Bisimulation Safe Fixed Point Logic

Faried Abu Zaid [1]

Mathematical Foundations of Computer Science, RWTH Aachen University
D-52056 Aachen

Erich Grädel [2]

Mathematical Foundations of Computer Science, RWTH Aachen University
D-52056 Aachen

Stephan Jaax [3]

Mathematical Foundations of Computer Science, RWTH Aachen University
D-52056 Aachen

Abstract

We define and investigate a new modal fixed-point logic, called bisimulation safe fixed-point logic BSFP, which is a calculus of binary relations that extends both PDL and the modal μ-calculus. The logic is motivated by concepts and results due to van Benthem and Hollenberg on bisimulation safety which plays a similar role for binary relations as the more familiar notion of bisimulation invariance plays for monadic ones. We prove that BSFP is indeed bisimulation invariant for state formulae and bisimulation safe for action formulae. We investigate the expressive power of BSFP and show that it is not limited to monadic second-order definability. Further, we reveal a close relationship of BSFP with context-free languages. We identify a fragment of BSFP that is equivalent to the extension of PDL by context-free grammars. Although BSFP is far more expressive than the modal μ-calculus, its model-checking problem has the same complexity. On the other side, the satisfiability problem for BSFP is highly undecidable.

Keywords: Modal Logic, Dynamic logic, Fixed Point Logic. Bisimulation Invariance, Safety for Bisimulation

1 Introduction

Bisimulation is a fundamental notion for the analysis of modal logics and the behaviour of transition systems. Intuitively, two states v, v' in transition systems

[1] abuzaid@logic.rwth-aachen.de
[2] graedel@logic.rwth-aachen.de
[3] stephan.jaax@rwth-aachen.de

$\mathcal{T}, \mathcal{T}'$ are bisimilar if the set of possible traces from these states are equivalent in a strong sense. Bisimilar states must share the same local properties and any transition from v to w in $\mathcal{T}$ must have a transition of the same kind from v' to w' in $\mathcal{T}'$ (and vice versa) such that w and w' are again bisimilar.

Modal logics are invariant under bisimulation. This means that for any pair of bisimilar nodes $v \in \mathcal{T}$ and $v' \in \mathcal{T}'$, and for any modal formula ψ we have that $\mathcal{T}, v \models \psi$ if, and only if $\mathcal{T}', v' \models \psi$. This *bisimulation invariance* holds not only for the basic propositional modal logic ML, but also for the extensions to stronger logics used in program analysis and verification such as the computation tree logics CTL, CTL*, the propositional dynamic logic PDL, and fixed-point logics such as the modal μ-calculus L_μ and the modal iteration calculus MIC [3]. Since every pointed transition system $\mathcal{T}, v$ can be unraveled from v to a tree $\mathcal{T}^*$ with root v, such that $\mathcal{T}, v \sim \mathcal{T}^*, v$ it follows that every bisimulation-invariant logic has the *tree model property*: every satisfiable formula is true at the root of a tree model. The tree model property is also algorithmically very important since it paves the way to the use of automata-based methods for satisfiability testing.

The relationship between modal logic and bisimulation can in fact be taken an important step further to *model-theoretic characterization theorems*. It is a classical result by van Benthem [12] that modal logic is precisely the bisimulation-invariant fragment of first-order logic. This means that an arbitrary first-order formula $\varphi(x)$ (in a vocabulary of unary and binary relations) is invariant under bisimulation if, and only if, it is equivalent to a formula of ML. An important counterpart of van Benthem's characterization is the Theorem by Janin and Walukiewicz [10] saying that, in precisely the same sense, the modal μ-calculus L_μ is the bisimulation-invariant fragment of monadic second-order logic MSO. For more details, including characterizations theorems for several other variants of bisimulations we refer to the survey [5] and the references there.

In this paper we study a related notion, called *bisimulation safety*, that has been introduced by van Benthem (see [13,9]). To motivate this notion, we have a closer look at the propositional dynamic logic PDL [7]. Recall that PDL is a logic with a two-sorted syntax that distinguishes between state formulae and programs, defined by the mutual induction

$$\varphi ::= P \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\alpha\rangle\varphi$$
$$\alpha ::= E \mid \varphi? \mid \alpha \cup \alpha \mid \alpha;\alpha \mid \alpha^*$$

In a given transition system $\mathcal{T}$ over a set of states V, a state formula φ defines a set of states, $[\![\varphi]\!]^{\mathcal{T}} = \{v : \mathcal{T}, v \models \varphi\} \subseteq V$, whereas a PDL-program α defines a set of transitions, i.e. a binary relation $[\![\alpha]\!]^{\mathcal{T}} \subseteq V \times V$. State formulae and programs are linked in one direction by using programs as modalities to form state formulae $\langle\alpha\rangle\varphi$, saying that there is a transition in α leading to a new state w at which φ holds, and in he other direction by the possibility to form test programs $\varphi?$ defining transitions (v, v) at states where φ holds.

When one says that PDL can be embedded into the modal μ-calculus L_μ and that PDL is bisimulation-invariant, one just considers the state formulae. The

PDL-programs have no direct counterpart in L_μ for the trivial reason that L_μ is a logic of state formulae only and the extension of any L_μ formula is a set of states rather than a set of transitions. Thus, the notion of bisimulation invariance applies to state formulae only, not to programs. However PDL-programs are *bisimulation-safe* in the sense that they do not destroy bisimulations.

Definition 1.1 A binary global relation φ that associates with every transition system $\mathcal{T}$ (of a fixed vocabulary τ) a set of transitions $[\![\varphi]\!]^{\mathcal{T}}$ is *safe for bisimulations* if every bisimulation Z between two transition systems $\mathcal{T}$ and $\mathcal{T}'$ is also a bisimulation between the expansions $(\mathcal{T}, [\![\varphi]\!]^{\mathcal{T}})$ and $(\mathcal{T}', [\![\varphi]\!]^{\mathcal{T}'})$.

Typical bisimulation safe operations are the union and composition of two binary relations whereas intersection and complementation are unsafe for bisimulation. In the same sense as ML is the bisimulation-invariant fragment of first-order logic, van Benthem [13] also proved a similar correspondence between the bisimulation-safe fragment of first-order logic and the class of PDL-programs that do not contain the Kleene star: A first-order formula $\varphi(x, y)$ is bisimulation-safe if, and only if it is equivalent to some $*$-free PDL-program.

This result, together with the Janin-Walukiewicz Theorem raises the following questions.

(1) Can one characterize in a similar way the bisimulation-safe fragment of monadic second-order logic?

