Model Checking as Static Analysis: Revisited

Fuyuan Zhang, Flemming Nielson, and Hanne Riis Nielson

DTU Informatics, Technical University of Denmark, DK-2800 Lyngby, Denmark
{fuzh,nielson,riis}@imm.dtu.dk

Abstract. We show that the model checking problem of the μ-calculus can be viewed as an instance of static analysis. We propose Succinct Fixed Point Logic (SFP) within our logical approach to static analysis as an extension of Alternation-free Least Fixed Logic (ALFP). We generalize the notion of stratification to weak stratification and establish a Moore Family result for the new logic as well. The semantics of the μ-calculus is encoded as the intended model of weakly stratified clause sequences in SFP.

1 Introduction

Both *model checking* [1, 5] and *static analysis* [7] are prominent approaches to detecting software errors. Model Checking is a successful formal method for verifying properties specified in modal logics with respect to transition systems. Static analysis is also a powerful method for validating program properties which can predict safe approximations to program behaviors.

The link between model checking and static analysis has been studied for many years. Recent research [13] takes the point of view that model checking problems can be reduced to static analysis and presents a flow logic approach to static analysis which encodes the model checking problem of *Action Computation Tree Logic* [14] in *Alternation-free Least Fixed Point Logic* (ALFP [15]). It is shown in [21] that model checking for the alternation-free μ-calculus can be encoded in ALFP as well. However, as is suggested in the negative result there, ALFP is not well-suited for the encoding of the full fragment of the μ-calculus, where nesting of the least and greatest fixed points are allowed.

Continuing these lines of work, we propose *Succinct Fixed Point Logic* (SFP) as an extension of ALFP within the framework of our logical approach to static analysis and show that the model checking problem of the μ-calculus [1, 6] can be encoded in SFP. We first propose the notion of *weak stratification* which allows a convenient specification of nested fixed points in the μ-calculus. Then, we give the definition of the *intended model* of SFP clause sequences. Unlike in ALFP, we explicitly introduce a least fixed point operator in SFP to facilitate our development. Last, we explain our approach to the analysis of the μ-calculus and show that the intended model of an SFP clause sequence specifying a μ-calculus formula exactly characterizes the set of states which satisfy this μ-calculus formula over a given Kripke structure.

J. Derrick et al. (Eds.): IFM 2012, LNCS 7321, pp. 99–112, 2012.

The structure of this paper is as follows. In Section 2, we briefly introduce Kripke structure and the syntax and semantics of the μ-calculus. Section 3 explains our logical approach to static analysis, where we first review ALFP and then propose SFP, which is a main contribution of this paper. We show through an example that we cannot take the greatest lower bound of the set of models of an SFP clause sequence as the intended model, since this does not match the fixed point semantics of the μ-calculus. Section 4 is the other main contribution of our work, where we encode the model checking problem of the μ-calculus in SFP. We conclude our work in Section 5.

2 Modal μ-Calculus

2.1 Kripke Structures

The definition of *Kripke Structure* is modified slightly in comparison with [1] to distinguish different transitions in a system. Here, a Kripke structure over a set $\boldsymbol{P}$ of atomic propositions is a tuple $M = (S, T, L)$, where S is a set of states, T is a set of transition relations, and $L : S \to 2^{\boldsymbol{P}}$ labels each state with the set of true atomic propositions. Each element a in T is a transition relation and $a \subseteq S \times S$. As in [1] we also assume that the Kripke structure is total, although this is not necessary for our development.

2.2 Syntax and Semantics of the Modal μ-Calculus

Definition 1 (Syntax of the Modal μ-calculus). *Let Var be a set of variables, and $\boldsymbol{P}$ be a set of atomic propositions. The syntax of the modal μ-calculus is defined as follows:*

$$\phi ::= \ p \mid Q \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle \phi \mid [a]\phi \mid \mu Q.\phi \mid \nu Q.\phi$$

Here $p \in \mathbf{P}$, $Q \in Var$ and $a \in T$. The μ (resp. ν) operator is the least (resp. greatest) fixed point operator. For $\mu Q.\phi$ and $\nu Q.\phi$, it is required that all occurrences of Q in ϕ are under *an even number of negations* within ϕ. In this case, ϕ is said to be *syntactically monotone* in Q. A variable is *free* if it is not bound by any fixed point operator in a formula. A formula is *closed* if there are no free variables in it.

A formula ϕ is interpreted as the set of states, on a given Kripke structure, that make it true and this set of states is denoted by $[\![\phi]\!]_e$, where $e : Var \to 2^S$ is an environment. We use $e[Q \mapsto S]$ to denote the new environment updated from e by binding the relational variable Q to the set of states S. The semantics of μ-calculus formulas are defined as follows.

- $[\![p]\!]_e = \{ \ s \mid p \in L(s) \ \}$
- $[\![Q]\!]_e = e(Q)$
- $[\![\neg\phi]\!]_e = S \setminus [\![\phi]\!]_e$
- $[\![\phi_1 \vee \phi_2]\!]_e = [\![\phi_1]\!]_e \cup [\![\phi_2]\!]_e$

- $[\![\phi_1 \wedge \phi_2]\!]_e = [\![\phi_1]\!]_e \cap [\![\phi_2]\!]_e$
- $[\![\langle a \rangle \phi]\!]_e = \{ \ s \mid \exists s' : \ (s, \ s') \in a \ \texttt{and} \ s' \in [\![\phi]\!]_e \}$
- $[\![[a]\phi]\!]_e = \{ \ s \mid \forall s' : \ (s, \ s') \in a \ \texttt{implies} \ s' \in [\![\phi]\!]_e \}$
- $[\![\mu Q.\phi]\!]_e$ is the least fixpoint of the function $\tau(S) = [\![\phi]\!]_{e[Q \mapsto S]}$
- $[\![\nu Q.\phi]\!]_e$ is the greatest fixpoint of the function $\tau(S) = [\![\phi]\!]_{e[Q \mapsto S]}$

The boolean operators have the usual meanings. If $(s, \ s') \in a$, we call s' an a-derivative of s. Due to the restricted use of negations in ϕ, monotonicity is guaranteed [1] for the function $\tau(S) = [\![\phi]\!]_{e[Q \mapsto S]}$. The dualities $\neg[a]\phi \equiv \langle a \rangle \neg\phi$, $\neg\langle a \rangle \phi \equiv [a]\neg\phi$, $\neg\mu Q.\phi \equiv \nu Q.\neg\phi[\neg Q/Q]$, and $\neg\nu Q.\phi \equiv \mu Q.\neg\phi[\neg Q/Q]$ are useful when transforming a formula to an equivalent form according to the semantics of the μ-calculus. The notation $\phi[\neg Q/Q]$ refers to a formula resulting from ϕ by substituting all occurrences of Q in ϕ with $\neg Q$. We give another syntax of the μ-calculus using only the μ operator as follows, which will facilitate our static analysis approach to the analysis of the μ-calculus.

