our notion generalises the existing notions of counterexample and witness for LTL and ACTL* model checking.

**1998 ACM Subject Classification** F.4.1 Mathematical Logic

**Keywords and phrases** fixpoint logic, diagnostics, counterexample, model checking, stuttering bisimilarity, ACTL*

**Digital Object Identifier** 10.4230/LIPIcs.CSL.2015.78

## 1 Introduction

Many of the techniques and tools developed for the purpose of verification of software and hardware systems – such as model checkers for various temporal logics, and refinement checkers for various behavioural equivalences and preorders – perform checks that can also be encoded in fixpoint extensions of first-order logic. Rather than having specialised tools to compute answers to specific verification problems, one can then use a solver for fixpoint logic to compute answers to such problems. This approach is, *e.g.*, taken by the mCRL2 tool set [7], which solves model checking problems for transition systems (which are essentially described by first-order structures) by translating the model checking problem to the problem of checking validity of a formula in fixpoint logic [15]; in a similar vein, the tool set offers tools that decide whether there is a behavioural equivalence between two transition systems by translating the decision problem to fixpoint logic [3].

While using fixpoint logic as a unifying framework for verification problems is desirable from a theoretical point of view, there are some practical aspects that need to be considered to prepare such a framework for large scale use. One pertinent problem is the issue of diagnostic generation. Many specialised tools for, *e.g.*, LTL model checking, provide diagnostics (for instance in the form of counterexample traces). For fixpoint logic in general, no such notion yet exists to our knowledge, and it is not immediately obvious how diagnostics generation for specific verification problems can be fit into the more generic framework of fixpoint logic.

Our contribution is an approach to diagnostic generation in fixpoint logic. As a starting point for our investigation, we use the notion of a proof graph [9] for a particular fixpoint logic called *parameterised Boolean equation systems* (PBES) [16]. Proof graphs for PBESs are loosely based on the notion of *support set* of Tan and Cleaveland [23], and provide a

24th EACSL Annual Conference on Computer Science Logic (CSL 2015).
Editor: Stephan Kreutzer; pp. 78–93

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

formal representation of an argument for the validity or invalidity of a fixpoint formula in a first-order structure. They capture only relevant information about predicates defined by fixpoints, and their interdependencies; in particular, they abstract from general first-order reasoning. To build a general theory of evidence for fixpoint logics based on proof graphs, we need to adapt the notion of proof graph proposed in [9] in two ways:

- we include nodes to reveal information about first-order predicates in our proof graphs, capturing some extra information needed for evidence extraction, and
- we integrate the concept of negation (of fixpoint formulas) into our proof graphs, which was not yet needed in the context of PBES.

Intuitively, evidence is that part of a first-order structure that captures all relevant information in an argument for the validity or invalidity of a fixpoint formula. Given a proof graph representing such an argument, we propose to define evidence as a weak substructure in which all information referred to in the proof graph is still available. The weakest such substructure we call the evidence projection associated with the proof graph.

We confirm the usefulness of our definition of evidence by suitably instantiating it to get formal notions of counterexample and witness in the context of behavioural equivalence checking, and in the context of model checking. First, we define stuttering bisimilarity on Kripke structures as a general fixpoint formula, and then show how a counterexample can be obtained using the notion of evidence presented earlier. Second, we show how to retrieve counterexamples of a linear form (traces) for LTL model checking, and tree-like ones (as presented by Clarke et al. in [4]) for ∀CTL*/∃CTL* model checking.

The theory of proof graphs has much in common with other formal systems for representing proofs in the context of logics and calculi including notions of fixpoints or inductively defined predicates, such as the various tableaux systems for fixpoint logics [17, 11], proof systems for recursive types (see, *e.g.*, [12] for an overview), and the proof systems for inductively defined predicates discussed in [1]. It also has a close resemblance with parity games [13]. The notion of proof graph, however, serves a different purpose, as a stepping stone towards a formal notion of evidence for fixpoint logics with a focus on the predicates defined as fixpoints. It therefore only captures relevant information about first-order predicates, predicates defined by fixpoints, and their interdependencies; in particular, it abstracts from general first-order reasoning.

To simplify the presentation, we discuss our results in the context of *Least Fixpoint Logic* (LFP) which we introduce in Section 2.[1] In Section 3 we recap and adapt the notion of proof graph, and extend it with a notion of evidence for LFP. In Section 4.1, we first define stuttering bisimilarity on Kripke structures as a general LFP formula, and then show how a counterexample can be obtained using our notion of evidence. We illustrate how our proof graphs can also be used to generate counterexamples and witnesses for model checking problems in Section 4.2, and we conclude in Section 5.

## 2 Least fixpoint logic

We recall the syntax and semantics of LFP, an extension of first-order logic with least and greatest fixpoint operators, and we fix some additional notation and concepts needed in the rest of the paper.

[1] In [6] it is shown that they also hold for a more general fixpoint logic that encompasses both LFP and PBES.

Let $\Sigma = \langle \mathcal{R}, \mathcal{F}, \mathsf{ar} \rangle$ be a *signature* consisting of a set of relation symbols $\mathcal{R}$, a set of function symbols $\mathcal{F}$, and a function $\mathsf{ar} \colon \mathcal{R} \cup \mathcal{F} \to \mathbb{N}$ that assigns an *arity* to every symbol in $\mathcal{R}$ and $\mathcal{F}$. Furthermore, we assume a countably infinite set of *first-order variables*. We inductively define a *term* to be either a first-order variable, or a function symbol of arity $n$ applied to a sequence of $n$ terms; a term is *open* if it contains variables, and *closed* otherwise. We also presuppose a set $\mathcal{X}$ of *second-order variables*, each with an associated arity; a second-order variable $X$ of arity $n$ can be thought of as ranging over $n$-ary relations. Then, the set of *LFP formulas* over $\Sigma$ is defined by the following grammar:

$$\varphi, \psi ::= X\bar{t} \mid R\bar{t} \mid t_1 = t_2 \mid \neg\varphi \mid \varphi \vee \psi \mid \exists_x \varphi \mid [\mathbf{lfp}\, X\bar{x}.\, \varphi]\bar{t}$$

with $X$ ranging over the second-order variables in $\mathcal{X}$, $R$ ranging over the relation symbols in $\mathcal{R}$, $x$ ranging over first-order variables, $t_1$ and $t_2$ ranging over terms, and $\bar{x}$ and $\bar{t}$ ranging over finite sequences of first-order variables and terms of appropriate length. Not all LFP formulas generated by the grammar above are considered *well-formed*; for formulas of the form $[\mathbf{lfp}\, X\bar{x}.\, \varphi]\bar{t}$ we impose the additional syntactic requirement that all occurrences of $X$ in $\varphi$ are *positive*, *i.e.*, in the scope of an even number of negations. Henceforth, we will only consider well-formed LFP formulas. We write $\varphi \sqsubseteq \psi$ if $\varphi$ is a subformula of $\psi$.

A *(first-order) structure* $\mathfrak{A}$ is a tuple $\langle A, \Sigma, \mathcal{I} \rangle$ in which $A$ is a set (the *domain of discourse*), $\Sigma$ is a signature and $\mathcal{I}$ is an *interpretation function*. The interpretation function $\mathcal{I}$ is a mapping that associates with every relation symbol $R$ in $\Sigma$ a relation $R^{\mathfrak{A}}$ on $A$ of appropriate arity, and with every function symbol $f$ in $\Sigma$ a function $f^{\mathfrak{A}}$ on $A$ of appropriate arity.

A structure $\mathfrak{B}$ is a *weak substructure* of structure $\mathfrak{A}$, denoted $\mathfrak{B} \sqsubseteq_{\mathrm{w}} \mathfrak{A}$, if it has a domain of discourse $B \subseteq A$, the same relation and function symbols as $\mathfrak{A}$, and an interpretation function such that for every $n$-ary relation symbol $R$ we have $R^{\mathfrak{B}} \subseteq R^{\mathfrak{A}} \cap B^n$, and for every $n$-ary function symbol $f$ we have that $f^{\mathfrak{B}}(\bar{b}) = f^{\mathfrak{A}}(\bar{b}) \in B$ for all $\bar{b} \in B^n$.

