Fixpoint Games on Continuous Lattices

PAOLO BALDAN, Università di Padova, Italy
BARBARA KÖNIG, Universität Duisburg-Essen, Germany
CHRISTINA MIKA-MICHALSKI, Universität Duisburg-Essen, Germany
TOMMASO PADOAN, Università di Padova, Italy

Many analysis and verifications tasks, such as static program analyses and model-checking for temporal logics, reduce to the solution of systems of equations over suitable lattices. Inspired by recent work on lattice-theoretic progress measures, we develop a game-theoretical approach to the solution of systems of monotone equations over lattices, where for each single equation either the least or greatest solution is taken. A simple parity game, referred to as fixpoint game, is defined that provides a correct and complete characterisation of the solution of systems of equations over continuous lattices, a quite general class of lattices widely used in semantics. For powerset lattices the fixpoint game is intimately connected with classical parity games for μ-calculus model-checking, whose solution can exploit as a key tool Jurdziński's small progress measures. We show how the notion of progress measure can be naturally generalised to fixpoint games over continuous lattices and we prove the existence of small progress measures. Our results lead to a constructive formulation of progress measures as (least) fixpoints. We refine this characterisation by introducing the notion of selection that allows one to constrain the plays in the parity game, enabling an effective (and possibly efficient) solution of the game, and thus of the associated verification problem. We also propose a logic for specifying the moves of the existential player that can be used to systematically derive simplified equations for efficiently computing progress measures. We discuss potential applications to the model-checking of latticed μ-calculi.

CCS Concepts: • **Theory of computation → Verification by model checking**; *Logic and verification*; *Modal and temporal logics*; • **Software and its engineering → Model checking**; *Semantics*; Automated static analysis;

Additional Key Words and Phrases: fixpoint equation systems, continuous lattices, parity games, μ-calculus

ACM Reference Format:
Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. 2019. Fixpoint Games on Continuous Lattices. *Proc. ACM Program. Lang.* 3, POPL, Article 26 (January 2019), 29 pages. https://doi.org/10.1145/3290339

1 INTRODUCTION

Systems of fixpoint equations are ubiquitous in formal analysis and verification. For instance, program analysis [Nielson et al. 1999] uses the flow graph of a program to generate a set of constraints specifying how the information of interest at the different program points is interrelated. The set

Authors' addresses: Paolo Baldan, Dipartimento di Matematica "Tullio Levi-Civita", Università di Padova, Via Trieste, 63, Padova, I-35121, Italy, baldan@math.unipd.it; Barbara König, Fakultät für Ingenieurwissenschaften, Abteilung Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, Lotharstraße 65, Duisburg, 47048, Germany, barbara_koenig@uni-due.de; Christina Mika-Michalski, Fakultät für Ingenieurwissenschaften, Abteilung Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, Lotharstraße 65, Duisburg, 47048, Germany, christina.mika-michalski@uni-due.de; Tommaso Padoan, Dipartimento di Matematica "Tullio Levi-Civita", Università di Padova, Via Trieste, 63, Padova, I-35121, Italy, padoan@math.unipd.it.

2475-1421/2019/1-ART26
https://doi.org/10.1145/3290339

of constraints can be viewed as a system of fixpoint equations, whose (least or greatest) solution provides a sound approximation of the properties of the program. Invariant/safety properties can be characterised as greatest fixpoints, while liveness/reachability properties as least fixpoints. Behavioural equivalences (for instance for process calculi) are typically defined as the solution of a fixpoint equation. The most famous example is bisimilarity that can be characterised as the greatest fixpoint of a suitable operator over the lattice of binary relations on the set of states (see, e.g., [Sangiorgi 2011]).

Almost invariably, in the mentioned settings, the involved functions are monotone and the domains of interest are complete lattices where the key result for deriving the existence of (least or greatest) fixpoints is Knaster-Tarski's fixpoint theorem [Tarski 1955].