(2) Is there an embedding of full PDL (state formulae and programs) into a natural fixed-point logic L that is not only bisimulation-invariant for state formulae but also bisimulation-safe for action formulae?

To the first question, an answer has been given by Marco Hollenberg [9] who considered so-called *μ-programs*. These can be defined by applying the program constructions of PDL not just to state formulae of PDL but to formulae of the modal μ-calculus. As for PDL, one can define μ-formulae and μ-programs by a mutual induction

$$\varphi ::= P \mid X \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\alpha\rangle\varphi \mid \mu X.\varphi$$
$$\alpha ::= E \mid \varphi? \mid \alpha \cup \alpha \mid \alpha;\alpha \mid \alpha^*$$

It is not difficult to see that μ-formulae (defined in this slightly nonstandard way) are bisimuation-invariant, μ-programs are bisimulation-safe and that this definition does not take us outside of monadic second-order logic. In particular, this way of defining μ-formulae is equivalent to the standard definition of the μ-calculus which does not refer to μ-programs at all. The main result of Hollenberg says that μ-programs coincide with the bisimulation-safe fragment of monadic second-order logic [9, Corollary 3.5.5]: An MSO-formula $\varphi(x, y)$ is safe for bisimulations if, and only if, it is equivalent to a μ-program.

It should be noted that the only enrichment of μ-programs with respect to PDL-programs concerns the application of test-instructions $\varphi?$ which now refer to μ-formulae rather that just PDL-formulae. The iteration mechanism

of μ-programs, however, remains limited to the Kleene star; in particular μ-programs do not have a full least (or greatest) fixed-point mechanism for sets of transitions.

In this paper, we shall address the second question and define a modal fixed-point logic for defining sets of transitions, which we call *bisimulation-safe fixed-point logic* BSFP. We shall analyse its expressive power and its model-theoretic and algorithmic properties. In particular we shall prove that BSFP is indeed safe for bisimulations whereas previously known extensions of the modal μ-calculus either remain limited to monadic fixed-points or are not bisimulation-safe. In particular, this is the case for the binary fragment of the least fixed point logic LFP and for the two-dimensional μ-calculus by Otto [11].

We shall provide several presentations of our logic. The first has a minimal syntax as a pure calculus of binary relations, with a projection operator to recover monadic relations. The second presentation is based on a two-sorted syntax, as for PDL and μ-programs, distinguishing between state formulae and action formulae. The equivalence of the two presentations will reveal that BSFP is the generalization of PDL- and μ-programs by admitting full binary fixed point definitions rather than just the Kleene star. We shall see that while this construction remains bisimulation-safe and does not increase the complexity of the model-checking problem, it nevertheless makes the logic much stronger. Contrary to the modal μ-calculus, BSFP admits infinity axioms, is not restricted to MSO-definability and is intimately connected to context-free languages. We shall see that all Boolean combinations of context-free languages are definable in BSFP, and we shall identify a fragment of BSFP that is equivalent to the extension of PDL by context-free grammars. As a consequence, the satisfiability problem for BSFP is highly undecidable.

2 Background from logic

We assume that the reader is familiar with modal logic, first-order logic (FO), monadic second-order logic (MSO), the extension of first-order logic by second-order quantification $\exists X$ and $\forall X$ over *sets* of elements of the structure on which the formula is evaluated. In contrast to second-order logic (SO), where quantification over arbitrary relations (or functions) is admitted, MSO is a much more manageable formalism; it is decidable on many interesting classes of structures (on words and on trees in particular) and amenable to automata-based methods.

We further assume that the reader is familiar with the modal μ-calculus L_μ, briefly described in the introduction of this paper, which extends propositional modal logic ML by least (and greatest) fixed points, and which plays a fundamental role in many areas of logic in computer science, in particular for the specification and verification of computing systems. In finite model theory, descriptive complexity and database theory, other fixed-point logics are of central importance (see [6]). Relevant for the purpose of this paper is the least fixed-point logic LFP which augments the power of first order logic by

least and greatest fixed points of definable relational operators and thus extends FO in a similar way as the μ-calculus extends propositional modal logic. The bisimulation safe fixed point logic BSFP that we are studying in this paper lies between L_μ and LFP. We will briefly recall some basic definitions for LFP here. For a more detailed account, we refer to [6].

Every formula $\psi(R, \overline{x})$, where R is a relation symbol of arity k and $\overline{x}$ is a tuple of k variables, defines, for any structure $\mathfrak{A}$ of appropriate vocabulary, an update operator $F : \mathcal{P}(A^k) \to \mathcal{P}(A^k)$ on the class of k-ary relations over the universe A of $\mathfrak{A}$, namely $F : R \mapsto \{\overline{a} : (\mathfrak{A}, R) \models \psi(R, \overline{a})\}$. If ψ is positive in R, that is, if every occurrence of R falls under an even number of negations, this operator is monotone in the sense that $R \subseteq R'$ implies $F(R) \subseteq F(R')$. It is well known that every monotone operator has a least fixed point and a greatest fixed point, which can be defined as the intersection and union, respectively, of all fixed points, but which can also be constructed by transfinite induction.

LFP is defined by adding to the syntax of first order logic the following *fixed point formation rule:* If $\psi(R, \overline{x})$ is a formula with a relational variable R occurring only positively and a tuple of first-order variables $\overline{x}$, and if $\overline{t}$ is a tuple of terms (such that the lengths of $\overline{x}$ and $\overline{t}$ match the arity of R), then $[\mathbf{lfp}\, R\overline{x}.\psi](\overline{t})$ and $[\mathbf{gfp}\, R\overline{x}.\psi](\overline{t})$ are also formulae, binding the occurrences of the variables R and $\overline{x}$ in ψ.

The semantics of least fixed-point formulae in a structure $\mathfrak{A}$, providing interpretations for all free variables in the formula, is the following: $\mathfrak{A} \models [\mathbf{lfp}\, R\overline{x}.\psi](\overline{t})$ if $\overline{t}^{\mathfrak{A}}$ belongs to the least fixed point of the update operator defined by ψ on $\mathfrak{A}$. Similarly for greatest fixed points.

Note that in formulae $[\mathbf{lfp}\, R\overline{x}.\psi](\overline{t})$ one may allow ψ to have other free variables besides $\overline{x}$.