Definition 2. *Let Var be a set of variables, $\boldsymbol{P}$ be a set of atomic propositions that is closed under negation. The syntax of the μ-calculus is defined as follows:*

$$\phi ::= \ p \mid Q \mid \neg Q \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle \phi \mid [a]\phi \mid \mu Q.\phi \mid \neg\mu Q.\phi$$

where no variable is quantified twice and ϕ is syntactically monotone in Q in the cases of $\mu Q.\phi$ and $\neg\mu Q.\phi$.

3 Logical Approach to Static Analysis

In our logical approach to static analysis, we specify analysis constraints in *clause sequences*. Assume that we are given a fixed countable set $\mathcal{X}$ of variables and a finite alphabet $\mathcal{R}$ of predicate symbols. We define the syntax of clause sequences cls, together with basic values v, pre-conditions pre and clauses cl as follows:

$$\begin{array}{lll} v & ::= & c \mid x \\ pre & ::= & R(v_1, ..., v_n) \mid \neg R(v_1, ..., v_n) \mid pre_1 \wedge pre_2 \\ & & \mid pre_1 \vee pre_2 \mid \forall x : \ pre \mid \exists x : \ pre \\ cl & ::= & R(v_1, ..., v_n) \mid \mathbf{true} \mid cl_1 \wedge cl_2 \mid pre \Rightarrow R(v_1, ..., v_n) \mid \forall x : \ cl \\ cls & ::= & cl_1, ..., cl_n \end{array}$$

The pre-conditions, clauses and clause sequences are interpreted over a finite and non-empty universe $\mathcal{U}$. A constant c is an element of $\mathcal{U}$, a variable $x \in \mathcal{X}$ ranges over $\mathcal{U}$, and the n-ary relation $R \in \mathcal{R}$ denotes a subset of $\mathcal{U}^n$. We use $pre \Rightarrow R(v_1, ..., v_n)$ instead of $pre \Rightarrow cl$ which is used in [15] to simplify our development, but this does not restrict the expressiveness (merely the succinctness) of our approach.

Occurrences of $R(v_1, ..., v_n)$ and $\neg R(v_1, ..., v_n)$ in pre-conditions are called *positive queries* and *negative queries*, respectively. All other occurrences of relations are *definitions* and often occur to the right of an implication. To deal with negations conveniently, we are often interested in some subsets of clause sequences defined by the above grammar.

Let $Int : \prod_k Rel_k \to \mathcal{P}(\mathcal{U}^k)$ be a mapping where Rel_k is a finite alphabet of k-ary predicate symbols and $\mathcal{P}(\mathcal{U}^k)$ is the powerset of $\mathcal{U}^k$. We define the satisfaction relations for pre-conditions, clauses and clause sequences $(\rho, \sigma)\ \underline{\texttt{sat}}\ pre$, $(\rho, \sigma)\ \underline{\texttt{sat}}\ cl$ and $(\rho, \sigma)\ \underline{\texttt{sat}}\ cls$ in Table 1, where $\rho \in Int$ is an interpretation of relations which maps each k-ary predicate symbol R to a subset of $\mathcal{U}^k$ and σ is an interpretation of variables. We write $\rho(R)$ for the set of k-tuples $(a_1, ...a_k)$ from $\mathcal{U}$ associated with the k-ary predicate R, we use $\sigma(x)$ to denote the atom of $\mathcal{U}$ bound to x and $\sigma[x \mapsto a]$ stands for the mapping that is σ except that x is mapped to a. We also treat a constant c as a variable by setting $\sigma(c) = c$.

Table 1. Semantics of Pre-conditions, Clauses and Clause Sequences

$(\rho, \sigma)\ \underline{\texttt{sat}}\ R(v_1, ..., v_n)$	$\texttt{iff}\ (\sigma(v_1), ..., \sigma(v_n)) \in \rho(R)$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ \neg R(v_1, ..., v_n)$	$\texttt{iff}\ (\sigma(v_1), ..., \sigma(v_n)) \notin \rho(R)$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ pre_1 \wedge pre_2$	$\texttt{iff}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ pre_1\ \texttt{and}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ pre_2$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ pre_1 \vee pre_2$	$\texttt{iff}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ pre_1\ \texttt{or}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ pre_2$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ \forall x :\ pre$	$\texttt{iff}\ (\rho, \sigma[x \mapsto a])\ \underline{\texttt{sat}}\ pre\ \texttt{for all}\ a \in \mathcal{U}$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ \exists x :\ pre$	$\texttt{iff}\ (\rho, \sigma[x \mapsto a])\ \underline{\texttt{sat}}\ pre\ \texttt{for some}\ a \in \mathcal{U}$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ R(v_1, ..., v_n)$	$\texttt{iff}\ (\sigma(v_1), ..., \sigma(v_n)) \in \rho(R)$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ \mathbf{true}$	$\texttt{iff}\ \mathbf{true}$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ cl_1 \wedge cl_2$	$\texttt{iff}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ cl_1\ \texttt{and}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ cl_2$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ pre \Rightarrow R(v_1, ..., v_n)$	$\texttt{iff}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ R(v_1, ..., v_n)\ \texttt{whenever}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ pre$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ \forall x :\ cl$	$\texttt{iff}\ (\rho, \sigma[x \mapsto a])\ \underline{\texttt{sat}}\ cl\ \texttt{for all}\ a \in \mathcal{U}$
$(\rho, \sigma)\ \underline{\texttt{sat}}\ cl_1, ..., cl_n$	$\texttt{iff}\ (\rho, \sigma)\ \underline{\texttt{sat}}\ cl_i\ \texttt{for all}\ i\ \texttt{where}\ 1 \leq i \leq n$

A clause sequence with no free variables is called *closed*, and in closed clause sequences the interpretation σ is of no importance. For a fixed interpretation σ_0, when cls is closed, we have that $(\rho, \sigma)\ \underline{\texttt{sat}}\ cls$ agrees with $(\rho, \sigma_0)\ \underline{\texttt{sat}}\ cls$. We call an interpretation ρ a solution, or a model, of cls whenever $(\rho, \sigma_0)\ \underline{\texttt{sat}}\ cls$ holds.