Terms and LFP formulas are evaluated on a first-order structure $\mathfrak{A}$ in a given *environment* $\theta$ that maps first-order variables to elements of $A$ and second-order variables to relations on $A$ of appropriate arity. If $t$ is a term, then $t^{\mathfrak{A},\theta}$ is the element of $A$ denoted by $t$ in environment $\theta$ defined in the usual way (*i.e.*, if $t$ is a variable, then $t^{\mathfrak{A},\theta} = \theta(t)$, and if $t = f(t_0, \ldots, t_n)$, then $t^{\mathfrak{A},\theta} = f^{\mathfrak{A}}(t_0^{\mathfrak{A},\theta}, \ldots, t_n^{\mathfrak{A},\theta})$). The evaluation of LFP formulas is less straightforward due to the fixpoint operators. The idea is that, given a second-order variable $X$ of arity $n$, a sequence of first-order variables $\bar{x}$ of length $n$, and an environment $\theta$, we can associate with every formula $\varphi$ an operator $\mathbf{T}^{\mathfrak{A},\theta}_{X,\bar{x},\varphi}$ on the complete lattice of $n$-ary relations on $A$. If all occurrences of $X$ in $\varphi$ are positive, then the operator $\mathbf{T}^{\mathfrak{A},\theta}_{X,\bar{x},\varphi}$ – which is to be defined precisely below – is monotone on the lattice of $n$-ary relations on $A$. Therefore, it has a unique least fixpoint (see [24]), which we will denote by $\mathbf{lfp}\, \mathbf{T}^{\mathfrak{A},\theta}_{X,\bar{x},\varphi}$. We proceed to define the relation $\mathfrak{A}, \theta \models \varphi$ and the operators $\mathbf{T}^{\mathfrak{A},\theta}_{X,\bar{x},\varphi} \colon A^n \to A^n$ ($n$ the arity of $X$) simultaneously by induction on the structure of $\varphi$:

$$\begin{array}{llll}
\mathfrak{A}, \theta \models R\bar{t} & \text{iff } \bar{t}^{\mathfrak{A},\theta} \in R^{\mathfrak{A}} & \mathfrak{A}, \theta \models X\bar{t} & \text{iff } \bar{t}^{\mathfrak{A},\theta} \in \theta(X) \\
\mathfrak{A}, \theta \models t_1 = t_2 & \text{iff } t_1^{\mathfrak{A},\theta} = t_2^{\mathfrak{A},\theta} & \mathfrak{A}, \theta \models \neg\varphi & \text{iff not } \mathfrak{A}, \theta \models \varphi \\
\mathfrak{A}, \theta \models \varphi \vee \psi & \text{iff } \mathfrak{A}, \theta \models \varphi \text{ or } \mathfrak{A}, \theta \models \psi & \mathfrak{A}, \theta \models \exists_x \varphi & \text{iff } \mathfrak{A}, \theta[x \mapsto a] \models \varphi \text{ for some } a \in A \\
\multicolumn{4}{l}{\mathfrak{A}, \theta \models [\mathbf{lfp}\, X\bar{x}.\, \varphi]\bar{t} \text{ iff } \bar{t} \in \mathbf{lfp}\, \mathbf{T}^{\mathfrak{A},\theta}_{\varphi,X,\bar{x}}, \text{ where } \mathbf{T}^{\mathfrak{A},\theta}_{\varphi,X,\bar{x}}(R) = \{\bar{a} \mid \mathfrak{A}, \theta[X \mapsto R, \bar{x} \mapsto \bar{a}] \models \varphi\}}
\end{array}$$

If $\mathfrak{A}, \theta \models \varphi$, then we say that $\varphi$ is *valid* in $\mathfrak{A}$ and $\theta$. If $\mathfrak{A}, \theta \models \varphi$ for all environments $\theta$, then we simply say that $\varphi$ is *valid* in $\mathfrak{A}$ and write $\mathfrak{A} \models \varphi$. We write $\mathfrak{A}, \theta \not\models \varphi$ to indicate that $\mathfrak{A}, \theta \models \varphi$ does not hold.

To get a more succinct presentation, whenever $\mathfrak{A}$, $\theta$, $\varphi$, $X$ and $\bar{x}$ are clear from the

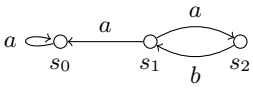



**Figure 1** An LTS as a first-order structure.

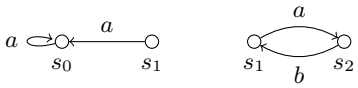



**Figure 2** Evidence for $\varphi_1$ and $\varphi_2$ in $\mathfrak{A}$.

context, we will omit the super- and subscripts of $\mathbf{T}^{\mathfrak{A},\theta}_{\varphi,X,\bar{x}}$ and simply write $\mathbf{T}$. For an in-depth treatment of fixpoint theory, see *e.g.* [19, 20].

The notions of free and bound variables of a formula $\varphi$ are defined as usual. By $\mathsf{fv}(\varphi)$ we denote the set of all variables with a free occurrence in $\varphi$. We sometimes need to express that environments $\theta$ and $\theta'$ agree on the free variables of some formula $\varphi$; we write $\theta \equiv_{\mathsf{fv}(\varphi)} \theta'$ if $\theta(x) = \theta'(x)$ for all first-order variables $x \in \mathsf{fv}(\varphi)$ and $\theta(X) = \theta'(X)$ for all second-order variables $X \in \mathsf{fv}(\varphi)$. Furthermore, we assume a *unique-naming convention*, by which a variable does not have both free and bound occurrences in the formula, and by which a variable is only bound by a single binder. Only in a few concrete examples we deviate from this convention in favour of readability.

The constructs $\wedge$, $\forall\_\ \_$ and $[\mathbf{gfp}\ \_\ \_.\ \_]\_$ are usually treated as syntactic abbreviations: $\varphi \wedge \psi$ for $\neg(\neg\varphi \vee \neg\psi)$, $\forall_x\ \varphi$ for $\neg\exists_x\ \neg\varphi$, and $[\mathbf{gfp}\, X\bar{x}.\, \varphi]\bar{t}$ for $\neg[\mathbf{lfp}\, X\bar{x}.\, \neg\varphi[X \mapsto \neg X]]\bar{t}$, where $\varphi[X \mapsto \neg X]$ denotes the formula obtained from $\varphi$ by replacing all free occurrences of $X$ in $\varphi$ by $\neg X$. For our theory of evidence, to be presented in the next section, it will be convenient, however, to associate proof graphs directly with formulas in the extended syntax, including, in particular, the syntactic construct $[\mathbf{gfp}\ \_\ \_.\ \_]\_$ as first-class citizen.

If a formula $\psi$ has a subformula of the form $[\sigma\, X\bar{x}.\, \chi]\bar{t}$ (with $\sigma \in \{\mathbf{lfp}, \mathbf{gfp}\}$), then we say that $X$ is *defined* in $\psi$, refer to the subformula $[\sigma\, X\bar{x}.\, \chi]\bar{t}$ as the subformula *defining* $X$, and refer to $\chi$ as the definition of $X$. We also let $\sigma_{\psi,X} = \sigma$, $\bar{x}_{\psi,X} = \bar{x}$ and $\varphi_{\psi,X} = \chi$. Usually, the intended formula $\psi$ will be clear from the context, and can therefore be safely omitted as a subscripts in these notations. By $\mathsf{dv}(\psi)$ we denote the set of all second-order variables defined in $\psi$. The formula $\psi$ induces a partial order $<_\psi$ on $\mathsf{dv}(\psi)$ defined by $X <_\psi Y$ if, and only if, the subformula defining $Y$ is a subformula of the subformula defining $X$.

▶ **Example 1.** A labelled transition system can be seen as a first-order structure with the set of states as domain of discourse, a binary relation symbol $\xrightarrow{a}$ for every transition label $a$, and a constant symbol for every state (see Figure 1).

LFP formulas can be used to express properties of a transition system. For instance,

$$\varphi_1 = [\mathbf{lfp}\, Xs.\, \forall_{s'}\ s \xrightarrow{a} s' \implies Xs']s_1$$

expresses that only finitely many consecutive $a$-transitions can be taken from the state denoted by the constant $s_1$; this formula is not valid on the labelled transition system in Figure 1. The LFP formula

$$\varphi_2 = [\mathbf{gfp}\, Xs.\, [\mathbf{lfp}\, Ys'.\, \exists_{s''}\ (s' \xrightarrow{a} s'' \wedge Ys'') \vee (s' \xrightarrow{b} s'' \wedge Xs'')]s]s_1$$

expresses – about a system in which every transition is labelled with either $a$ or $b$ – that from the state $s_1$ there is a path with infinitely many $b$-transitions; this formula is valid on the labelled transition system in Figure 1.

## 3 Evidence based on proof graphs

In this section, we adapt proof graphs, which were originally introduced in [9] for PBESs, to the setting of LFP. Intuitively, a proof graph serves to capture the essence of a reasoning

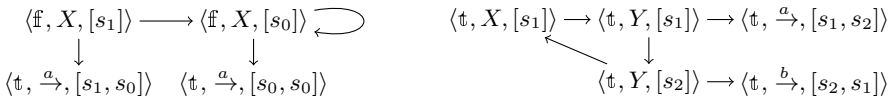



**Figure 3** Proof graphs for $\mathfrak{A} \not\models \varphi_1$ and $\mathfrak{A} \models \varphi_2$.

that proves or refutes the validity of an LFP formula in a first-order structure. The notion of proof graph will be tailored for the purpose of identifying evidence for the validity or invalidity of an LFP formula inside the first-order structure. Such evidence is a, preferably smaller, weak substructure that still admits the reasoning for the validity or invalidity as captured by the proof graph.

Before we formally introduce the notion of proof graph and the associated notion of evidence, we illustrate the idea by means of some examples.