The duality between least and greatest fixed point implies that for any ψ,

$$[\mathbf{gfp}\, R\overline{x}.\psi](\overline{t}) \equiv \neg[\mathbf{lfp}\, R\overline{x}.\neg\psi[R/\neg R]](\overline{t}).$$

The *width* of an LFP-formula is the maximal number of free variables in its subformulae. Further, an LFP-formula is *parameter-free* if in all its fixed-point expressions $[\mathbf{lfp}\, R\overline{x}\,.\,\varphi(R, \overline{x})](\overline{x})$ and $[\mathbf{gfp}\, R\overline{x}\,.\,\varphi(R, \overline{x})](\overline{x})$ the only free variables occurring in φ are those in $\overline{x}$. It is well-known that every LFP-formula can be translated into an equivalent one that is parameter-free, but this does, in general, increase the arity of the fixed-point variables and the width of the formulae.

Notice that any property of finite structures that is expressible by a fixed LFP-formula can be decided in polynomial time. In fact, on *linearly ordered* finite structures, precisely the polynomial-time decidable properties are LFP-definable, but this is not true in the absence of a linear order (although certain P-complete problems, such as winning regions of reachability games, remain definable in LFP and even in the modal μ-calculus). Indeed, it is a major open problem in finite model theory and descriptive complexity theory whether there exists an extension of LFP that precisely captures the polynomial-time properties of arbitrary (ordered or unordered) finite structures (see [6]).

Evaluation problems in logic, where the formula is not fixed, but part of the input, are more difficult to analyze. The model checking problem for a logic L is the problem to decide, given a formula $\psi \in L$ and a finite structure $\mathcal{K}$ (with elements instantiating the free variables of ψ) whether the formula is true in $\mathcal{K}$. Concerning the complexity of the model-checking problem for LFP and its fragments the following is known (see [6, Chapter 3.3] for details and references).

- For LFP-formulae of unbounded width, model-checking is EXPTIME-complete.
- For LFP-formulae of bounded width that may contain parameters it is PSPACE-complete.
- For parameter-free LFP-formulae of bounded width, as well as for the modal μ-calculus, the model-checking problem is in UP $\cap$ Co-UP and PTIME-hard. It is open whether it is solvable in polynomial time, and this is equivalent to the question whether winning regions of parity games are computable in polynomial time.

3 Bisimulation Safe Fixed Point Logic

In this section, we introduce several presentations of bisimulation safe fixed-point logic BSFP. We shall see that BSFP does not have the finite model property and that it is bisimulation invariant for state formulae and bisimulation safe for action formulae. This will also imply that BSFP is not contained in monadic second-order logic. Finally we will discuss simultaneous fixed points and present a normal form for BSFP.

We start by giving a minimal syntax for BSFP as a pure calculus of binary relations.

Minimal syntax. Let τ be a vocabulary of monadic predicates P_i and binary action predicates E_a, and let $Z_1, Z_2 \ldots$ be a collection of binary predicate variables. Formulae of BSFP in minimal syntax are build by the grammar

$$\alpha ::= \bot \mid P_i? \mid Z_j \mid E_a \mid \alpha \cup \alpha \mid \sim \alpha \mid \alpha \circ \alpha \mid \mu Z_j.\alpha$$

where, for formulae $\mu Z_j.\alpha$, we require that every free occurrence of Z_j in α is in the scope of an even number of $\sim$ symbols.

Semantics. Let $\mathcal{T} = (V, (P_i^{\mathcal{T}})_i, (E_a^{\mathcal{T}})_a)$ be a transition system (which interprets all monadic predicates P_i by $P_i^{\mathcal{T}} \subseteq V$, all transitions relations E_a and all variables Z that occur free in α as subsets of $V \times V$ denoted by $E_a^{\mathcal{T}}$ and $Z^{\mathcal{T}}$, respectively. When it is clear from the context, we will often omit the superscripts in the notation.) The extension $[\![\alpha]\!]^{\mathcal{T}}$ of a formula α in $\mathcal{T}$ is defined inductively by:

- $[\![\bot]\!]^{\mathcal{T}} := \varnothing$.
- $[\![P_i?]\!]^{\mathcal{T}} := \{(v, v) \in V^{\mathcal{T}} \times V^{\mathcal{T}} : v \in {P_i}^{\mathcal{T}}\}$.
- $[\![E_a]\!]^{\mathcal{T}} := {E_a}^{\mathcal{T}}$ for every $a \in \mathrm{ACT}$.

- $[\![\alpha_1 \cup \alpha_2]\!]^{\mathcal{T}} := [\![\alpha_1]\!]^{\mathcal{T}} \cup [\![\alpha_2]\!]^{\mathcal{T}}$.
- $[\![\alpha_1 \circ \alpha_2]\!]^{\mathcal{T}} := \{(u,w) \in V^{\mathcal{T}} \times V^{\mathcal{T}} : \exists v\,(u,v) \in [\![\alpha_1]\!]^{\mathcal{T}} \wedge (v,w) \in [\![\alpha_2]\!]^{\mathcal{T}}\}$.
- $[\![\sim \alpha]\!]^{\mathcal{T}} := \{(v,v) \in V^{\mathcal{T}} \times V^{\mathcal{T}} : \forall v'\,(v,v') \notin [\![\alpha]\!]^{\mathcal{T}}\}$.
- The μ-operator is a binary least-fixed-point operator:
  $[\![\mu Z.\alpha]\!]^{\mathcal{T}} := \bigcap \left\{ R \subseteq V^{\mathcal{T}} \times V^{\mathcal{T}} : [\![\alpha]\!]^{\mathcal{T}[Z:=R]} \subseteq R \right\}$

Some simple but important definable relations are the diagonal $D :=\sim \bot$ and the projection to the first component, denoted $\downarrow \alpha :=\sim\sim \alpha$. By definition $[\![\downarrow \alpha]\!]^{\mathcal{T}} = \{(v,v) \in V^{\mathcal{T}} \times V^{\mathcal{T}} : \exists v'(v,v') \in [\![\alpha]\!]^{\mathcal{T}}\}$.

We next present an extended syntax for BSFP which relates this logic to PDL and μ-programs in the sense that it defines state formulae and action formulae by mutual induction. In fact BSFP can be seen as the extension of PDL by the possibility to form unary and binary fixed points.

Two-sorted syntax. For a set $X_1, X_2, \ldots$ of monadic variables and a set $Z_1 Z_2, \ldots$ of binary variables, the state and action formulae are defined by

$$\varphi ::= P_i \mid X_i \mid \varphi \vee \varphi \mid \neg\varphi \mid \langle\alpha\rangle\varphi \mid \mu X_i.\varphi$$
$$\alpha ::= D \mid \varnothing \mid E_a \mid Z_k \mid \alpha \circ \alpha \mid \alpha \cup \alpha \mid \varphi? \mid \mu Z_j.\alpha$$

Again we require that for fixed-point formulae $\mu X_i.\varphi$ and $\mu Z_j.\alpha$, every free occurrence of X_i or Z_j is the scope of an even number of $\neg$ symbols.