Central to our approach to static analysis is the establishment of an *intended model* of cls. We often consider the least model of cls as a candidate, since that is the most precise analysis result. We briefly review ALFP in Section 3.1. ALFP restricts itself to the *stratified* fragment of clause sequences. The intended model of an ALFP formula is defined by the least model characterized by Moore Family properties. We propose *Succinct Fixed Point Logic* in Section 3.2. SFP restricts itself to the *weakly stratified* fragment of clause sequences. The Moore Family result of SPF is established in a slightly different way and the model of an SFP formula is defined as the least model characterized by Moore Family properties as well.

3.1 Alternation-Free Least Fixed Point Logic

Alternation-free Least Fixed Point Logic is more expressive than Datalog [19, 20] and has been used in a number of papers for specifying static analysis. It has proved to be very useful for obtaining efficient implementations of static analyses and there are a number of solvers available [17]. A clause sequence *cls* is called an ALFP formula iff it is stratified. The notion of *stratification* is given as follows.

Definition 3. *A clause sequence $cls = cl_1, ..., cl_n$ is stratified if there is a ranking function $\texttt{rank} : \mathcal{R} \to \{0, ..., n\}$ such that the following holds for $0 \leq i \leq n$:*

- *if cl_i contains a definition of R then $rank(R) = i$;*
- *if cl_i contains a positive query of R then $rank(R) \leq i$; and*
- *if cl_i contains a negative query of R then $rank(R) < i$.*

Example 1. The following clause sequence is not in ALFP since it is ruled out by the notion of stratification:

$$cls = (\forall x : R_1(x) \Rightarrow R_2(x)), (\forall x : \neg R_2(x) \Rightarrow R_1(x))$$

This is because it is not possible that we have both $rank(R_1) \leq rank(R_2)$ and $rank(R_2) < rank(R_1)$.

According to the choice of ranks we have made, we define a lexicographic ordering, $\sqsubseteq$, for the interpretations of relations, ρ, as follows: $\rho_1 \sqsubseteq \rho_2$ if there exists a rank $i \in \{0, ..., r\}$ such that (1) $\rho_1(R) = \rho_2(R)$ whenever $rank(R) < i$, (2) $\rho_1(R) \subseteq \rho_2(R)$ whenever $rank(R) = i$, and (3) either $i = r$ or $\rho_1(R) \subset \rho_2(R)$ for some R with $rank(R) = i$. We define $\rho_1 \subseteq \rho_2$ to mean $\rho_1(R) \subseteq \rho_2(R)$ for all $R \in \mathcal{R}$.

The set of interpretations of relations constitutes a complete lattice with respect to $\sqsubseteq$. We know from [15] that the set of solutions to an ALFP formula constitutes a Moore Family. Recall that a Moore Family [7] is a subset Y of a complete lattice $L = (L, \sqsubseteq)$ that is closed under greatest lower bounds: $\forall Y' \subseteq Y : \bigsqcap Y' \in Y$. The Moore Family result of ALFP is given as follows:

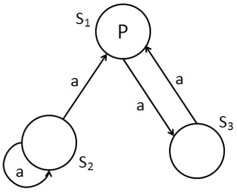
Proposition 1. *The set $\{\rho | (\rho, \sigma_0) \ \underline{\texttt{sat}}\ cls\}$ is a Moore Family, i.e. is closed under greatest lower bounds, whenever cls is closed and stratified; the greatest lower bound $\sqcap \ \{\rho | (\rho, \sigma_0) \ \underline{\texttt{sat}}\ cls\}$ is the least model of cls.*

More generally, given ρ_0 the set $\{\rho | (\rho, \sigma_0) \ \underline{\texttt{sat}}\ cls \wedge \rho_0 \subseteq \rho\}$ is a Moore Family and $\sqcap \ \{\rho | (\rho, \sigma_0) \ \underline{\texttt{sat}}\ cls \wedge \rho_0 \subseteq \rho\}$ is the least model.

The Moore Family result of ALFP formulas ensures the existence of a unique least model. We take the least model as the unique intended model of our analysis constraints specified by ALFP formulas.

ALFP suffices [21] to encode the alternation-free fragment of the μ-calculus, where nesting of least and greatest fixed points are prohibited. We give an example in the following.

Example 2. Consider a Kripke structure, given by the diagram to the left, where $S = \{s_1, s_2, s_3\}$, the transition relation $T = \{a\}$ is represented by edges labeled with a between states, and L labels s_1 with proposition p.



$\varrho(R_Q)$	$[\![\mu Q.[a](p \vee Q)]\!]$
$\{s_1, s_3\}$	$\{s_1, s_3\}$

We evaluate the formula $\mu Q.[a](p \vee Q)$ over the above Kripke structure using ALFP and the semantics of the μ-calculus respectively. The results are given in the table to the right.

In our static analysis approach, we first encode the above Kripke structure in ϱ_0 by defining $\varrho_0(P_p) = \{s_1\}$ and $\varrho_0(T_a) = \{(s_2, s_1), (s_2, s_2), (s_1, s_3), (s_3, s_1)\}$. Here, the universe is $\mathcal{U} = S$. The relation P_p specifies the set of states on which the atomic proposition p holds, and the relation T_a specifies the transition relation of the given Kripke structure. Then we specify the formula $\mu Q.[a](p \vee Q)$ with the clause sequence $cls = \forall s : \forall s' : \neg T_a(s, s') \vee P_p(s') \vee R_Q(s') \Rightarrow R_Q(s)$. The relation R_Q intends to characterize $[\![\mu Q.[a](p \vee Q)]\!]_{[]}$. The least solution ρ to *cls* subject to $\varrho_0 \subseteq \rho$ can be calculated by *Succinct Solver* [15].

3.2 Succinct Fixed Point Logic

The condition of stratification in ALFP requires that the definition of a relation R in *cls* only depends on relations with ranks less or equal to R. In particular, the requirement that a relation must be defined before they can be negatively queried is essential. This makes it inconvenient for ALFP to specify nested fixed points in the μ-calculus, where least and greatest fixed points are mutually dependent on each other.

In this section, we propose *Succinct Fixed Point Logic* (SFP) to encode nested fixed points in the μ-calculus. We first define the syntax of SFP, which include basic values v, pre-conditions *pre*, clauses *cl*, clause sequences *cls* and formulas f, as follows:

Definition 4 (Syntax of Succinct Fixed Point Logic)

$$\begin{array}{lll} v & ::= & c \mid x \\ pre & ::= & R(v_1, ..., v_n) \mid \neg R(v_1, ..., v_n) \mid pre_1 \wedge pre_2 \\ & & \mid pre_1 \vee pre_2 \mid \forall x :\ pre \mid \exists x :\ pre \\ cl & ::= & R(v_1, ..., v_n) \mid \boldsymbol{true} \mid cl_1 \wedge cl_2 \mid pre \Rightarrow R(v_1, ..., v_n) \mid \forall x :\ cl \\ cls & ::= & cl_1, ..., cl_n \\ f & ::= & \boldsymbol{LFP}(cls) \end{array}$$

where cls is weakly stratified.