▶ **Example 2.** Consider the LFP formulas of Example 1. To support our claim that $\varphi_1$ is not valid on the labelled transition system in Figure 1, we construct the proof graph depicted on the left in Figure 3. Let us represent by $\langle \mathbb{f}, X, [s_1] \rangle$ the statement that the relation defined by $X$ in $\varphi_1$ does not hold on $s_1$. According to the definition of $X$ in $\varphi_1$, to refute that $X$ holds on $s_1$ it is enough to find a state $s'$ such that $s_1 \xrightarrow{a} s'$ and then refute that $X$ holds on $s'$. The state $s_0$ appears to be a good candidate for $s'$; in our proof graph we will therefore include dependencies from $\langle \mathbb{f}, X, [s_1] \rangle$ to both $\langle \mathbb{t}, \xrightarrow{a}, [s_1, s_0] \rangle$ (expressing that $s_1 \xrightarrow{a} s_0$) and $\langle \mathbb{f}, X, [s_0] \rangle$ (expressing that $X$ does not hold on $s_0$). Similarly, to refute that $X$ holds on $s_0$ it suffices to include a dependency from $\langle \mathbb{f}, X, [s_0] \rangle$ to $\langle \mathbb{t}, \xrightarrow{a}, [s_0, s_0] \rangle$ and to itself.

Note that the reasoning expressed by this graph only involves the states $s_0$ and $s_1$ and the transitions $s_0 \xrightarrow{a} s_0$ and $s_1 \xrightarrow{a} s_0$, which suggests that the weak substructure of $\mathfrak{A}$ depicted on the left in Figure 2 provides sufficient evidence for the invalidity of $\varphi_1$ in $\mathfrak{A}$.

A similar reasoning can be held to construct the proof graph on the right in Figure 3 to show that $\varphi_2$ is valid in $\mathfrak{A}$, resulting in the evidence depicted on the right in Figure 2.

The proof graphs constructed in the example above explain the validity or invalidity of an LFP formula in a presupposed first-order structure in terms of primitive relations defined directly on the structure and the relations defined as fixpoints in the formula. Thus, they abstract from the standard first-order reasoning involved. Proof graphs are going to be the starting point for our theory of diagnostics for LFP formulas. We shall define evidence for the validity or invalidity of an LFP formula $\varphi$ in a first-order structure $\mathfrak{A}$, based on a proof graph $G$, as a weak substructure of $\mathfrak{A}$ for which $G$ is still a proof graph. For the approach to be valid, the information about $\mathfrak{A}$ that is included in $G$ should be both correct and sufficient. Correctness means that whenever a proof graph includes a node $\langle \alpha, V, \bar{a} \rangle$, then, depending on whether $\alpha$ is $\mathbb{t}$ or $\mathbb{f}$, the sequence $\bar{a}$ should or should not be in the relation on $\mathfrak{A}$ denoted by $V$. Sufficiency means that $G$ includes enough information from $\mathfrak{A}$ to facilitate reconstruction of the reasoning reflected by $G$ in every weak substructure of $\mathfrak{A}$ that inherits at least that information from $\mathfrak{A}$.

We shall now first formally define the notion of proof graph for a first-order structure $\mathfrak{A}$, an LFP formula $\varphi$, and an environment $\theta$, and then explain how it gives rise to a formal notion of evidence.

## 3.1 Proof graphs

The nodes of a proof graph are tuples of the form $\langle \alpha, V, \bar{a} \rangle$ in which $\alpha$ is either $\mathbb{t}$ or $\mathbb{f}$, $V$ is either a relation symbol or a second-order variable, and $\bar{a}$ is a sequence of elements of $\mathfrak{A}$ the

length of which corresponds to the arity of $V$. We denote by $\mathcal{S}$ the set of all such nodes, and, for $\mathcal{Y} \subseteq \mathcal{R} \cup \mathcal{X}$, we denote by $\mathcal{S}_\mathcal{Y}$ the subset of $\mathcal{S}$ consisting of all nodes of which the second element is from $\mathcal{Y}$, *i.e.*,

$$\mathcal{S}_\mathcal{Y} = \{\langle \alpha, V, \bar{a}\rangle \in \{\mathbb{t}, \mathbb{f}\} \times \mathcal{Y} \times A^* \mid \mathsf{ar}(X) = |\bar{a}|\}.$$

We will often use the subscript to refer to nodes in which only specific relations or variables occur, *e.g.*, $S_{\{X\}}$ for all nodes of which the second element is the second-order variable $X$. To concisely express that the statement represented by a node $\langle \alpha, V, \bar{a}\rangle$ is true in a structure $\mathfrak{A}$ and an environment $\theta$, we write

$$\mathfrak{A}, \theta \models \langle \alpha, V, \bar{a}\rangle \quad \text{for} \quad \begin{cases} \bar{a} \in \theta(V) \iff \alpha = \mathbb{t} & \text{if } V \in \mathcal{X}, \text{ and} \\ \bar{a} \in V^{\mathfrak{A}} \iff \alpha = \mathbb{t} & \text{if } V \in \mathcal{R}. \end{cases}$$

The purpose of the dependency relation of a proof graph is to reflect a sound and complete reasoning for the truth of each of its nodes of the form $\langle \alpha, X, \bar{a}\rangle$, with $X$ a variable defined in $\varphi$. Our definition of proof graph, below, will impose requirements on the dependency relation to ensure that the reflected reasoning is sound and complete. First, we present a local requirement on the dependency relation to ensure that the truth of a node $\langle \alpha, X, \bar{a}\rangle$ can, according to the definition of $X$ in $\varphi$, be inferred from the truth of all its successors together. This local requirement should, moreover, be satisfied in every substructure with sufficient information from the perspective of the reflected reasoning. Thus, we obtain the intermediate notion of *dependency graph*, which only admits reasonings of which every individual inference step is justified. To obtain a suitable notion of proof graph, we then still need to add a global requirement that excludes non-wellfounded reasonings to the extent that they are invalid. We shall establish that the resulting notion of proof graph is sound and complete: all nodes in a proof graph for $\mathfrak{A}$, $\theta$ and $\varphi$ are true in $\mathfrak{A}$ and $\theta$, and if a node is true in $\mathfrak{A}$ and $\theta$, then there is a proof graph that includes it.

Consider a node $v = \langle \alpha, X, \bar{a}\rangle$, with $X$ a variable defined in $\varphi$, and let $\varphi_X$ be the definition of $X$ in $\varphi$. We denote by $v^\bullet$ the set of successors (the *postset*) of $v$. To express that if the elements of $v^\bullet$ are all true, then $\varphi_X$ forces $v$ to be true as well, we need to be able to influence the values of the second-order variables defined in $\varphi_X$. Let us denote by $\mathsf{fo}(\varphi_X)$ the formula obtained from $\varphi_X$ by replacing every $[\mathbf{lfp}\, Y\bar{x}.\, \psi]\bar{t} \sqsubseteq \varphi_X$ by $Y\bar{t}$. We can then consider environments satisfying the nodes in $v^\bullet$, and require that such environments satisfy $\mathsf{fo}(\varphi_X)$ if, and only if, $\alpha = \mathbb{t}$. To avoid having to distinguish between the different values for $\alpha$ every time, we introduce the following shorthand notation:

$$\mathfrak{A}, \theta \overset{\alpha}{\models} \varphi \quad \text{denotes} \quad \mathfrak{A}, \theta \models \varphi \iff \alpha = \mathbb{t}.$$

Recall that we want to use proof graphs to define a notion of evidence for the validity or invalidity within a first-order structure $\mathfrak{A}$. Given a proof graph with nodes $S$, we will be looking at weak substructures of $\mathfrak{A}$ that have a domain of discourse that includes all the elements that are referenced by the nodes in $S$. We denote by $\mathfrak{A} \restriction S$ the smallest (with respect to $\sqsubseteq_\mathrm{w}$) weak substructure of $\mathfrak{A}$ of which the domain of discourse is a superset of $\{a \in A \mid \exists_{\langle \alpha, V, \bar{c}\rangle \in S}\ a \in \bar{c}\}$. Note that, according to the definition of weak substructure, if $\mathcal{I}$ is the interpretation function of $\mathfrak{A} \restriction S$, then $\mathcal{I}(R) = \emptyset$ for all first-order relation symbols $R$.

We enforce consistency of the reasoning represented by the graph by requiring that the successors of a node are never contradictory: a relation $\to \subseteq \mathcal{S} \times \mathcal{S}$ is *consistent* if and only if for all $v$, $X$ and $\bar{a}$, not both $\langle \mathbb{t}, X, \bar{a}\rangle \in v^\bullet$ and $\langle \mathbb{f}, X, \bar{a}\rangle \in v^\bullet$.