Semantics. For state formulae the extension $[\![\varphi]\!]^{\mathcal{T}}$ is defined in the standard way, as for PDL and μ-programs. For action formulae, the extensions are defined as in the minimal syntax. We use the expression $[\alpha]\varphi$ as shorthand for $\neg\langle\alpha\rangle\neg\varphi$ and $\varphi \wedge \psi$ as a shorthand for $\neg(\neg\varphi \vee \neg\psi)$. As usual we write $\mathcal{T}, v \models \varphi$ to denote that $v \in [\![\varphi]\!]^{\mathcal{T}}$ and $\mathcal{T}, (v,w) \models \alpha$ to denote that $(v,w) \in [\![\alpha]\!]^{\mathcal{T}}$.

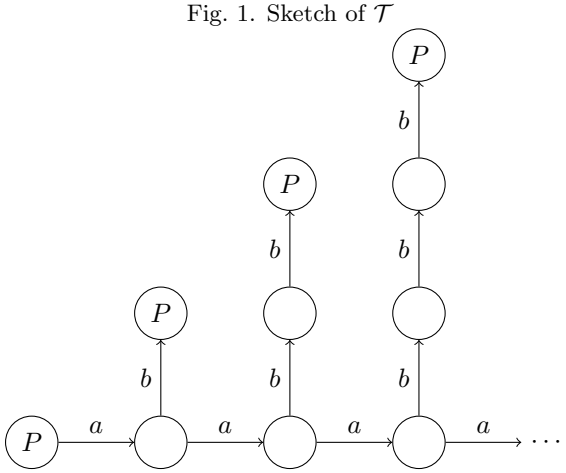
It is not difficult to see that the two presentations of BSFP are equivalent.

Theorem 3.1 *For every* BSFP *state formula φ in two-sorted syntax there is a formula $\hat{\varphi}$ in minimal syntax such that $\mathcal{T}, v \models \varphi \Leftrightarrow \mathcal{T}, (v,v) \models \downarrow \hat{\varphi}$ and for every action formula α there is an equivalent formula $\hat{\alpha}$ in minimal syntax.*

Proof. The translations from φ to $\hat{\varphi}$ and from α to $\hat{\alpha}$ leave the atomic predicates and variables invariant, but monadic variables X_i of φ and α are considered as binary variables in $\hat{\varphi}$ and $\hat{\alpha}$. We then inductively translate the formulae by the following rules:

- if $\varphi = \varphi_1 \vee \varphi_2$, we set $\hat{\varphi} := \hat{\varphi}_1 \cup \hat{\varphi}_2$,
- if $\varphi = \neg\varphi_1$, we set $\hat{\varphi} :=\sim \hat{\varphi}_1$,
- if $\varphi = \langle\alpha_1\rangle\varphi_1$, we set $\hat{\varphi} := \hat{\alpha}_1 \circ \hat{\varphi}_1$,
- if $\varphi = \mu X.\varphi_1$, we set $\hat{\varphi} := \mu X. \downarrow \hat{\varphi}_1$.
- if $\alpha = \alpha_1 \otimes \alpha_2$ with $\otimes \in \{\circ, \cup\}$ simply set $\hat{\alpha} := \hat{\alpha}_1 \otimes \hat{\alpha}_2$,
- if $\alpha = \varphi_1?$ set $\hat{\alpha} :=\downarrow \hat{\varphi}_1$, and

Fig. 1. Sketch of $\mathcal{T}$



- if $\alpha = \mu Z.\alpha_1$ set $\hat{\alpha} := \mu Z.\hat{\alpha}_1$

It is easily verified that this translation gives us a formula with the desired properties in minimal syntax. □

Example 3.2 (i) The action formula $\mu Z.(D \cup (Z \circ E_a))$ defines the set of pairs of states connected by a path of the form a^*.

(ii) The action formula $\mu Z.(D \cup (E_a \circ Z \circ E_b))$ defines the set of pairs of states connected by a path of the form $a^n b^n$ for $n \geq 0$.

We generalize these examples to show that BSFP admits formulae that only have infinite models. We use a construction taken essentially from [8] for $\mathrm{PDL}_{\mathrm{CFG}}$, a logic that is in fact closely related to BSFP (see Sect. 5 below).

Theorem 3.3 BSFP *does not have the finite model property.*

Proof. For BSFP action formulae α, β let $\alpha^* := \mu Z.(D \cup Z \circ \alpha)$ and $\alpha^\Delta \beta^\Delta := \mu Z.(D \cup \alpha \circ Z \circ \beta)$. We claim that the formula

$$\varphi = (P \wedge [E_a^*]\langle E_a \circ E_b^* \rangle P) \wedge [(E_a \cup E_b)^* \circ E_b \circ E_a]\bot \\ \wedge [E_a^* \circ E_a \circ E_a^\Delta E_b^\Delta]\neg P \wedge [E_a^\Delta E_b^\Delta \circ E_b]\bot$$

is satisfiable but has no finite model. Consider the structure

$$\begin{aligned} \mathcal{T} &= (\{w \in \{a,b\}^* \mid w = a^n b^m \text{ with } n \geq m\}, E_a, E_b, P) \text{ with} \\ E_a &= \{(a^n, a^{n+1}) \mid n \geq 0\}, \\ E_b &= \{(a^n b^m, a^n b^{m+1}) \mid n > m\} \text{ and} \\ P &= \{a^n b^n \mid n \geq 0\}. \end{aligned}$$

Obviously $\mathcal{T}$ fulfils all conjuncts of φ from the node ε (c.f. Figure 1), hence $\mathcal{T}, \varepsilon \models \varphi$. Now suppose $\mathcal{T}', v \models \varphi$ for some finite transition system $\mathcal{T}'$ over

the signature $\{P, E_a, E_b\}$. We can interpret $\mathcal{T}'$ as a finite automaton with initial state v and accepting states P. The regular language L accepted by this automaton is determined by the labels of the paths connecting v to a state in P. Therefore, the second conjunct enforces that $L \subseteq a^*b^*$ and the third and fourth conjunct enforce that $L \subseteq \{a^n b^n \mid n \geq 0\}$. Given the other parts of the formula, the first conjunct enforces that for every $n \geq 0$ $a^n b^n \in L$. Thus $L = \{a^n b^n \mid n \geq 0\}$ which is not regular. A contradiction. □

Corollary 3.4 BSFP *is strictly more expressive than the modal μ-calculus.*

We are now ready to show that BSFP has the desired properties with respect to bisimulation.