Here, we require that clause sequences are weakly stratified. The definition of *weak stratification* will be given later. We introduce a least fixed point operator **LFP** and $f = \mathbf{LFP}(cls)$ is defined as SFP formulas. This is mainly to facilitate the definition of the intended model of weakly stratified clause sequences. Our intention is that ρ is the intended model of cls iff ρ satisfies the formula $\mathbf{LFP}(cls)$.

To formalize the notion of weak stratification, we first give the definition of *Dependency Graph* as follows.

Definition 5 (Dependency Graph). *The dependency graph DG_{cls} of $cls = cl_1, ..., cl_n$ is a directed graph where each edge is labeled with a sign. The nodes of DG_{cls} are cl_1,...,cl_n. We define a positive (resp. negative) edge from cl_i to cl_j iff a relation defined in cl_i is positively (resp. negatively) queried in cl_j, where $1 \leq i, j \leq n$.*

We say that cl_j *depends positively (resp. negatively)* on cl_i iff there exists a path in DG_{cls} from cl_i to cl_j with even (resp. odd) number of negative edges.

Definition 6 (Weak Stratification). *A clause sequence $cls = cl_1, ..., cl_n$ is weakly stratified iff the following conditions hold, where $1 \leq i, j \leq n$, $i \neq j$ and $R \in \mathcal{R}$:*

- *if R is defined in cl_i, then R is not defined in cl_j, and*
- *cl_i does not depend negatively on itself.*
- *if cl_i depends positively (resp. negatively) on cl_j, then cl_i does not depend negatively (resp. positively) on cl_j.*

The first condition in the above definition simply says that we use only one clause to define each relation. The second condition imposes *syntactic monotonicity* to the clause sequence. The last condition is actually used to facilitate the establishment of a Moore Family result for SFP.

Example 3. The following clause sequence satisfies the condition of weak stratification.

$$cls = (\forall x : \neg R_2(x) \Rightarrow R_1(x)), (\forall x : \neg R_1(x) \Rightarrow R_2(x))$$

Example 4. The following clause sequence is ruled out by the notion of weak stratification. We can see that the clause $(\forall x : R_2(x) \Rightarrow R_1(x))$ depends negatively on itself.

$$cls = (\forall x : R_2(x) \Rightarrow R_1(x)), (\forall x : \neg R_1(x) \Rightarrow R_2(x))$$

Let's consider the following example where we specify a μ-calculus formula of nested fixed points with a weakly stratified clause sequence.

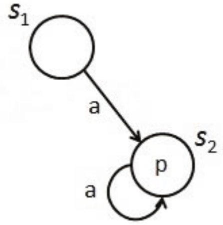
Example 5. Consider the μ-calculus formula $\phi = \mu Q_1.(\neg \mu Q_2.(Q_2 \vee (\neg Q_1 \wedge p)))$, which is semantically equivalent to $\mu Q_1.(\nu Q_2.(Q_2 \wedge (Q_1 \vee \neg p)))$ and therefore consists of nested fixed points. The formula ϕ can be specified by the following clause sequence cls.

$$cls = [\forall s : \neg R_{Q_2}(s) \Rightarrow R_{Q_1}(s)], [\forall s : [R_{Q_2}(s) \vee (\neg R_{Q_1}(s) \wedge P_p(s))] \Rightarrow R_{Q_2}(s)]$$

The clause sequence *cls* is weakly stratified. The relation P_p intends to specify the set of states, in a given Kripke structure, on which the atomic proposition p holds. The relation R_{Q_1} (resp. R_{Q_2}) intends to characterize $[\![\phi]\!]_{[]}$ (resp. $[\![\mu Q_2.(Q_2 \vee (\neg Q_1 \wedge p))]\!]_{[Q_1 \mapsto [\![\phi]\!]_{[]}]}$).

The next step is to define an intended model ρ of *cls*. In our setting, this amounts to define the semantics of formulas $f = \mathbf{LFP}(cls)$. Our intention is to use ρ to encode the fixed point semantics in the μ-calculus. Our first try is to define it in a similar way as we do in ALFP. Let's assume that all relations defined in a clause cl_i have the same rank and that all predefined relations have rank 0. However, we show through the following example that we cannot define the intended model ρ of *cls* as $\sqcap\{\rho|(\rho, \sigma_0)\ \underline{\mathtt{sat}}\ cls \wedge \rho_0 \subseteq \rho\}$, where ρ_0 defines all predefined relations, with respect to $\sqsubseteq$, since it does not capture the fixed point semantics.

Example 6. Consider the Kripke structure $M = (S, T, L)$, given by the diagram to the left, where $S = \{s_1, s_2\}$, $T = \{a\}$, $a = \{(s_1, s_2), (s_2, s_2)\}$, and L labels s_2 with the proposition p. We encode the μ-calculus formula $\phi = \mu Q_1.(\neg \mu Q_2.(Q_2 \vee (\neg Q_1 \wedge p)))$ in the same clause sequence $cls = [\forall s : \neg R_{Q_2}(s) \Rightarrow R_{Q_1}(s)], [\forall s : [R_{Q_2}(s) \vee (\neg R_{Q_1}(s) \wedge P_p(s))] \Rightarrow R_{Q_2}(s)]$ as we do in Example 5. We evaluate ϕ over M using SFP and the semantics of the μ-calculus respectively.



	ρ_1	ρ_2	ρ_3
R_{Q_2}	$\{s_1, s_2\}$	$\emptyset$	$\{s_2\}$
R_{Q_1}	$\emptyset$	$\{s_1, s_2\}$	$\{s_1\}$
P_p	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$

Assume we have an initial interpretation ρ_0, where $\rho_0(P_p) = \{s_2\}$ and $\rho_0(R_{Q_1}) = \rho_0(R_{Q_2}) = \emptyset$. We now consider the set of interpretations $I = \{\rho|(\rho, \sigma_0)\ \underline{\mathtt{sat}}\ cls \wedge \rho_0 \subseteq \rho\}$ according to the semantics in Table 1. There are at least three solutions ρ_1, ρ_2 and ρ_3, given in the table to the right, in the set I.