▶ **Definition 3** (dependency graph)**.** A *dependency graph* for $\mathfrak{A}, \theta$ and $\varphi$ is a directed graph $\langle S, \to\rangle$ with $S \subseteq \mathcal{S}$ and $\to \subseteq S \times S$, such that $\to$ is consistent, and for all $\langle \alpha, V, \bar{a}\rangle \in S$:

- if $V \in \mathsf{dv}(\varphi)$:

  for all $\mathfrak{A} \restriction S \sqsubseteq_{\mathrm{w}} \mathfrak{B} \sqsubseteq_{\mathrm{w}} \mathfrak{A}$ and all $\eta$ such that $\eta \equiv_{\mathsf{fv}(\varphi)} \theta$,
  if $\mathfrak{B}, \eta \models v$ for all $v \in \langle \alpha, V, \bar{a} \rangle^{\bullet}$ then $\mathfrak{B}, \eta[\bar{x}_V \mapsto \bar{a}] \stackrel{\alpha}{\models} \mathsf{fo}(\varphi_V)$

- if $V \notin \mathsf{dv}(\varphi)$:

$$\mathfrak{A}, \theta \models \langle \alpha, V, \bar{a} \rangle \qquad \text{and} \qquad \langle \alpha, V, \bar{a} \rangle^{\bullet} = \emptyset$$

▶ **Example 4.** Let $\mathfrak{A}$ be the transition system in Figure 1. The left graph in Figure 3 is a dependency graph for $\mathfrak{A}$, any $\theta$ and $\varphi_1$, and the right graph in Figure 3 is a dependency graph for $\mathfrak{A}$, any $\theta$ and $\varphi_2$. Let us verify in some detail that the left graph in Figure 3 indeed satisfies the conditions of dependency graphs.

Clearly, the dependency relation of the graph in Figure 3 is consistent. Furthermore, the nodes $\langle \mathbb{t}, \xrightarrow{a}, [s_0, s_0] \rangle$ and $\langle \mathbb{t}, \xrightarrow{a}, [s_1, s_0] \rangle$ do not have successors and they are true in $\mathfrak{A}$.

For the nodes $\langle \mathbb{f}, X, [s_0] \rangle$ and $\langle \mathbb{f}, X, [s_1] \rangle$ we need to check the first condition of Definition 3. We show how to check the condition for $\langle \mathbb{f}, X, [s_0] \rangle$, the check for $\langle \mathbb{f}, X, [s_1] \rangle$ is similar. Consider a weak substructure $\mathfrak{B}$ of $\mathfrak{A}$ and an environment $\eta$ ($\eta \equiv_{\mathsf{fv}(\varphi_1)} \theta$ holds for every $\eta$ because $\mathsf{fv}(\varphi_1) = \emptyset$) such that $\mathfrak{B}, \eta \models \langle \mathbb{f}, X, [s_0] \rangle$ and $\mathfrak{B}, \eta \models \langle \mathbb{t}, \xrightarrow{a}, [s_0, s_0] \rangle$. The latter means that $\mathfrak{B}, \eta \models s_0 \xrightarrow{a} s_0$, and the former means that $\mathfrak{B}, \eta \not\models X s_0$, so $\mathfrak{B}, \eta \not\models s_0 \xrightarrow{a} s_0 \implies X s_0$. This is equivalent to $\mathfrak{B}, \eta^{\mathbb{f}} \models \mathsf{fo}(\varphi_X)$.

A least fixpoint is proved by an inductive argument, which is necessarily well-founded. A greatest fixpoint is proved by a coinductive argument, which need not be well-founded. In fact, a coinductive argument of a statement can be thought of as a reasoning that argues the absence of a well-founded reasoning for the negation of the statement. The quantification over all well-founded reasonings gives rise to a reasoning that is inherently not well-founded. A dependency graph admits reasonings that are not well-founded without taking into account whether such reasonings are justified. We formulate an extra requirement on dependency graphs to filter out invalid non-well-founded reasonings. Let $I_X(\pi)$ denotes the set of indices $i$ on path $\pi$ such that $\pi_i \in \mathcal{S}_{\{X\}}$.

▶ **Definition 5** (proof graph)**.** A dependency graph $G = \langle S, \to \rangle$ for $\mathfrak{A}, \theta$ and $\varphi$ is a *proof graph* iff for every infinite path $\pi$ in $G$, for all $X$ minimal w.r.t. $<_\varphi$ such that $I_X(\pi)$ is infinite it holds that:

- if $\sigma_X = \mathbf{lfp}$, then $\{i \in I_X(\pi) \mid \exists_{\bar{a}}\ \pi_i = \langle \mathbb{t}, X, \bar{a} \rangle\}$ is finite; and
- if $\sigma_X = \mathbf{gfp}$, then $\{i \in I_X(\pi) \mid \exists_{\bar{a}}\ \pi_i = \langle \mathbb{f}, X, \bar{a} \rangle\}$ is finite.

▶ **Example 6.** As we saw in Example 4, the graphs in Figure 3 are dependency graphs; it remains to verify that they satisfy the condition of Definition 5 to conclude that they are proof graphs. For the proof graph on the right in Figure 3 the reasoning is as follows: On every infinite path $\pi$ in the graph, each of the three nodes $\langle \mathbb{t}, X, [s_1] \rangle$, $\langle \mathbb{t}, Y, [s_1] \rangle$, and $\langle \mathbb{t}, Y, [s_2] \rangle$ occur infinitely often. Note that $X$ is the (only) $<_{\varphi_2}$-minimal second-order variable on such an infinite path. We have that $\sigma_{\varphi_2, X} = \mathbf{gfp}$, and, indeed, the set $\{i \in I_X(\pi) \mid \exists_{\bar{a}}\ \pi_i = \langle \mathbb{f}, X, \bar{a} \rangle\}$ is finite.

The following theorem, which is proved for an equally expressive, but syntactically more general fixpoint logic (it has LFP as a normal form) in [6], establishes that the notion of proof graph defined above is sound and complete for capturing reasoning about the validity of relations defined by fixpoints in LFP formulas. Soundness here means that whenever a proof graph for a structure $\mathfrak{A}$, an environment $\theta$ and a formula $\varphi$ includes a node $\langle \alpha, X, \bar{a} \rangle$ with $X$ a second-order variable defined in $\varphi$, then $\langle \alpha, X, \bar{a} \rangle$ expresses a true statement with

respect to $\mathfrak{A}$, $\theta$ and $\varphi$. That is, the relation on $\mathfrak{A}$ associated with $X$ by $\varphi$ includes $\bar{a}$ if $\alpha = \mathbb{t}$, and the relation on $\mathfrak{A}$ associated with $X$ by $\varphi$ does not include $\bar{a}$ if $\alpha = \mathbb{f}$. Conversely, completeness means that if $\langle \alpha, X, \bar{a}\rangle$ expresses a true statement with respect to $\mathfrak{A}$, $\theta$ and $\varphi$, then there exists a proof graph for $\mathfrak{A}$, $\theta$ and $\varphi$ including this node.

▶ **Theorem 7.** *Let $\mathfrak{A}$ be a first-order structure, let $\theta$ be an environment, let $\varphi$ be an LFP formula, and let $X$ be a second-order variable defined in $\varphi$. Then, for all $\langle \alpha, X, \bar{a}\rangle \in \mathcal{S}_{\{X\}}$, the following are equivalent:*

- *There exists a proof graph for $\mathfrak{A}$, $\theta$ and $\varphi$ that includes the node $\langle \alpha, X, \bar{a}\rangle$.*
- $\bar{a} \in \sigma_X \mathbf{T}^{\mathfrak{A},\theta}_{\varphi_X, X, \bar{x}_X} \iff \alpha = \mathbb{t}$,

The notion of proof graph defined above deviates in two respects from the notion introduced for PBESs in [9]. Firstly, the new notion requires nodes to be included that capture the information about the first-order relations in the original structure that is needed in the reasoning reflected by the proof graph. These nodes will be used in Section 3.2 to extract an appropriate weak substructure that can serve as evidence. Secondly, the new notion includes an extra $\{\mathbb{t}, \mathbb{f}\}$-element in a node, which facilitates the inclusion in a proof graph of both positive and negative statements as to whether relations hold for certain sequences of elements, and an associated additional consistency requirement for the dependency relation. Such a facility was not needed in the setting of PBESs by the absence of negation.

## 3.2 Evidence

We proceed to define a general notion of evidence based on proof graphs. In Section 4 we illustrate how this notion can be used to provide diagnostics for behavioural equivalence checking, and we will show that this notion specialises to familiar notions of evidence in the context of model checking.

It follows from Theorem 7 that for LFP formulas of the form $[\sigma X\bar{x}.\,\varphi]\bar{t}$ we have that

$$\mathfrak{A}, \theta \overset{\alpha}{\models} [\sigma X\bar{x}.\,\varphi]\bar{t} \quad \text{iff} \quad \begin{array}{l}\text{there exists a proof graph for } \mathfrak{A}, \theta \text{ and } [\sigma X\bar{x}.\,\varphi]\bar{t} \\ \text{that includes a node } \langle \alpha, X, \bar{t}^{\mathfrak{A},\theta}\rangle.\end{array}$$

We shall refer to a proof graph for $\mathfrak{A}$, $\theta$ and $[\sigma X\bar{x}.\,\varphi]\bar{t}$ that includes the node $\langle \alpha, X, \bar{t}^{\mathfrak{A},\theta}\rangle$ as *a proof graph for* $\mathfrak{A}, \theta \overset{\alpha}{\models} [\sigma X\bar{x}.\,\varphi]\bar{t}$. Since, in fact, every LFP formula $\psi$ is equivalent to $[\mathbf{lfp}\, X_0.\,\psi]$, provided that $X_0 \notin \mathsf{dv}(\psi) \cup \mathsf{fv}(\psi)$, we may use the terminology more generally: a proof graph for $\mathfrak{A}, \theta \overset{\alpha}{\models} \psi$ is a proof graph for $\mathfrak{A}, \theta \overset{\alpha}{\models} [\mathbf{lfp}\, X_0.\,\psi]$ if $\psi$ is not already of the form $[\sigma X\bar{x}.\,\varphi]\bar{t}$ for some $X$, $\bar{x}$, $\varphi$ and $\bar{t}$. We now propose the following definition of evidence for LFP formulas, which formalises that evidence based on some proof graph for the (in)validity of an LFP formula in a first-order structure is that part of the structure that facilitates the reasoning reflected by that proof graph.