Theorem 3.5 *State formulae of* BSFP *are bisimulation invariant, and action formulae of* BSFP *are safe for bisimulation.*

Proof. We have to prove that, for every bisimulation S between two transition systems $\mathcal{T}$ and $\mathcal{T}'$ with $(v, v') \in S$, it holds that

(1) v and v' satisfy the same BSFP state formulae, and

(2) whenever $(v, w) \in [\![\alpha]\!]^{\mathcal{T}}$ for an action formula α, then there exists a w' such that $(w, w') \in S$ and $(v', w') \in [\![\alpha]\!]^{\mathcal{T}'}$.

By Theorem 3.1 it suffices to establish (2) for formulae in minimal syntax. Claim (1) then also follows. Indeed, suppose that there is BSFP state formula φ such that $\mathcal{T}, v \models \varphi$ but $\mathcal{T}', v' \not\models \varphi$. By Theorem 3.1 there is a formula $\hat{\varphi}$ such that $\mathcal{T}, (v, v) \models\!\downarrow \hat{\varphi}$ but $\mathcal{T}', (v', v') \not\models\!\downarrow \hat{\varphi}$. But then $\downarrow \hat{\varphi}$ would be unsafe for bisimulation.

Apart from the least fixed-point operator μ, every BSFP-operator has an analogous counterpart in PDL and PDL-operators are known to be safe for bisimulation. It thus suffices to show that if α is safe for bisimulation, then so is $\mu Z.\alpha$. But this follows by a straightforward induction over the stages α^η of the least fixed point induction defined by α. Indeed, for all ordinals η and all transition systems $\mathcal{T}, \mathcal{T}'$ it holds holds that if $(v, w) \in [\![\alpha^\eta]\!]^{\mathcal{T}}$, then there exists a state $w' \in V^{\mathcal{T}'}$ such that $(v', w') \in S$ and $(w, w') \in [\![\alpha^\eta]\!]^{\mathcal{T}'}$.

Zero case: For $\eta = 0$ the claim is trivial.

Successor case: Let $(v, w) \in [\![\varphi^{\eta+1}]\!]^{\mathcal{T}}$. Hence, by definition, we have that $(v, w) \in [\![\varphi]\!]^{\mathcal{T}[X:=[\![\varphi^\eta]\!]^{\mathcal{T}}]}$. Applying the induction hypothesis, we obtain that there exists a $w' \in V^{\mathcal{T}}$ with $(v', w') \in S$ and $(w, w') \in [\![\varphi]\!]^{\mathcal{T}'[X:=[\![\varphi^\eta]\!]^{\mathcal{T}'}]}$ which is, by definition, equivalent to $(w, w') \in [\![\varphi^{\eta+1}]\!]^{\mathcal{T}'}$.

Limit case: Let λ be a limit ordinal such that $(v, v') \in [\![\varphi^\lambda]\!]^{\mathcal{T}}$. Thus we have that $(v, v') \in [\![\varphi^\eta]\!]^{\mathcal{T}}$ for some $\eta < \lambda$. Applying the induction hypothesis, we obtain that there exists a $w' \in V^{\mathcal{T}'}$ with $(v', w') \in S$ and $(w, w') \in [\![\varphi^\eta]\!]^{\mathcal{T}}$ and thus, $(w, w') \in [\![\varphi^\lambda]\!]^{\mathcal{T}'}$. □

Corollary 3.6 BSFP *is not a fragment of* MSO.

Proof. Consider the infinity axiom in BSFP presented in the proof of Theorem 3.3. If it were equivalent to an MSO-formula it would, being bisimulation invariant, also be equivalent to a formula in L_μ. But this is impossible since L_μ has the finite model property. □

Simultaneous Fixed Points. As for the μ-calculus and other fixed-point logics one can generalize also BSFP to admit systems of simultaneous fixed points. These do not increase the expressive power but sometimes allow for more straightforward formalisations. Here one associates with any tuple $\overline{\psi} = (\psi_1, \ldots, \psi_k)$ of formulae $\psi_i(\overline{X}) = \psi_i(X_1, \ldots, X_k)$, in which all occurrences of all X_i are positive, a new formula $\varphi = \mu\overline{X} \,.\, \overline{\psi}$. The semantics of φ is induced by the least fixed point of the monotone operator $\psi^{\mathcal{T}}$ mapping $\overline{X}$ to $\overline{X}'$ where $X_i' = [\![\psi_i]\!]^{(\mathcal{T},\overline{X})}$. More precisely, $\mathcal{K}, (v, w) \models \varphi$ iff (v, w) is an element of the first component of the least fixed point of the above operator. It is known that simultaneous least fixed points can be eliminated in favour of nested individual fixed points by the so-called Békic principle (see e.g. [1, page 27]). Indeed, $\mu XY \,.\, [\psi(X, Y), \varphi(X, Y)]$ is equivalent to $\mu X.\psi(X, \mu Y.\varphi(X, Y))$, and this equivalence generalizes to larger systems in the obvious way.

On this basis, we now introduce a normal form for BSFP action formulae which will be helpful when we investigate the expressive power of BSFP.

Definition 3.7 A action formula α is in normal form if $\alpha = \mu\overline{Z}(\alpha_{Z_1}, \ldots, \alpha_{Z_k})$ with $\alpha_{Z_\ell} = \bigcup_i \beta_i$ and $\beta_i = \gamma_{i1} \circ \cdots \circ \gamma_{in_i}$ where γ_{ij} is either a binary predicate (or binary predicate variable) or $\varphi?$.

Lemma 3.8 *For every* BSFP *action formula α there is an equivalent formula $\hat{\alpha}$ in normal form.*

Proof. Let α be a action formula. We obtain $\hat{\alpha}$ by the following procedure: By applying the Békic principle we get that $\alpha \equiv \mu\overline{Z}(\alpha_{Z_1}, \ldots, \alpha_{Z_k})$ where the α_{Z_i} are $\cup, \circ$ combinations formed from binary predicate symbols (and variables) and tests. Such an α_{Z_ℓ} can be transformed into the form $\bigcup_i(\beta_{i1} \circ \cdots \circ \beta_{in_i})$ by the equivalences $\alpha \circ (\beta \cup \gamma) \equiv \alpha \circ \beta \cup \alpha \circ \gamma$ and $(\alpha \cup \beta) \circ \gamma \equiv \alpha \circ \gamma \cup \beta \circ \gamma$. □

4 Relationship with other fixed-point logics and model-checking

Clearly bisimulation safe fixed-point logic BSFP extends the modal μ-calculus L_μ and can be embedded into the least fixed-point logic LFP, in short $L_\mu \leq$ BSFP $\leq$ LFP. Hence every property expressible in BSFP can be checked in polynomial time, and there exist P-complete properties that are definable in BSFP. Modal fixed-point logics with a similar status are the k-dimensional μ-calculi L_μ^k by Martin Otto [11], for any $k \geq 1$. We investigate the relationship with these other fixed-point logics more closely.