We can take at most two essentially different ranking functions $rank_1$ and $rank_2$, where $rank_1(P_p) = 0$, $rank_1(R_{Q_1}) = 1$ and $rank_1(R_{Q_2}) = 2$, $rank_2(P_p) = 0$, $rank_2(R_{Q_1}) = 2$ and $rank_2(R_{Q_2}) = 1$. Let $e = [Q_1 \mapsto [\![\phi]\!]_{[]}, Q_2 \mapsto [\![\mu Q_2.(Q_2 \vee (\neg Q_1 \wedge p))]\!]_{[Q_1 \mapsto [\![\phi]\!]_{[]}]}]$. According to the semantics of the μ-calculus, we know that $[\![Q_1]\!]_e = \{s_1\}$ and $[\![Q_2]\!]_e = \{s_2\}$. We can see that ρ_3 exactly characterizes the semantics of the μ-calculus in our example. However, due to the existence of ρ_1 and ρ_2, the solution ρ_3 is not the least model in I for either $rank_1$ or $rank_2$.

The method of establishing an intended model of *cls* in the above example can be summarized as follows. First, we calculate all the models that satisfy *cls*. Second, we make a choice of ranks for all those relations defined in *cls*. Last, we choose the least model as the intended model of *cls*, according to the lexicographic ordering with respect to the choice of ranks we have made. This method applies well when we approximate an analysis where analysis information only flows from the lowest rank to the highest rank. Therefore, ALFP successfully characterizes the semantics of the alternation-free μ-calculus, where information flows from inner fixed points to outer fixed points since nesting of fixed points operators of different types are prohibited.

In the following, we define the semantics of formulas f. We assume that $cls = cl_1, ..., cl_n$ and write $\rho = \varrho_0, \varrho_1, ..., \varrho_n$ to mean that ϱ_0 is an interpretation for some predefined relations and ϱ_i $(1 \leq i \leq n)$ is an interpretation of relations defined in cl_i. We use $\rho[\varrho'_i/\varrho_i]$ to denote a new interpretation updated from ρ by substituting ϱ_i with ϱ'_i. Let ϱ_i and ϱ'_i be two interpretations of relations defined in cl_i. We define that $\varrho_i \subseteq \varrho'_i$ iff for all relations R defined in cl_i, $\varrho_i(R) \subseteq \varrho'_i(R)$ holds. The set of interpretations defined in cl_i constitute a complete lattice with respect to $\subseteq$. The satisfaction relation (ρ, σ) $\underline{\texttt{sat}}$ $\mathbf{LFP}(cl_1, ..., cl_n)$ is defined in the following.

Definition 7 (Semantics of SFP formulas). *Let $\rho = \varrho_0, ..., \varrho_n$ be an interpretation and $cls = cl_1, ..., cl_n$ a weakly stratified clause sequence. The satisfaction relation (ρ, σ) $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_1, ..., cl_n)$ is defined inductively as follows:*

- (ρ, σ) $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_n)$ *iff* $\varrho_n = \sqcap\{\varrho'_n \mid (\rho[\varrho'_n/\varrho_n], \sigma)$ $\underline{\texttt{sat}}$ $cl_n\}$
- (ρ, σ) $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_i, ..., cl_n)$ *iff*
  1. (ρ, σ) $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_{i+1}, ..., cl_n)$, *and*
  2. $\varrho_i = \sqcap\{\varrho'_i \mid \exists \varrho'_{i+1}, ..., \varrho'_n : (\rho[\varrho'_i/\varrho_i, ..., \varrho'_n/\varrho_n], \sigma)$ $\underline{\texttt{sat}}$ $cl_i \wedge$ $(\rho[\varrho'_i/\varrho_i, ..., \varrho'_n/\varrho_n], \sigma)$ $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_{i+1}, ..., cl_n)\}$

The Moore Family properties for weakly stratified clause sequence $cls = cl_1, ..., cl_n$ is established as follows.

Theorem 1. *Let $\rho = \varrho_0, ..., \varrho_n$ be an interpretation, $cls = cl_1, ..., cl_n$ a weakly stratified clause sequence and $1 \leq i \leq n$. Then, we have the followings:*

- *The set of interpretations* $\{\varrho'_n \mid (\rho[\varrho'_n/\varrho_n], \sigma)$ $\underline{\texttt{sat}}$ $cl_n\}$ *is a Moore Family*
- *The set of interpretations* $\{\varrho'_i \mid \exists \varrho'_{i+1}, ..., \varrho'_n : (\rho[\varrho'_i/\varrho_i, ..., \varrho'_n/\varrho_n], \sigma)$ $\underline{\texttt{sat}}$ $cl_i \wedge$ $(\rho[\varrho'_i/\varrho_i, ..., \varrho'_n/\varrho_n], \sigma)$ $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_{i+1}, ..., cl_n)\}$ *is a Moore Family.*

We define the intended model of a weakly stratified clause sequence below.

Definition 8. *Assume that $cls = cl_1, ..., cl_n$ is a weakly stratified clause sequence. The model ρ is an intended model of cls iff (ρ, σ) $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_1, ..., cl_n)$.*

The Moore Family properties of SFP leads to the following theorem which guarantees the existence and the uniqueness of the intended model of cls.

Theorem 2. *Let $cls = cl_1, ..., cl_n$ be a weakly stratified clause sequence. The model ρ such that (ρ, σ) $\underline{\texttt{sat}}$ $\boldsymbol{LFP}(cl_1, ..., cl_n)$ exists and is unique.*

Example 7. Let's reconsider the problem in Example 6 again and show how to find the model $\rho = \varrho_0, \varrho_1, \varrho_2$ to the formula $\mathbf{LFP}(cls)$. Let's write $cls = cl_1, cl_2$ where $cl_1 = [\forall s : \neg R_{Q_2}(s) \Rightarrow R_{Q_1}(s)]$ and $cl_2 = [\forall s : [R_{Q_2}(s) \vee (\neg R_{Q_1}(s) \wedge P_p(s))] \Rightarrow R_{Q_2}(s)]$. According to Definition 7, (ρ, σ) $\underline{\texttt{sat}}$ $\mathbf{LFP}(cl_1, cl_2)$ iff (ρ, σ) $\underline{\texttt{sat}}$ $\mathbf{LFP}(cl_2)$ and $\varrho_1 = \sqcap\{\varrho_1' \mid \exists \varrho_2' : (\rho[\varrho_1'/\varrho_1, \varrho_2'/\varrho_2], \sigma)$ $\underline{\texttt{sat}}$ $cl_1 \wedge (\rho[\varrho_1'/\varrho_1, \varrho_2'/\varrho_2], \sigma)$ $\underline{\texttt{sat}}$ $\mathbf{LFP}(cl_2)\}$.