▶ **Definition 8** (evidence)**.** By *evidence* for $\mathfrak{A}, \theta \overset{\alpha}{\models} \varphi$ we mean a weak substructure $\mathfrak{B} \sqsubseteq_{\mathrm{w}} \mathfrak{A}$ such that there is a proof graph for $\mathfrak{B}, \theta \overset{\alpha}{\models} \varphi$ that is also a proof graph for $\mathfrak{A}, \theta \overset{\alpha}{\models} \varphi$.

Naturally, we would like evidence to be as concise as possible, and so we would like to obtain, given a proof graph for $\mathfrak{A}, \theta \overset{\alpha}{\models} \varphi$, the smallest weak substructure of $\mathfrak{A}$ that serves as suitable evidence.

▶ **Definition 9** (evidence projection)**.** Given a proof graph $\langle S, \to\rangle$ for $\mathfrak{A}, \theta \overset{\alpha}{\models} \varphi$, define $\mathsf{ev}(\langle S, \to\rangle)$ as the smallest $\mathfrak{B} \sqsubseteq_{\mathrm{w}} \mathfrak{A}$ such that for each node $\langle \beta, V, \bar{a}\rangle \in S$ (with $V \in \mathcal{R} \cup \mathcal{X}$) we have $\bar{a} \in B^*$, and for each $v \in S \setminus \mathcal{S}_{\mathcal{X}}$ we have $\mathfrak{B}, \theta \models v$.

It is easy to see that $\mathsf{ev}(\langle S, \to \rangle)$ as defined in the preceding definition always exists, and that it can be computed straightforwardly from $\mathfrak{A} \restriction S$ by adding, for every node $\langle \mathbb{t}, R, \bar{a} \rangle$, the sequence $\bar{a}$ to the interpretation of relation symbol $R$.

▶ **Theorem 10.** *If $G$ is a proof graph for $\mathfrak{A}, \theta \models^{\alpha} \varphi$, then $\mathsf{ev}(G)$ is evidence for $\mathfrak{A}, \theta \models^{\alpha} \varphi$.*

Our notion of evidence projection results in the smallest evidence for $\mathfrak{A}, \theta \models^{\alpha} \varphi$ given a particular proof graph. There may, however, be many proof graphs for $\mathfrak{A}, \theta \models^{\alpha} \varphi$, and it appears that, roughly speaking, smaller proof graphs lead to smaller evidence. A proof graph may contain redundant information; a subgraph that contains no redundant nodes or edges is called a *minimal* proof graph. We refer to [9] for a more elaborate discussion on minimality.

## 4 Counterexamples and witnesses

Some problems that can be encoded in fixpoint logic consist of checking an 'implementation' against a 'specification'. For instance, if the behaviour of some system is described as a Kripke structure, and we want to establish correctness properties on that Kripke structure, then we may view it as an 'implementation' of sorts, which we could check against a set of CTL* formulas, the 'specifications'. We might also want to check if this Kripke structure refines another, more abstract Kripke structure. In this case, the 'specification' is not a formula, but another Kripke structure. We refer to such problems as *model checking* problems.

For problems that have this characteristic of a division into implementation and specification, we tend to think of the specification as being given and well-understood, whereas the implementation may contain mistakes that need to be clarified with diagnostics. Such diagnostics should highlight the parts of the implementation that cause a problem, but should not include details from the specification. To achieve this, we propose a general scheme that combines an implementation $\mathfrak{A}$ with a specification $\mathfrak{B}$, and that extracts the information from $\mathfrak{A}$ from evidence relating to the combined structure. These combination and extraction operations are defined in terms of the operators $\cup$ and $\cap$. Essentially, the $\cup$ operator must merge two structures $\mathfrak{A}$ and $\mathfrak{B}$ together, and the $\cap$ operator must be able to retrieve a weak substructure of $\mathfrak{A}$ again from the merged structure. Natural candidates to implement these operations on the domain of discourse of the two structures are the set union and set intersection operations. The function and relation symbols are also obtained by taking the set union or intersection of the symbols in $\mathfrak{A}$ and $\mathfrak{B}$.

If $R$ is a relation symbol with interpretations in both $\mathfrak{A}$ and $\mathfrak{B}$, then $R^{\mathfrak{A} \cup \mathfrak{B}} = R^{\mathfrak{A}} \cup R^{\mathfrak{B}}$ and $R^{\mathfrak{A} \cap \mathfrak{B}} = R^{\mathfrak{A}} \cap R^{\mathfrak{B}}$, and if $R$ only has an interpretation in $\mathfrak{A}$ (resp. $\mathfrak{B}$), then $\mathcal{R}^{\mathfrak{A} \cup \mathfrak{B}} = \mathcal{R}^{\mathfrak{A}}$ (resp. $R^{\mathfrak{A} \cup \mathfrak{B}} = R^{\mathfrak{B}}$). A natural way to define the interpretation of a function symbol $f$ in $\mathfrak{A} \cap \mathfrak{B}$ is to define $f^{\mathfrak{A} \cap \mathfrak{B}}$ as the restriction of $f^{\mathfrak{A}}$ to the new domain of discourse, $A \cap B$. Defining the interpretation for $f$ in $\mathfrak{A} \cup \mathfrak{B}$ is problematic however, if $f^{\mathfrak{A}}$ and $f^{\mathfrak{B}}$ do not agree on the intersection of their domains. If they *do* agree on this intersection, we can define $f^{A \cup \mathfrak{B}}$ such that it assigns to every input the same output as $f^{\mathfrak{A}}$ does if the input is in the domain of $f^{\mathfrak{A}}$, or the output of $f^{\mathfrak{B}}$ if the input is in the domain of $f^{\mathfrak{B}}$.

For pairs of structures in which the interpretation of the function symbols are compatible in this way – we will call such structures *composable* – we define union and intersection operators $\cup$ and $\cap$ as described above. Using these operators, witnesses and counterexamples can be extracted as follows.

▶ **Definition 11.** If $\mathfrak{A}$ and $\mathfrak{B}$ are composable structures, $\mathfrak{E} \sqsubseteq_{\mathrm{w}} \mathfrak{A} \cup \mathfrak{B}$, $\theta$ is an environment and $\varphi$ is an LFP formula such that $\mathfrak{E}$ is evidence for $\mathfrak{A} \cup \mathfrak{B}, \theta \models^{\alpha} \varphi$, then we call $\mathfrak{E} \cap \mathfrak{A}$ an $\mathfrak{A}$-*witness* if $\alpha = \mathbb{t}$. We call it an $\mathfrak{A}$-*counterexample* if $\alpha = \mathbb{f}$.

If, from the context, it is clear which structure was used to extract the $\mathfrak{A}$-witness, we simply refer to the resulting structure as a witness; likewise for counterexamples. A desirable property of a witness is that it can be related to the structure from which it is derived. Furthermore, a witness should contain all the information that is essential in proving the same LFP formula $\varphi$.

▶ **Theorem 12** ([6]). *If $\mathfrak{A}$ and $\mathfrak{B}$ are composable, $\theta$ is an environment, $\varphi$ is an LFP formula over $\mathfrak{A} \cup \mathfrak{B}$, and $\mathfrak{C}$ is an $\mathfrak{A}$-counterexample or $\mathfrak{A}$-witness for $\mathfrak{A} \cup \mathfrak{B}, \theta \models^{\alpha} \varphi$, then*

$$\mathfrak{C} \sqsubseteq_w \mathfrak{A} \quad \text{and} \quad \mathfrak{C} \cup \mathfrak{B}, \theta \models^{\alpha} \varphi.$$

Usually, $\varphi$ will be a closed formula, in which case the value of $\theta$ is irrelevant. In such cases, we will not explicitly mention $\theta$, but assume that an arbitrary environment is given. In the following sections we will give an example of a formula that encodes stuttering equivalence checking, in which case $\mathfrak{A}$ and $\mathfrak{B}$ are Kripke structures, and an example of a formula that encodes ∃ECTL* model checking, in which case $\mathfrak{A}$ is a Kripke structure, and $\mathfrak{B}$ is a structure that represents an ∃ECTL* formula. This approach differs from those in [3, 14, 15], in which a different fixpoint formula is generated for every $\mathfrak{A}$ and $\mathfrak{B}$.

## 4.1 Counterexamples for stuttering bisimulation checking

To illustrate the use of the $\cup$ and $\cap$ operators on structures, and to illustrate how counterexamples can be extracted for an equivalence checking problem, we consider the problem of checking that two systems are stuttering bisimilar. We use Namjoshi's formulation of stuttering bisimulation [21], because it already closely resembles our definition in LFP.