It is known that formulae of the μ-calculus can be translated into parameter-free LFP-formulae of width two. We observe that there is similar embedding of BSFP into LFP which, however, produces formulae of width three.

Proposition 4.1 *There is a linear translation mapping every* BSFP*-formula φ to an equivalent LFP-formula $\varphi^{\#}(x, y)$ which is parameter-free and of width at most three.*

The translation is straightforward; it maps P_i to $x = y \wedge P_i y$ and $\sim \varphi$ to $x = y \wedge \neg\exists y : \varphi^{\#}(x, y)$, translates $\vee$ and least fixed-points literally, and only needs to introduce a third variable for expressing composition: $(\varphi \circ \psi)^{\#}(x, y) := \exists z(\varphi^{\#}(x, z) \wedge \psi^{\#}(z, y))$

Corollary 4.2 *The model-checking problem for* BSFP *is in UP ∩ Co-UP and* PTIME*-hard. It is polynomial time equivalent to the model-checking problem of the modal μ-calculus.*

We next consider the relationship of BSFP with the logic L^2_μ from [11] which is also a modal fixed-point calculus of binary relations. On transition systems $\mathcal{T}$ with universe V and a vocabulary of monadic relations P_i and, for simplicity, just one binary relation E, the k-dimensional μ-calculus L^k_μ is defined by taking the usual μ-calculus L_μ on an expanded system $\mathcal{T}^k$ with universe V^k monadic relations P_{ij} and binary relations E_j, for $j = 1, \ldots, k$ and additional binary relations E_σ, for every substitution $\sigma : \{1, \ldots, k\} \to \{1, \ldots, k\}$. The relations P_{ij} and E_j on V^k are given by P_i and E on the jth component and the relations E_σ contain the transitions from $(v_1, \ldots, v_k)$ to $(v_{\sigma(1)}, \ldots, v_{\sigma(k)})$. The meaning of an L^k_μ-formula ψ on $\mathcal{T}$ is given as the k-ary relation of all tuples $\overline{v}$ such that $\mathcal{T}^k, \overline{v} \models \psi$ (in the sense of L_μ).

A typical relation expressible in L^2_μ is bisimilarity. Two nodes v_1, v_2 are bisimilar in $\mathcal{T}$ if $\mathcal{T}, v_1, v_2 \models \nu Z.(\bigwedge_i (P_{i1} \leftrightarrow P_{i2}) \wedge [1]\langle 2\rangle Z \wedge [2]\langle 1\rangle Z)$.

Martin Otto proved that the multi-dimensional μ-calculus $L^\omega_\mu = \bigcup_{k\in\omega} L^k_\mu$ captures precisely the bisimulation-invariant fragment of polynomial time. Given that BSFP and L^2_μ both are fixed-point calculi that extend the modal μ-calculus to binary relations while respecting bisimulation in some sense, the question arises of how the expressive power of L^2_μ and BSFP compare. A closer look reveals that the two logics respect bisimulations in a rather different sense. First of all we observe that L^2_μ is closed under all Boolean operations and can therefore not be bisimulation safe. For instance, the formula that defines bisimilarity of two nodes in a given transition system is clearly not safe for bisimulation. On the other side L^2_μ is *component-wise invariant under bisimulations*: For any two pairs $v, w \in \mathcal{T}$ and $v', w' \in \mathcal{T}'$ such that v and v' but also w and w' are bisimilar, and any formula $\psi \in L^2_\mu$ it follows that $\mathcal{T}, v, w \models \psi$ if, and only if $\mathcal{T}', v', w' \models \psi$ (see [11]). However, there are quite simple BSFP-formulae, such as for instance the diagonal D, that violate this component-wise bisimulation invariance.

Proposition 4.3 *Concerning expressive power, the two logics* BSFP *and L^2_μ are incomparable.*

5 Flat BSFP

In this section we define the flat fragment of BSFP and show that it is equivalent to $\mathrm{PDL}_{\mathrm{CFG}}$, the extension of PDL by context-free grammars. We first recall the definition of $\mathrm{PDL}_{\mathrm{CFG}}$ from [8] which extends the definition of PDL by a more powerful construction for programs: The set of programs of $\mathrm{PDL}_{\mathrm{CFG}}$ consists of all context-free grammars α whose terminals are atomic actions and test formulae. Such a grammar defines a language $L(\alpha) \subseteq (A \cup \{\varphi_1?, \ldots, \varphi_n?\})^*$. The binary relation, defined by α on a transition system $\mathcal{T}$ is the set of pairs (u, v) such that there is path from u to v in $\mathcal{T}$ (expanded by $[\![\varphi_1?]\!]^{\mathcal{T}}, \ldots, [\![\varphi_n?]\!]^{\mathcal{T}}$) labelled by a word in $L(\alpha)$. $\mathrm{PDL}_{\mathrm{CFG}}$ is known to be much more powerful than PDL. For details, see [8]. We next define Flat BSFP.

Definition 5.1 Flat BSFP is the fragment of all BSFP formulae formed by the following rules:

$$\varphi ::= P_i \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle\alpha\rangle\varphi$$
$$\alpha ::= E_a \mid Z_i \mid \alpha \circ \alpha \mid \alpha \cup \alpha \mid \varphi? \mid \mu Z_i.\alpha$$

For tests $\varphi?$ we additionally demand that φ is closed, which means φ does not contain any free variables.