We first calculate the set of interpretations such that (ρ, σ) $\underline{\texttt{sat}}$ $\mathbf{LFP}(cl_2)$. To this end, we first list all the interpretations such that (ρ, σ) $\underline{\texttt{sat}}$ cl_2 in Table 2. In this case, relations P_p and R_{Q_1} are predefined relations for the clause cl_2.

Table 2. (ρ, σ) $\underline{\texttt{sat}}$ cl_2

	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9	ρ_{10}	ρ_{11}	ρ_{12}
R_{Q_2}	$\{s_2\}$	$\{s_1, s_2\}$	$\{s_2\}$	$\{s_1, s_2\}$	$\emptyset$	$\{s_1$	$\{s_2\}$	$\{s_1, s_2\}$	$\emptyset$	$\{s_1\}$	$\{s_2\}$	$\{s_1, s_2\}$
R_{Q_1}	$\emptyset$	$\emptyset$	$\{s_1\}$	$\{s_1\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_1, s_2\}$	$\{s_1, s_2\}$	$\{s_1, s_2\}$	$\{s_1, s_2\}$
P_p	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$	$\{s_2\}$

The next step is to select those interpretations which satisfy $\mathbf{LFP}(cl_2)$ from Table 2. From all those interpretations which coincide on predefined relations, we choose the one with the best analysis result for R_{Q_2}. Let's take ρ_1 and ρ_2 as an example. The models ρ_1 and ρ_2 coincide on their interpretations for P_p and R_{Q_1}. However, $\rho_1(R_{Q_2}) = \sqcap\{\rho_1(R_{Q_2}), \rho_2(R_{Q_2})\}$. Therefore, (ρ_1, σ) $\underline{\texttt{sat}}$ $\mathbf{LFP}(cl_2)$. The result of our selection are $\{\rho_1, \rho_3, \rho_5, \rho_9\}$. These are the interpretations which satisfy $\mathbf{LFP}(cl_2)$.

We now select those interpretations which satisfy cl_1 from $\{\rho_1, \rho_3, \rho_5, \rho_9\}$ and see that only ρ_3 and ρ_9 do. The last step is to select from ρ_3 and ρ_9 the one which satisfies $\mathbf{LFP}(cl_1, cl_2)$. Since $\rho_3(R_{Q_1}) = \sqcap\{\rho_3(R_{Q_1}), \rho_9(R_{Q_1})\}$, we know that (ρ_3, σ) $\underline{\texttt{sat}}$ $\mathbf{LFP}(cl_1, cl_2)$. Notice that ρ_3 exactly characterized the fixed point semantics here.

4 Model Checking as Static Analysis

Here, we use Definition 2 to give the syntax of the μ-calculus. Given a μ-calculus formula ϕ, for each variable Q in ϕ, a relation R_Q is defined. We specify our analysis with a pair $\langle cls_\phi, pre_\phi \rangle$, where cls_ϕ is a weakly stratified clause sequence and pre_ϕ is a pre-condition.

Assume that $\rho = \varrho_0, ..., \varrho_n$ such that (ρ, σ) $\underline{\texttt{sat}}$ $\mathbf{LFP}(cls_\phi)$, where ϱ_0 is an initial interpretation which encodes a given Kripke structure and defines relations $R_{Q_1}, ..., R_{Q_n}$, where $Q_1, ..., Q_n$ are all the free variables in ϕ. The intention of our development is that $s' \in [\![\phi]\!]_{e[Q_1 \mapsto S_1, ..., Q_n \mapsto S_n]}$ iff $(\rho, \sigma[s \mapsto s'])$ $\underline{\texttt{sat}}$ pre_ϕ, and that when ϕ takes the form $\mu Q.\phi$, we have that $[\![\mu Q.\phi]\!]_{e[Q_1 \mapsto S_1, ..., Q_n \mapsto S_n]}$ equals $\rho(R_Q)$.

We encode a Kripke structure $M = (S, T, L)$ into SFP by defining the corresponding relations in ϱ_0 as follows. Assume that the universe is $\mathcal{U} = S$,

Table 3. μ-calculus in Succinct Fixed Point Logic

p	$\longmapsto$	$\langle \mathbf{true},\ P_p(s)\rangle$
Q	$\longmapsto$	$\langle \mathbf{true},\ R_Q(s)\rangle$
$\neg Q$	$\longmapsto$	$\langle \mathbf{true},\ \neg R_Q(s)\rangle$
$\phi_1 \vee \phi_2$	$\longmapsto$	$\langle (cls_{\phi_1}, cls_{\phi_2}),\ pre_{\phi_1} \vee pre_{\phi_2}\rangle$
		`whenever` $\phi_1 \longmapsto \langle cls_{\phi_1},\ pre_{\phi_1}\rangle$ `and` $\phi_2 \longmapsto \langle cls_{\phi_2},\ pre_{\phi_2}\rangle$
$\phi_1 \wedge \phi_2$	$\longmapsto$	$\langle (cls_{\phi_1}, cls_{\phi_2}),\ pre_{\phi_1} \wedge pre_{\phi_2}\rangle$
		`whenever` $\phi_1 \longmapsto \langle cls_{\phi_1},\ pre_{\phi_1}\rangle$ `and` $\phi_2 \longmapsto \langle cls_{\phi_2},\ pre_{\phi_2}\rangle$
$\langle a\rangle\phi$	$\longmapsto$	$\langle cls_{\phi},\ \exists s' : T_a(s,s') \wedge pre_{\phi}[s'/s]\rangle$
		`whenever` $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$
$[a]\phi$	$\longmapsto$	$\langle cls_{\phi},\ \forall s' : \neg T_a(s,s') \vee pre_{\phi}[s'/s]\rangle$
		`whenever` $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$
$\mu Q.\phi$	$\longmapsto$	$\langle ([\forall s : pre_{\phi} \Rightarrow R_Q(s)], cls_{\phi}),\ R_Q(s)\rangle$
		`whenever` $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$
$\neg\mu Q.\phi$	$\longmapsto$	$\langle cls_{\mu Q.\phi},\ \neg R_Q(s)\rangle$
		`whenever` $\mu Q.\phi \longmapsto \langle cls_{\mu Q.\phi},\ pre_{\mu Q.\phi}\rangle$

- for each atomic proposition p we define a predicate P_p such that $s \in \varrho_0(P_p)$ if and only if $p \in L(s)$,
- for each element a in T, we define a binary relation T_a such that $(s,t) \in \varrho_0(T_a)$ if and only if $(s,t) \in a$.