▶ **Definition 13.** Given a Kripke structure $\langle A, AP, \to, \ell \rangle$, a relation $X \subseteq A \times A$ is a *stuttering bisimulation* if and only if it is symmetric, and there exist a well-founded order $\langle W, \prec \rangle$ and some mapping $rank : A \times A \times A \to W$ such that for all $s, t$ such that $Xst$:

$$\begin{aligned} \ell(s) = \ell(t) \wedge \forall_u\ s \to u \implies & ((Xut \wedge rank(u,u,t) \prec rank(s,s,t)) \vee \\ & \exists_v\ t \to v \wedge ((Xsv \wedge rank(u,s,v) \prec rank(u,s,t)) \vee Xuv)). \end{aligned}$$

States $s$ and $t$ are said to be *stuttering bisimilar*, denoted $s \simeq t$, if a stuttering bisimulation exists that relates $s$ and $t$.

▶ **Proposition 1** ([6]). *Let $\mathfrak{A}$ be a Kripke structure $\langle A, AP, \to, \ell \rangle$.*

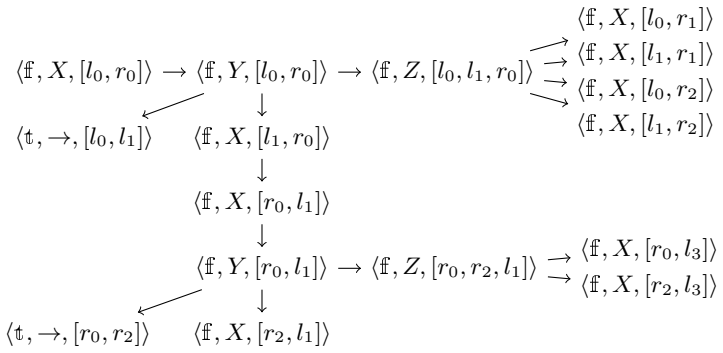
$$\begin{aligned} \Phi lr \triangleq\ & [\mathbf{gfp}\, Xst.\, Xts \wedge \ell(s) = \ell(t) \wedge \\ & [\mathbf{lfp}\, Yst.\, \forall_u\ s \to u \implies ((Xut \wedge Yut) \vee \\ & [\mathbf{lfp}\, Zsut.\, \exists_v\ t \to v \wedge ((Xsv \wedge Zsuv) \vee Xuv))]sut]st]lr \end{aligned}$$

*If $l$ and $r$ are terms of $\mathfrak{A}$ and $s = l^{\mathfrak{A}}$ and $t = r^{\mathfrak{A}}$, then $\mathfrak{A} \models \Phi lr$ if and only if $s \simeq t$.*

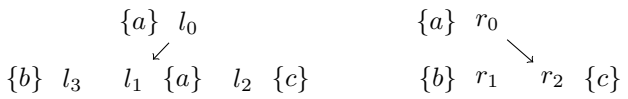
Consider the following two Kripke structures, that are stutter trace equivalent, but not stutter bisimulation equivalent.

$$L = \begin{array}{ccccc} & & \{a\} & l_0 & \\ & & \swarrow & & \searrow \\ \{b\} & l_3 \leftarrow & l_1 & \{a\} & l_2 \quad \{c\} \end{array} \qquad R = \begin{array}{ccccc} \{a\} & r_0 & & & \\ & \downarrow & & \searrow & \\ \{b\} & r_1 & & & r_2 \quad \{c\} \end{array}$$

Let $\mathfrak{A} = L \cup R$, and suppose that $l^L = l_0$ and $r^R = r_0$. Consider the following proof graph for $\mathfrak{A} \not\models \Phi lr$.




To extract evidence from this proof graph, we construct an evidence projection as per Definition 9. That is, we construct a substructure $\mathfrak{B} \sqsubseteq_{\mathrm{w}} \mathfrak{A}$ which must contain at least those states from $L$ and $R$ referred to in the proof graph (all states from $L$ and $R$), and which satisfies $l_0 \to l_1$ and $r_0 \to r_2$. This yields the following Kripke structure $\mathfrak{B}$ as evidence. Note that $\mathfrak{B} \cap L$ and $\mathfrak{B} \cap R$ return the offending parts of $L$ and $R$, respectively.




Observe that in $\mathfrak{B}$, $l_0$ and $r_0$ are again not stuttering bisimilar, and moreover, they can be shown not to be equivalent with the same reasoning: the transition from $l_0$ to a state unrelated to $r_0$ with label $a$ cannot be mimicked by $r_0$. All the states from $\mathfrak{A}$ are retained in the evidence, because the existential quantifier requires an explanation for the invalidity of *every* $X$-node in the proof graph. Taking the projection of the proof graph minimised with respect to $\mathfrak{B}$ would yield evidence in which only the reachable states from $l_0$ and $r_0$ are included.

Other proof graphs are possible, leading to different evidence. For instance, if we had chosen $\langle \mathbb{F}, X, [l_0, r_0] \rangle$ to depend on $\langle \mathbb{F}, X, [r_0, l_0] \rangle$ (using the symmetry of stuttering bisimulation), we could have obtained evidence in which only the edge $r_0 \to r_1$ was retained. The explanation here is that it is sufficient to show that $r_0$ can reach an equivalence class labelled with $b$, without moving through another class first, whereas $l_0$ cannot do so.

We would like to remark that there are alternatives to our notion of evidence for bisimulation and stuttering bisimulation. For instance, a common notion is a distinguishing formula in Hennessy-Milner logic (for bisimulation [5]) or CTL*\X (for stuttering bisimulation [18]). However, in our experience, such formulas tend to get very unwieldy and do not always offer much insight. We believe that our notion of evidence can be a more practical alternative to distinguishing formulas in such cases.

## 4.2 Counterexamples for LTL and ACTL* model checking

In [4], Clarke et al. noted that for certain model checking problems, instead of generating substructures, one can generate counterexamples of a specific form: for LTL model checking, counterexamples are usually defined as a single (possibly infinite) trace through the model that does not satisfy the specification. These traces can again be seen as Kripke structures that do not satisfy the desired property. For model checking ∀CTL* – a subset of CTL* which adds to LTL universal quantification over branches – counterexamples consist of a number of traces that are attached to each other in a tree-like fashion. More formally, a tree-like counterexample is a Kripke structure that can be simulated by the system under

scrutiny, which does not satisfy the desired property, in which every strongly connected component (SCC) consists of a single cycle, and of which the SCC decomposition is a tree.

In the remainder of this section, we show that these special types of counterexample can be obtained from proof graphs. To simplify presentation, we consider the dual problem of generating witnesses for ∃CTL*. Furthermore, to also capture the expressivity of the $\omega$-regular extensions used in [4], we consider the extended logic ∃ECTL* (originally presented in [25]), which uses Büchi automata as primitives. We note that it is also possible to define what follows directly for ∃CTL*, but this requires encoding the translation of LTL to Büchi automata in first-order logic, as was done in [8].

An ∃ECTL* formula $f$ can be described by a structure $\mathfrak{B}_f$ over a domain that includes at least one element for every subformula and every set of subformulas of $f$, and for each subformula of the form $\mathsf{E}(\mathcal{B})$ a unique element for every state of $\mathcal{B}$. We let $\mathfrak{B}_f$ contain an element representing $AP$ and an element representing the set $F$ of accepting Büchi states. We assume that it also includes the usual relations on sets, relations to recover the structure of formulas, and a ternary transition relation $\rightarrow$ for the Büchi automata. To distinguish CTL* operators from Boolean connectives, we add a dot to the CTL* operators: $\dot\neg$ for CTL* negation, and $\dot\wedge, \dot\forall$ for CTL* conjunction and disjunction. A structured LFP encoding of the ∃ECTL* model checking problem is given below.

▶ **Proposition 2.** *Let $\Phi$ be defined as:*

$$\Phi sf \triangleq [\mathbf{lfp}\, Xsf.\ \begin{cases} (f \in AP \wedge f \in \ell(s)) \\ \vee\, \exists_g\ (f = \dot\neg g \wedge g \notin \ell(s)) \\ \vee\, \exists_{g,h}\ (f = g \dot\vee h \wedge (Xsg \vee Xsh)) \\ \vee\, \exists_{g,h}\ (f = g \dot\wedge h \wedge Xsg \wedge Xsh)) \\ \vee\, \exists_b\ (f = \mathsf{E}(\mathcal{B}(b)) \wedge \Psi sb) \end{cases} ]sf$$

$$\Psi sb \triangleq [\mathbf{gfp}\, Ysb.$$
$$[\mathbf{lfp}\, Zsb. \exists_{s',b',g}\ s \rightarrow s' \wedge b \xrightarrow{g} b' \wedge Xsg \wedge$$
$$((b' \in F \wedge Ys'b') \vee (b' \notin F \wedge Zs'b'))]sb]sb$$

*Let $\mathfrak{A}$ be a Kripke structure over $AP$, and let $f$ be an ∃ECTL* formula over $AP$. If $s$ is a term of $\mathfrak{A}$ and $f$ is a term of $\mathfrak{B}$, and $\hat a = s^{\mathfrak{A}}$ and $\hat b = f^{\mathfrak{B}_f}$, then $\mathfrak{A} \cup \mathfrak{B}_f \models \Phi sf$ if and only if $\mathfrak{A}, \hat a \models \hat b$.*