Theorem 5.2 Flat BSFP $\equiv \mathrm{PDL}_{\mathrm{CFG}}$

Proof. Since the building rules for state formulae of flat BSFP and $\mathrm{PDL}_{\mathrm{CFG}}$ coincide, we only need to show how to translate action formulae. First we show how to translate $\mathrm{PDL}_{\mathrm{CFG}}$ programs into BSFP action formulae. We need to show that for every set $\{\varphi_1, \ldots, \varphi_n\} \subseteq \mathrm{PDL}_{\mathrm{CFG}}$ and every context-free language L over the alphabet $A \cup \{\varphi_1?, \ldots, \varphi_n?\}$ the global relation defined by L is definable by a flat BSFP action formula. Let G be a context-free grammar with non-terminals $\Gamma = \{Z_1, \ldots, Z_n\}$, terminals $\Sigma = A \cup \{\varphi_1?, \ldots, \varphi_n?\}$ and start-symbol Z_1. We may assume that we have already constructed BSFP formulae $\hat{\varphi}_1, \ldots, \hat{\varphi}_n$ such that $\hat{\varphi}_i$ is equivalent to φ_i. With every string $s = s_0 s_1 \cdots s_j \in (\Sigma \cup \Gamma)^*$ we associate the BSFP-formula $\alpha(s) := \alpha(s_0) \circ \alpha(s_1) \circ \ldots \circ \alpha(s_j)$ where $\alpha(s_k) = s_k$, if $s_k \in \Gamma$, $\alpha(s_k) = E_{s_k}$ for $s_k \in A$ and $\alpha(s_k) = \hat{\varphi}_i?$ for $s_k = \varphi_i?$. For instance $\alpha(aZ\varphi_1?Z) = E_a \circ Z \circ \hat{\varphi}_1? \circ Z$. Furthermore, with every production-rule

$$Z_i \longrightarrow A_1 | A_2 | \ldots | A_k \text{ with } A_j \in (\Sigma \cup \Gamma)^*$$

we associate a BSFP action formula $\alpha_{Z_i} := \bigcup_{1 \le j \le k} \alpha(A_j)$, and claim that $\alpha = \mu\overline{Z}.(\alpha_{Z_1}, \ldots, \alpha_{Z_n})$ defines the same relation as G. To see this one recalls that $L(G)$ is the simultaneous least fixed-point, projected on the start symbol Z_1 of the system of equations defined by the production rules of G [2]. For a given transition system $\mathcal{T}$ let $(Z_1^\eta, \ldots, Z_n^\eta)$ denote the η-fold approximation of the least fixed point of the formulae $(\alpha_{Z_1}, \ldots, \alpha_{Z_n})$ over $\mathcal{T}$ and $(X_1^\eta, \ldots, X_n^\eta)$ be the η-fold approximation of the fixed point of the operator associated with the grammar G. One can show via induction that for all ordinals η we have $(u, v) \in Z_\ell^\eta$ if, and only if there is a path from u to v labelled by a word in X_ℓ^η.

For the other direction, it suffices to find for every flat action formula α a context-free grammar G_α over atomic actions and $\mathrm{PDL}_{\mathrm{CFG}}$ tests that defines the same relation as α. By Lemma 3.8 we have that $\alpha \equiv \mu\overline{Z}(\alpha_{Z_1}, \ldots, \alpha_{Z_k})$ with $\alpha_{Z_\ell} = \bigcup_i \beta_i$. Every β_i is a composition of atomic formulae and tests $\beta_i = \beta_{i1} \circ \cdots \circ \beta_{in_i}$. We assign to each such composition a word $w(\beta_i) = w(\beta_{i1})w(\beta_{i2}) \ldots w(\beta_{in_i})$. Here we set $w(E_a) = a, w(Z_i) = Z_i$ and $w(\vartheta?) = \hat{\vartheta}?$ where $\hat{\vartheta}$ is an $\mathrm{PDL}_{\mathrm{CFG}}$ formula equivalent to ϑ (which exists by the induction hypothesis). For every $\alpha_{Z_\ell} = \bigcup_{1 \leq i \leq n_\ell} \beta_i$ we add the production rule

$$Z_\ell \longrightarrow w(\beta_1) | \ldots | w(\beta_{n_\ell}).$$

Again by an induction over the stages of the fixed point iteration one shows that for every transition system $\mathcal{T}$ and every pair of nodes (u, v) we have that $\mathcal{T}, (u, v) \models \alpha$ if, and only if, there is a path from u to v labelled by a word in $L(G_\alpha)$. □

We therefore know that $\mathrm{PDL}_{\mathrm{CFG}} \leq \mathrm{BSFP}$ and even $\mathrm{PDL}_{\mathrm{CFG}} \lneq \mathrm{BSFP}$ since it is known that $\mathrm{PDL}_{\mathrm{CFG}}$ is incomparable to the modal μ-calculus, which is a fragment of BSFP. The satisfiability problem for $\mathrm{PDL}_{\mathrm{CFG}}$ is known to be Σ^1_1-complete [8].

Corollary 5.3 *The satisfiability problem for* BSFP *is* Σ^1_1*-hard.*

The precise complexity level of Sat(BSFP) remains open. It is known that the satisfiability problem for LFP is in the stronger class Σ^2_1 [4], and we do not know whether satisfiablity for BSFP is as hard as for LFP, or Σ^1_1-complete.

We quickly turn our attention to another fragment of BSFP. A BSFP action formula is test-free if it does not contain any test $\varphi?$. Obviously every test-free action formula is flat. An inspection of the proof of Theorem 5.2 reveals that every test-free action formula can be translated into a context-free grammar over terminals in A (i.e. without tests) and vice versa. Hence we obtain the following result for test-free action formulae.

Corollary 5.4 *A global binary relation R is definable by a test-free* BSFP *action formula if, and only if there is a context-free language $L \subseteq A^*$ such that for all transition systems $\mathcal{T}$ it holds that $(u, v) \in R^{\mathcal{T}}$ iff there is a path from u to v that is labelled by some word in L.*

6 Definability of Languages

An important aspect of the expressive power of a logic is the question which classes of languages it can define. There are several possibilities to model the specification of a language by a BSFP formula. The standard way is to identify words with certain structures and associate with a BSFP-formula the language of all words such that the corresponding word structure is a model of the formula. With a finite word $w = w_0 w_2 \ldots w_{n-1} \in \Sigma^n$ of length $n \geq 0$ we may associate the (unlabelled) transition system $\mathcal{T}(w) := (\{0, \ldots, n\}, (E_a)_{a \in \Sigma})$ where $(i, j) \in E_a$ iff $j = i + 1$ and $w_i = a$. A BSFP state formula φ then defines the language

$L(\varphi) := \{w \in \Sigma^* : \mathcal{T}(w), 0 \models \varphi\}$. It is also possible to specify a language by an action formula α, by defining $L(\alpha) := \{w \in \Sigma^* : \mathcal{T}(w), (0, |w|) \models \alpha\}$. It is not hard to see that these two definitions capture the same class of languages.