The mapping rules for $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$ is given in Table 3. The clause sequence cls_{ϕ} is used to define all the relations R_Q where Q is a bounded variable in ϕ. We use $pre_{\phi}[s'/s]$ to denote a pre-condition resulting from pre_{ϕ} by substituting the free variable s in pre_{ϕ} with s'.

In Table 3, the choice of the ordering of clauses in cls_{ϕ} is essential in our approach. Assume that $cls_{\phi} = cl_1, ..., cl_n$. We define only one relation in each clause cl_i $(1 \leq i \leq n)$. Assume that we are given a μ-calculus formula ϕ. We call a subformula of ϕ a μ-subformula iff its main connective is μ. Assume that $\mu Q_i.\varphi_1$ and $\mu Q_j.\varphi_2$ are two μ-subformulas in ϕ and we define R_{Q_i} (resp. R_{Q_j}) in cl_i (resp. cl_j), our intention is to ensure that $i < j$ if $\mu Q_j.\varphi_2$ is a subformula of $\mu Q_i.\varphi_1$. Therefore, in the case of $\mu Q.\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$, for example, we have that $cls_{\mu Q.\phi} = ([\forall s : pre_{\phi} \Rightarrow R_Q(s)], cls_{\phi})$ instead of $cls_{\mu Q.\phi} = (cls_{\phi}, [\forall s : pre_{\phi} \Rightarrow R_Q(s)])$.

We first explain the case of $\mu Q.\phi$. Here, Q is a bounded variable. Under the assumption that $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$ holds, we define $cls_{\mu Q.\phi}$ as $([\forall s : pre_{\phi} \Rightarrow R_Q(s)], cls_{\phi})$. The clause $[\forall s : pre_{\phi} \Rightarrow R_Q(s)]$ defines the relation R_Q and the clause sequence cls_{ϕ} defines all those relations $R_{Q'}s$ where Q' is a bounded variable in ϕ. We define $pre_{\mu Q.\phi}$ as $R_Q(s)$.

For atomic proposition p, we simply define cls_p as **true** since there are no bounded variables in p. We make use of the predefined predicate P_p and define pre_p as $P_p(s)$. For a variable Q, we also define cls_Q as **true** since the Q is a free variable here. We define pre_Q as $R_Q(s)$. For $\neg Q$, we define $cls_{\neg Q}$ as **true** and define $pre_{\neg Q}$ as $\neg R_Q(s)$.

For $\phi_1 \vee \phi_2$, we assume that $\phi_1 \longmapsto \langle cls_{\phi_1},\ pre_{\phi_1}\rangle$ and $\phi_2 \longmapsto \langle cls_{\phi_2},\ pre_{\phi_2}\rangle$. This means that for each subformula $\mu Q.\phi$ in ϕ_1 (resp. ϕ_2), the relation R_Q is defined in cls_{ϕ_1} (resp. cls_{ϕ_2}) and that pre_{ϕ_1} and pre_{ϕ_2} are also defined as expected. We define $cls_{\phi_1 \vee \phi_2}$ as $(cls_{\phi_1}, cls_{\phi_2})$. This ensures that for each bounded variable Q in $\phi_1 \vee \phi_2$, R_Q is defined in $(cls_{\phi_1}, cls_{\phi_2})$. It's natural to define $pre_{\phi_1 \vee \phi_2}$ as $pre_{\phi_1} \vee pre_{\phi_2}$. The case for $\phi_1 \wedge \phi_2$ follows the same pattern.

For $\langle a\rangle\phi$, we assume that $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$. We simply define that $cls_{\langle a\rangle\phi} = cls_{\phi}$ and this suffices to guarantee that for each bounded variable Q in $\langle a\rangle\phi$, the relation R_Q is defined in $cls_{\langle a\rangle\phi}$. We define $pre_{\langle a\rangle\phi}$ as $\exists s' : T_a(s, s') \wedge pre_{\phi}[s'/s]$. This means for any state s if $pre_{\phi}[s'/s]$ holds on any of the a-derivative s' of s, then $pre_{\langle a\rangle\phi}$ holds on state s. This matches the semantics for $\langle a\rangle\phi$.

For $[a]\phi$, we also assume that $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$. For a similar reason as in the case for $\langle a\rangle\phi$, we define that $cls_{[a]\phi} = cls_{\phi}$. We define $pre_{[a]\phi}$ by $\forall s' : \neg T_a(s, s') \vee pre_{\phi}[s'/s]$. This means for any state s if $pre_{\phi}[s'/s]$ holds on all of the a-derivative s' of s, then $pre_{[a]\phi}$ holds on state s.

For $\neg\mu Q.\phi$, we assume that $\mu Q.\phi \longmapsto \langle cls_{\mu Q.\phi},\ pre_{\mu Q.\phi}\rangle$. We define that $cls_{\neg\mu Q.\phi} = cls_{\mu Q.\phi}$. We simply define $pre_{\neg\mu Q.\phi}$ as $\neg R_Q(s)$.

We have the following lemma which ensures that our specification of the μ-calculus formulas is within SFP.

Lemma 1. *Given a closed μ-calculus formula ϕ, assume that $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$ holds according to Table 3, the clause sequence cls_{ϕ} is closed and weakly stratified.*

The following theorem shows that the pre-condition pre_{ϕ} in our mapping $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$ correctly characterizes the semantics of ϕ.

Theorem 3. *Let ϕ be a μ-calculus formula with $Q_1, ..., Q_n$ being all the free variables in it. Assume that $\phi \longmapsto \langle cls_{\phi},\ pre_{\phi}\rangle$. Let $\rho = \varrho_0, ..., \varrho_n$ be an interpretation such that (ρ, σ) $\underline{\mathtt{sat}}$ $\boldsymbol{LFP}(cls_{\phi})$, where $\varrho_0(R_{Q_1}) = S_1, ..., \varrho_0(R_{Q_n}) = S_n$ and ϱ_0 defines P_p and T_a. Then, $s' \in [\![\phi]\!]_{e[Q_1 \mapsto S_1, ..., Q_n \mapsto S_n]}$ iff $(\rho, \sigma[s \mapsto s'])$ $\underline{\mathtt{sat}}$ pre_{ϕ}.*

We focus on closed μ-calculus formulas of the form $\mu Q.\phi$. This is not a restriction since $[\![\phi]\!] = [\![\mu Q.\phi]\!]$ when Q is not a free variable in ϕ. From Theorem 3, we have the following corollaries saying that the model of SFP formulas for the analysis of the μ-calculus coincides with the solution for the corresponding model checking problem.