Let $G$ be a minimal proof graph for $\mathfrak{A} \cup \mathfrak{B}_f \models \Phi sf$. Firstly, the first element of all nodes in $G$ that are also in $\mathcal{S}_{\mathcal{X}}$ is equal to $\mathbb{t}$. Notice that per Definition 5, $G$ cannot contain cycles that pass through $\mathcal{S}_{\{X\}}$. Because $G$ is minimal, nodes from $\mathcal{S}_{\{Y,Z\}}$ have exactly one successor in $\mathcal{S}_{\{Y,Z\}}$. Therefore, the only cycles in $G$ are cycles through $\mathcal{S}_{\{Y,Z\}}$, and every node in the proof graph can be in at most one cycle, leading to the following property:

▶ **Property 1.** *Let $G$ be a minimal proof graph for $\mathfrak{A} \cup \mathfrak{B}_f \models \Phi sf$. Then every SCC in $G$ consists of a single cycle.*

▶ **Example 14.** Consider the ∃ECTL* formula $\mathsf{E}(\mathcal{B}(b_0)) \dot\wedge \mathsf{E}(\mathcal{B}(b_1))$, where $b_0$ and $b_1$ are states of the Büchi automata below; the formula expresses that there are infinite $y$-paths and infinite $x$-paths.

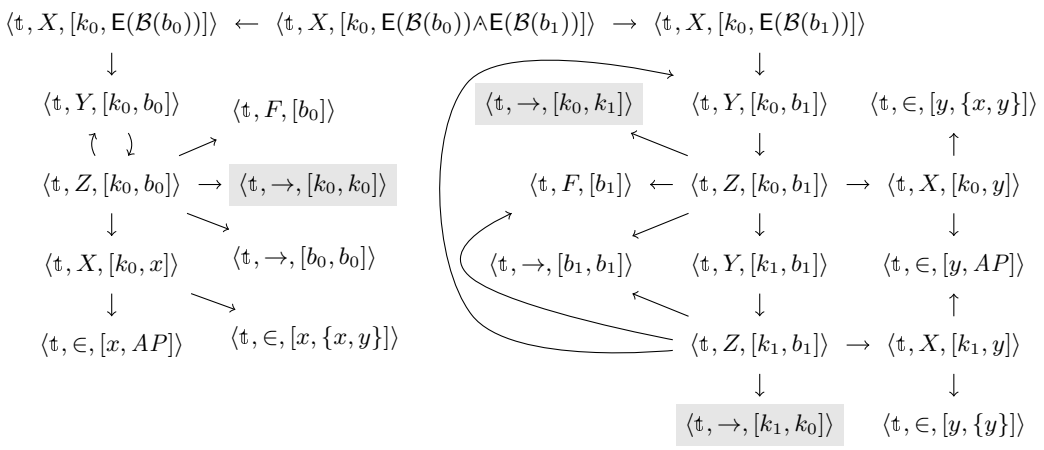



**Figure 4** A proof graph explaining that state $k_0$ satisfies $\mathsf{E}(\mathcal{B}(b_0))\wedge\mathsf{E}(\mathcal{B}(b_1))$.

The state $k_0$ of the Kripke structure satisfies the ∃ECTL* formula. Following Proposition 2, $\Phi k_0(\mathsf{E}(\mathcal{B}(b_0))\wedge\mathsf{E}(\mathcal{B}(b_1)))$ must therefore hold. Indeed, a proof graph explaining this is given in Figure 4. This proof graph is minimal: none of its edges or nodes are redundant. The proof graph contains two SCCs, and no cycle passes through nodes from $\mathcal{S}_{\{X\}}$. The witness obtained from the proof graph of Figure 4, consisting of all Kripke structure states and relations defined by the shaded proof graph nodes, is essentially the original Kripke structure.

Our goal is to obtain a tree-like witness from a proof graph for $\mathfrak{A}\cup\mathfrak{B}_f \models \Phi sf$, if state $\hat{a}$ satisfies ∃ECTL* formula $\hat{b}$. We do so by first finding a witness in which every SCC is again a single cycle. We can however not use the witness obtained from $G$ by Definition 11, because disjoint cycles in $G$ may correspond to cycles in $\mathfrak{A}$ that share nodes. This may lead to a witness with SCCs that are no longer simple cycles.

To ensure that SCCs become simple cycles, in [4], cycles that share a subset of their nodes are 'unrolled'. In *ibid.* this is done by running a model checking algorithm not on $\mathfrak{A}$, but on a bisimilar *indexed Kripke structure* $\mathfrak{A}^\omega$, which contains for every cycle in $\mathfrak{A}$ an infinitely unrolled path. We adopt the same approach: we transform $G$ to a proof graph for $\mathfrak{A}^\omega\cup\mathfrak{B}_f \models \Phi sf$, and then use Definition 11 to extract a witness from this proof graph.

▶ **Definition 15.** Given a Kripke structure $\mathfrak{A} = \langle A, AP, \to, \ell\rangle$, its corresponding *indexed Kripke structure* $\mathfrak{A}^\omega$ is the Kripke structure $\langle A^\omega, AP, \to^\omega, \ell^\omega\rangle$ such that:

- $A^\omega = A\times\mathbb{N}$,
- $\to^\omega$ is such that $\langle a,i\rangle \to^\omega \langle a',j\rangle$ iff $a\to a'$ (for all $a, a', i$ and $j$),
- $\ell^\omega$ is such that $\ell^\omega(\langle a,i\rangle) = \ell(a)$.

Note that every $a\in A$ is bisimilar to all $\langle a,i\rangle\in A^\omega$. Therefore, fixing some $i\in\mathbb{N}$ and replacing every $a\in A$ occurring as a parameter of a node in $G$ by $\langle a,i\rangle$ yields a valid proof graph $G^i$ (for $\mathfrak{A}^\omega\cup\mathfrak{B}_f$). For distinct $i$ and $j$, the sets of nodes of $G^i$ and $G^j$ that have outgoing edges are disjoint, so $G^i\cup G^j$ is again a valid proof graph. By the same reasoning, so is $G^\omega = \bigcup_{i\in\mathbb{N}} G^i$. Associate with every node $v$ of $G$ a distinct number $k(v)$, fixing $k(\langle\mathbb{t}, X, [\hat{a},\hat{b}]\rangle) = 0$, and extend $k$ to nodes of $G^\omega$ by defining for $v$ in $G$ and $v'$ in $G^\omega$ that $k(v') = k(v)$ iff $v'$ is equal to $v$ in which every $a\in A$ is replaced by $\langle a,i\rangle$.

For every $v$ in $G^\omega\cap\mathcal{S}_{\{Z\}}$, we replace every edge $v\to\langle\mathbb{t},V,[\langle a,i\rangle,b]\rangle$ with $V\in\{Y,Z\}$ and $i\neq k(v)$ by $v\to\langle\mathbb{t},V,[\langle a,k(v)\rangle,b]\rangle$. Note that $v$ also has a successor $\langle\mathbb{t},\to,[\langle a',i\rangle,\langle a,i\rangle]\rangle$.

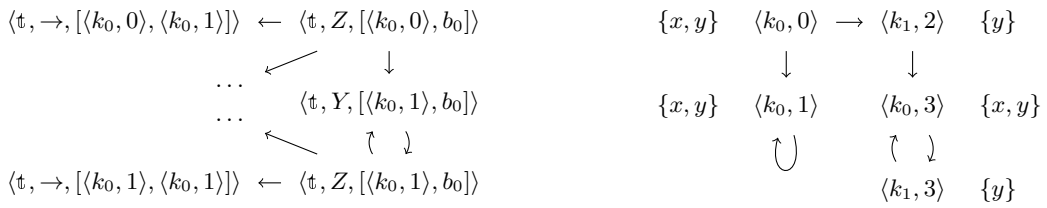
Replace this successor by the node $\langle \mathtt{t}, \to, [\langle a', i\rangle, \langle a, k(v)\rangle]\rangle$. Let $G^{\mathrm{t}}$ be the result of this transformation, restricted to the part that is reachable from $v_0 = \langle \mathtt{t}, X, [\langle \hat{a}, 0\rangle, \langle \hat{b}, 0\rangle]\rangle$. This transformation preserves Property 1:

▶ **Property 2.** *Every SCC in $G^t$ consists of a single cycle.*

$G^{\mathrm{t}}$ is a valid dependency graph again; the restriction to the reachable part from $v_0$ is easily seen to preserve the conditions of Definitions 3 and 5. In the replacements we made, only the first and last conjunct in the right-hand side of the equation for $Z$ are affected by a different choice for $s'$. These two conjuncts are represented by the new successors we introduced, satisfying the constraint from Definition 3.