Lemma 6.1 *A language $L \subseteq \Sigma^*$ is definable by a* BSFP *state formula iff it is definable by a* BSFP *action formula.*

Proof. Let φ be a BSFP formula over the signature $\tau = \{E_a \mid a \in \Sigma\}$. Consider the action formula $\alpha = \mu Z.(\varphi? \cup Z \circ E)$ where E is shorthand for $\bigcup_{a\in\Sigma} E_a$. Obviously for a word w we have

$$[\![\alpha]\!]^{\mathcal{T}(w)} = \{(i, j) \mid 0 \leq i \leq j \leq |w| \text{ and } \mathcal{T}(w), i \models \varphi\}$$

and therefore $\mathcal{T}(w), (0, |w|) \models \alpha$ iff $\mathcal{T}(w), 0 \models \varphi$.

Now consider an action formula α over τ. For $\varphi = \langle\alpha\rangle[E]\bot$ we have by definition that $\mathcal{T}(w), 0 \models \varphi$ iff there is a j such that $(0, j) \in [\![\alpha]\!]^{\mathcal{T}(w)}$ and j has no successor in $\mathcal{T}(w)$ which means $j = |w|$. □

While the modal μ-calculus and PDL have on words the same expressive power as MSO and therefore capture exactly the regular languages, from Corollary 5.4 we know that even the test-free BSFP formulae capture a much richer class of languages.

Corollary 6.2 *A language $L \subseteq \Sigma^*$ is context-free if, and only if, it is definable by a test-free* BSFP *action formula. As a consequence, every Boolean combination of context-free languages is* BSFP*-definable.*

Example 6.3 For the context-free languages

$$L_1 = \{a^n b^n c^m \mid n, m \in \mathbb{N}\} \text{ and } L_2 = \{a^m b^n c^n \mid n, m \in \mathbb{N}\}$$

let $\varphi_{L_1}, \varphi_{L_2}$ be BSFP formulae that define the respective languages. Then $\varphi_{L_1} \wedge \varphi_{L_2}$ defines the context-sensitive language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.

We remark that the way we associated structures with words differs slightly from the way this is usually done when one proves that MSO or L_μ capture exactly the regular languages. There one associates with every word w the structure $\mathcal{T}'(w) = (\{0, \ldots, |w| - 1\}, (P_a)_{a\in\Sigma}, E)$ with $P_a = \{i \mid w_i = a\}$ and $E = \{(i, i + 1) \mid 0 \leq i \leq |w| - 2\}$. However, it is not hard to see that on this class of structures BSFP is not less expressive.

Lemma 6.4 *For every* BSFP *action formula α there is a formula $\hat{\alpha}$ such that*

$$\mathcal{T}(w), (0, |w|) \models \alpha \Leftrightarrow \mathcal{T}'(w), (0, |w|) \models \hat{\alpha}.$$

Proof. From α construct $\hat{\alpha}$ in the following way: first replace every sub-formula E_a by $P_a?$ and then replace every formula of the form $\alpha_1 \circ \alpha_2$ by $\alpha_1 \circ E \circ \alpha_2$.□

A different approach for defining a language with BSFP is to consider the structure $\mathcal{T}_{\Sigma^*} = (\Sigma^*, (E_a)_{a\in\Sigma})$ with $E_a^{\mathcal{T}_{\Sigma^*}} = \{(w, wa) \mid w \in \Sigma^*\}$. We say a BSFP formula α defines a language L in $\mathcal{T}_{\Sigma^*}$ if $L = \{w \in \Sigma^* \mid \mathcal{T}_{\Sigma^*}, (\varepsilon, w) \models \alpha\}$

Theorem 6.5 *A language L is* BSFP *definable in $\mathcal{T}_{\Sigma^*}$ if, and only if, it is context-free.*

Proof. We claim that for every action formula α there is a test-free action formula $\hat{\alpha}$ such that $[\![\alpha]\!]^{\mathcal{T}_{\Sigma^*}} = [\![\hat{\alpha}]\!]^{\mathcal{T}_{\Sigma^*}}$. Since BSFP state formulae are invariant under bisimulation and every pair of nodes in $\mathcal{T}_{\Sigma^*}$ is bisimilar, every test $\varphi?$ holds either for every node or for no node at all. We can therefore replace every test in α either by D or by $\varnothing$ and arrive at a test-free BSFP formula with the same extension in $\mathcal{T}_{\Sigma^*}$. With Corollary 5.4 we get that $\hat{\alpha}$ corresponds to some context-free language L and therefore $\{w \in \Sigma^* \mid \mathcal{T}_{\Sigma^*}, (\varepsilon, w) \models \alpha\} = L$. □

References

[1] Arnold, A. and D. Niwiński, "Rudiments of μ-calculus," North Holland, 2001.

[2] Bertoni, A., C. Choffrut and R. Radicioni, *The inclusion problem of context-free languages: Some tractable cases*, in: *Developments in language theory*, Springer, 2009, pp. 103–112.

[3] Dawar, A., E. Grädel and S. Kreutzer, *Inflationary fixed points in modal logic*, ACM Transactions on Computational Logic **5** (2004), pp. 282 – 315.

[4] Dawar, A. and Y. Gurevich, *Fixed point logics*, Bulletin of Symbolic Logic **8** (2002), pp. 65–88.

[5] Grädel, E. and M.Otto, *The freedoms of (guarded) bisimulation*, in: *Johan F.A. K.van Benthem on Logical and Informational Dynamics*, Springer, 2014 p. To appear.

[6] Grädel, E. et al., "Finite Model Theory and Its Applications," Springer, 2007.

[7] Harel, D., D. Kozen and J. Tiuryn, "Dynamic Logic," MIT Press, 2000.

[8] Harel, D., A. Pnueli and J. Stavi, *Propositional dynamic logic of nonregular programs*, Journal of Computer and System Sciences **26** (1983), pp. 222–243.

[9] Hollenberg, M., "Logic and Bisimulation," Ph.D. thesis, Utrecht University (1998).

[10] Janin, D. and I. Walukiewicz, *On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic*, in: *Proceedings of 7th International Conference on Concurrency Theory CONCUR '96*, number 1119 in Lecture Notes in Computer Science (1996), pp. 263–277.

[11] Otto, M., *Bisimulation-invariant Ptime and higher-dimensional mu-calculus*, Theoretical Computer Science **224** (1999), pp. 237–265.

[12] van Benthem, J., "Modal Correspondence Theory," Ph.D. thesis, University of Amsterdam (1976).

[13] van Benthem, J., *Program constructions that are safe for bisimulation*, Studia Logica **60** (1998), pp. 311–330.