Corollary 1. *Let $\mu Q.\phi$ be a closed μ-calculus formula. Assume that $\mu Q.\phi \longmapsto \langle cl_{\mu Q.\phi},\ pre_{\mu Q.\phi}\rangle$ holds. Let $\rho = \varrho_0, ..., \varrho_n$ be an interpretation such that (ρ, σ) $\underline{\mathtt{sat}}$ $\boldsymbol{LFP}(cls_{\mu Q.\phi})$, where ϱ_0 defines P_p and T_a. Then, we have that $[\![\mu Q.\phi]\!] = \rho(R_Q)$.*

5 Conclusion

Early works [9–12] have taken the view that static analysis problems can be reduced to model checking. In the other research direction, we have generalized the work in [13, 21] by showing that the model checking problem of the μ-calculus can also be reduced to static analysis as well. We first propose Succinct Fixed Point Logic as a specification language which allows convenient specifications of nest fixed points in the μ-calculus and then present a mapping which can encode the full fragment of the μ-calculus to SFP. We show that μ-calculus formulas of nested fixed points can be characterized as the intended model of SFP clause sequences.

A number of previous papers (surveyed in [8, 18]) have developed a uniform approach to static analysis using ALFP as the specification language. On top of the many theoretical results established for this approach also a number of solvers have been developed [17] to calculate the least model of ALFP. ALFP can be encoded in SFP by showing that the least model of an ALFP formula can be characterized as the model of a corresponding SFP formula. This encoding is conceptually obvious and we didn't give it here.

The link between model checking and logic programming has been investigated in [22–26], where model checkers based on logic programming have been implemented. In our future work, we are interested in developing an efficient solver to calculate the model for SFP formulas so that a model checker for the μ-calculus is also implicitly implemented.

Acknowledgements. The research presented in this paper has been supported by MT-LAB, a VKR Centre of Excellence for the Modelling of Information Technology.

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
2. Emerson, E.A., Lei, C.-L.: Efficient Model Checking in Fragments of the Propositional Mu-Calculus (Extended Abstract). In: LICS 1986, pp. 267–278 (1986)
3. Cleaveland, R., Steffen, B.: A Linear-Time Model-Checking Algorithm for the Alternation-Free Modal Mu-Calculus. Formal Methods in System Design 2(2), 121–147 (1993)
4. Andersen, H.R.: Model Checking and Boolean Graphs. Theor. Comput. Sci. 126(1), 3–30 (1994)
5. Baier, C., Katoen, J.-P.: Principles of model checking, pp. I-XVII, 1-975. MIT Press (2008)
6. Kozen, D.: Results on the Propositional mu-Calculus. Theor. Comput. Sci. 27, 333–354 (1983)
7. Nielson, F., Nielson, H.R., Hankin, C.: Principles of program analysis (2. corr. print), pp. I-XXI, 1-452. Springer (2005)
8. Nielson, H.R., Nielson, F.: Flow Logic: A Multi-paradigmatic Approach to Static Analysis. In: Mogensen, T.Æ., Schmidt, D.A., Sudborough, I.H. (eds.) The Essence of Computation. LNCS, vol. 2566, pp. 223–244. Springer, Heidelberg (2002)
9. Steffen, B.: Data Flow Analysis as Model Checking. In: Ito, T., Meyer, A.R. (eds.) TACS 1991. LNCS, vol. 526, pp. 346–365. Springer, Heidelberg (1991)
10. Steffen, B.: Generating Data Flow Analysis Algorithms from Modal Specifications. Sci. Comput. Program. 21(2), 115–139 (1993)

11. Schmidt, D.A., Steffen, B.: Program Analysis *as* Model Checking of Abstract Interpretations. In: Levi, G. (ed.) SAS 1998. LNCS, vol. 1503, pp. 351–380. Springer, Heidelberg (1998)
12. Schmidt, D.A.: Data Flow Analysis is Model Checking of Abstract Interpretations. In: POPL 1998, pp. 38–48 (1998)
13. Nielson, F., Nielson, H.R.: Model Checking *Is* Static Analysis of Modal Logic. In: Ong, L. (ed.) FOSSACS 2010. LNCS, vol. 6014, pp. 191–205. Springer, Heidelberg (2010)
14. De Nicola, R., Vaandrager, F.W.: Action Versus State Based Logics for Transition Systems. In: Guessarian, I. (ed.) LITP 1990. LNCS, vol. 469, pp. 407–419. Springer, Heidelberg (1990)
15. Nielson, F., Seidl, H., Nielson, H.R.: A Succinct Solver for ALFP. Nord. J. Comput. 9(4), 335–372 (2002)
16. Nielson, F.: Two-Level Semantics and Abstract Interpretation. Theor. Comput. Sci. 69(2), 117–242 (1989)
17. Filipiuk, P., Nielson, H.R., Nielson, F.: Explicit Versus Symbolic Algorithms for Solving ALFP Constraints. Electr. Notes Theor. Comput. Sci. 267(2), 15–28 (2010)
18. Nielson, H.R., Nielson, F., Pilegaard, H.: Flow Logic for Process Calculi. ACM Comput. Surv. 44(1), 3 (2012)
19. Apt, K.R., Blair, H.A., Walker, A.: Towards a Theory of Declarative Knowledge. In: Foundations of Deductive Databases and Logic Programming, pp. 89–148 (1988)
20. Chandra, A.K., Harel, D.: Computable Queries for Relational Data Bases. J. Comput. Syst. Sci. 21(2), 156–178 (1980)
21. Zhang, F., Nielson, F., Nielson, H.R.: Fixpoints vs. Moore Families. Student Research Forum at SOFSEM 2012 (2012)
22. Ramakrishna, Y.S., Ramakrishnan, C.R., Ramakrishnan, I.V., Smolka, S.A., Swift, T., Warren, D.S.: Efficient Model Checking Using Tabled Resolution. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 143–154. Springer, Heidelberg (1997)
23. Ramakrishnan, C.R.: A Model Checker for Value-Passing Mu-Calculus Using Logic Programming. In: Ramakrishnan, I.V. (ed.) PADL 2001. LNCS, vol. 1990, pp. 1–13. Springer, Heidelberg (2001)
24. Ramakrishnan, C.R., Ramakrishnan, I.V., Smolka, S.A., Dong, Y., Du, X., Roychoudhury, A., Venkatakrishnan, V.N.: XMC: A Logic-Programming-Based Verification Toolset. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 576–580. Springer, Heidelberg (2000)
25. Delzanno, G., Podelski, A.: Model Checking in CLP. In: Cleaveland, W.R. (ed.) TACAS 1999. LNCS, vol. 1579, pp. 223–239. Springer, Heidelberg (1999)
26. Delzanno, G., Podelski, A.: Constraint-based deductive model checking. STTT 3(3), 250–270 (2001)