To see that $G^{\mathrm{t}}$ is also a proof graph for $\mathfrak{A}^\omega \cup \mathfrak{B}_f \models \Phi s f$, notice that no 'bad' cycles were introduced during the transformation: if we view proof graphs as Kripke structures in which two nodes are labelled identically if and only if they differ only in the index of a state in $\mathfrak{A}^\omega$ (*i.e.*, if they are of the form $\langle \mathtt{t}, V, [\langle a, i\rangle, b]\rangle$ and $\langle \mathtt{t}, V, [\langle a, j\rangle, b]\rangle$), then all identically labelled nodes in $G^\omega$ are bisimilar. Moreover, we only replaced edges $v \to u$ by $v \to u'$ such that $u$ and $u'$ are bisimilar.

▶ **Example 16.** Consider the nodes $\langle \mathtt{t}, Z, [\langle k_0, j\rangle, b_0]\rangle$ in $G^\omega$. Note that these have edges to $\langle \mathtt{t}, Y, [\langle k_0, j\rangle, b_0]\rangle$. Let $k(\langle \mathtt{t}, Z, [k_0, b_0]\rangle) = 1$. The transformation then yields edges $\langle \mathtt{t}, Z, [\langle k_0, j\rangle, b_0]\rangle \to \langle \mathtt{t}, Y, [\langle k_0, 1\rangle, b_0]\rangle$. Likewise, the successors $\langle \mathtt{t}, \to, [\langle k_0, j\rangle, \langle k_0, j\rangle]\rangle$ are all replaced with $\langle \mathtt{t}, \to, [\langle k_0, j\rangle, \langle k_0, 1\rangle]\rangle$; see the small snippet of $G^{\mathrm{t}}$ depicted below (left). A witness from $G^{\mathrm{t}}$, as per Definition 11 is depicted below (right); note that it is tree-like.




We now define a witness $\mathfrak{C}$ as defined in Definition 11, *i.e.*, $\mathfrak{C} = \mathsf{ev}(G^{\mathrm{t}}) \cap \mathfrak{A}^\omega$. The following theorem characterises the shape of $\mathfrak{C}$; the theorem essentially follows from a correspondence between the transition relation of $G^{\mathrm{t}}$ and $\mathfrak{C}$.

▶ **Theorem 17** ([6])**.** *Every SCC in $\mathfrak{C}$ consists of a single cycle and $\mathfrak{C}$ is weakly connected.*

If the SCC decomposition of $\mathfrak{C}$ is not a tree, $\mathfrak{C}$ can be transformed to a bisimilar, tree-like structure $\mathfrak{C}^{\mathrm{t}}$ by duplicating SCCs with more than one incoming transition. $\mathfrak{A}$ simulates $\mathfrak{C}^{\mathrm{t}}$ because $\mathfrak{A}$ is bisimilar to $\mathfrak{A}^\omega$, $\mathfrak{C} \sqsubseteq_{\mathrm{w}} \mathfrak{A}^\omega$, and $\mathfrak{C}$ is again bisimilar to $\mathfrak{C}^{\mathrm{t}}$. The fact that $f$ holds on $\mathfrak{C}$ follows from Proposition 2, so we may conclude that $f$ also holds on the bisimilar $\mathfrak{C}^{\mathrm{t}}$.

▶ **Corollary 18.** *$\mathfrak{C}^t$ is a tree-like witness.*

In ∃ELTL model checking there is at most one Büchi automaton in the formula and therefore at most one cycle in $G$. The SCC duplication described above is then unnecessary.

▶ **Corollary 19.** *If $f$ was an ∃ELTL formula, then $\mathfrak{C}$ is a linear witness.*

## 5 Concluding remarks

The diagnostics generation framework presented in this paper is inspired by Tan's attempt at diagnostics generation from *support sets* [23], which was shown to be flawed in [9]. The

diagnostics generation frameworks by Chechik and Gurfinkel [2], and Shoham and Grumberg [22] generate counterexamples and witnesses for CTL, for the purpose of counterexample guided abstraction refinement. The game-based approach of these frameworks is similar to ours, but aimed at a specific application, and in case of [22], at efficient computation. Our framework is more general in comparison, because it defines counterexamples and witnesses for a large variety of model checking problems, and also provides a notion of evidence for the more general setting of least fixpoint logic. Our contribution lies in providing a framework in which diagnostics for a wider variety of problems can be understood in the same way, while focusing less on how such diagnostics are obtained (although suggestions on how to do this are given in [6]).

In order to define our notion of evidence, we have adapted the notion of proof graph from [9] to include constructs that deal with negation. A side effect is that these proof graphs also induce a semantics for non-monotone formulas; for instance, one could assert that the formula $[\mathbf{lfp}\, Xn.\, n = 0 \vee \neg X(n-1)]4$ holds on the first-order structure $\langle \mathbb{Z}, -, 0 \rangle$, because there is a proof graph that witnesses it. It would be interesting to investigate whether this yields a usable semantics, and in particular, how it relates to the fixpoint theorem for non-monotonic functions in [10].

Our notion of evidence projection (see Definition 9) yields that part of a first-order structure that is relevant for the particular proof or refutation of a fixpoint formula captured by a proof graph. In some cases, the evidence projection alone will provide sufficient insight as to why the formula holds or does not hold, but in other cases it may be necessary to combine the evidence projection with additional information from the proof graph. We leave it as future work to further develop a theory of practical diagnostics based on the notions of proof graph and evidence discussed here.

**Acknowledgements.** The authors would like to thank the CSL reviewers for their constructive feedback and useful suggestions.

## References

1 J. Brotherston and A. Simpson. Sequent calculi for induction and infinite descent. *J. Log. Comput.*, 21(6):1177–1216, 2011.

2 M. Chechik and A. Gurfinkel. A framework for counterexample generation and exploration. *STTT*, 9(5-6):429–445, 2007.

3 T. Chen, B. Ploeger, J.C. van de Pol, and T.A.C. Willemse. Equivalence checking for infinite systems using parameterized boolean equation systems. In *CONCUR 2007*, volume 4703 of *LNCS*, pages 120–135. Springer, 2007.

4 E.M. Clarke, S. Jha, Y. Lu, and H. Veith. Tree-like counterexamples in model checking. In *LICS 2002*, pages 19–29. IEEE Computer Society, 2002.

5 R. Cleaveland. On automatically explaining bisimulation inequivalence. In *CAV 1990*, volume 531 of *LNCS*, pages 364–372. Springer, 1991.

6 S. Cranen. *Getting the point – Obtaining and understanding fixpoints in model checking.* PhD thesis, Technische Universiteit Eindhoven, 2015. Available at `http://repository.tue.nl/791780`.

7 S. Cranen, J.F. Groote, J.J.A. Keiren, F.P.M. Stappers, E.P. de Vink, J.W. Wesselink, and T.A.C. Willemse. An overview of the mCRL2 toolset and its recent advances. In *TACAS 2013*, volume 7795 of *LNCS*, pages 199–213. Springer, 2013.

8 S. Cranen, J.F. Groote, and M.A. Reniers. A linear translation from CTL* to the first-order modal $\mu$-calculus. *Theoretical Computer Science*, 412(28):3129–3139, 2011.

**9** S. Cranen, B. Luttik, and T.A.C. Willemse. Proof graphs for parameterised boolean equation systems. In *CONCUR 2013*, volume 8052 of *LNCS*, pages 470–484. Springer, 2013.

**10** Z. Ésik and P. Rondogiannis. A fixed point theorem for non-monotonic functions. *Theoretical Computer Science*, 574:18–38, 2015.

**11** O. Friedmann and M. Lange. The modal $\mu$-calculus caught off guard. In *TABLEAUX 2011*, volume 6793 of *LNCS*, pages 149–163. Springer, 2011.

**12** C. Grabmayer. *Relating Proof Systems for Recursive Types.* PhD thesis, Vrije Univesiteit Amsterdam, 2005.

**13** E. Grädel. Model checking games. *Electr. Notes Theor. Comput. Sci.*, 67:15–34, 2002.

**14** J.F. Groote and R. Mateescu. Verification of temporal properties of processes in a setting with data. In *AMAST 1998*, volume 1548 of *LNCS*, pages 74–90. Springer, 1999.

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

**16** J.F. Groote and T.A.C. Willemse. Parameterised boolean equation systems. *Theoretical Computer Science*, 343(3):332–369, 2005.

**17** D. Janin. Automata, tableaus and a reduction theorem for fixpoint calculi in arbitrary complete lattices. In *LICS 1997*, pages 172–182. IEEE Computer Society, 1997.

**18** H. Korver. Computing distinguishing formulas for branching bisimulation. In *CAV 1991*, volume 575 of *LNCS*, pages 13–23. Springer, 1992.

**19** L. Libkin. *Elements of Finite Model Theory.* Springer, 2004.

**20** Y. Moschovakis. *Elementary induction on abstract structures.* North Holland, 1974.

**21** K.S. Namjoshi. A simple characterization of stuttering bisimulation. In *FSTTCS 1997*, volume 1346 of *LNCS*, pages 284–296. Springer, 1997.

**22** S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. *ACM Trans. Comput. Log.*, 9(1), 2007.

**23** L. Tan. *Evidence-Based Verification.* PhD thesis, Department of Computer Science, State University of New York, 2002.

**24** A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

**25** W. Thomas. Computation tree logic and regular omega-languages. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop 1988*, volume 354 of *LNCS*, pages 690–713. Springer, 1989.