Least and greatest fixpoint can be profitably mixed, in order to obtain expressive specification logics, among which the μ-calculus [Kozen 1983] is a classical example. The μ-calculus is very expressive, but the nesting of fixpoints increases the complexity of model-checking. Common approaches to the model-checking problem rely on an encoding in terms of parity games (see, e.g., [Bradfield and Walukiewicz 2018; Emerson and Jutla 1991; Stirling 1995]). The seminal paper [Jurdziński 2000] provides an algorithm for the solution of parity games which is polynomial in the number of states and exponential in (half of) the alternation depth, recently improved to quasi-polynomial in [Calude et al. 2017]. A detailed discussion of the complexity of μ-calculus model-checking can be found in [Bradfield and Walukiewicz 2018].

It has been recently observed in [Hasuo et al. 2016] that progress measures, a key ingredient in Jurdzisńki's algorithm for solving parity games, are amenable to a generalisation to systems of fixpoint equations over general lattices. A constructive characterisation of such progress measures is given in the case of powerset lattices and used to derive model-checking procedures for (branching and linear) coalgebraic logic. For general lattices, however, the notion of progress measure in [Hasuo et al. 2016] does not exactly correspond to Jurdzinski's notion. In particular, there is no algorithm for actually computing such progress measures, they rather play the role of invariants respectively ranking functions that have somehow to be provided. While the possibility of deriving generic algorithms for solving systems of equations is very appealing, the restriction to powerset lattices limits the applicability of the technique. Often program analysis relies on lattices which are not powerset lattices (and neither distributive, hence they cannot be seen as sublattices of powerset lattices). Moreover also settings involving fuzziness, probabilities or in general quantitative information are not captured by restricting to powerset lattices.

Inspired by the mentioned work, in this paper we devise a game-theoretical approach to the solution of systems of fixpoint equations over a vast class of lattices, the so-called continuous lattices. Originally studied by Scott in connection with the semantics of the λ-calculus [Scott 1972], they have later been recognised as a fundamental structure, with a plethora of applications in the semantics of programming languages and, more generally, in the theory of computation [Abramsky and Jung 1994; Gierz et al. 2003]. They include discrete structures, such as most domains used in program analysis, and continuous structures, such as the real interval $[0, 1]$.

The possibility of characterising the least or the greatest fixpoint of a (single) monotone function over a powerset lattice in terms of a game between an existential and an universal player is probably folklore and has been observed in [Venema 2008] where the game is referred to as an unfolding game. As a first result, here we show how the unfolding game can be extended to work for a *system* of fixpoint equations over lattices, resulting in a surprisingly simple game that we refer to as a *fixpoint game*. Mixing least and greatest fixpoint equations requires a non-trivial winning condition, which however arises as a natural adaptation to our setting of the one for parity games.

For the simpler case of powerset lattices the interaction between the players in the fixpoint game fundamentally relies on the possibility of testing the presence of elements in the image of a set and on the fact that a subset is completely determined by the elements that belongs to it. When moving to a more general class of lattices we need to ensure that this kind of interaction can be suitably mimicked. We argue in the paper that continuous lattices provide an extremely natural setting for this extension, providing exactly the necessary machinery for stating results in a way which is analogous to the powerset case. In fact, they come equipped with a notion of "finitary" approximation based on the *way-below* relation and each element arises as the join of the elements which are way-below it, in the same way as a subset is the union of its singletons.

We show how Jurdziński's approach for solving parity games [Jurdziński 2000] can be generalised to systems of fixpoint equations over continuous lattices. In particular we introduce a notion of progress measure for fixpoint games over continuous lattices and we prove the existence of suitably defined small progress measures. This result enables a constructive characterisation of progress measures as (least) fixpoints and provides a recipe for computing the progress measure that can be straightforwardly implemented, at least for finite lattices.

We refine the fixpoint characterisation of progress measures by introducing the notion of selection, which basically constrains the moves of the existential player in the parity game, still preserving correctness and completeness, thus enabling a more efficient solution of the game. We also define a logic for providing a symbolic representation of the moves of the existential player that can be directly translated into a system of fixpoint equations describing the progress measure.

As an example of application beyond standard μ-calculus model-checking we will discuss the case of latticed μ-calculi, where the evaluation of a formula for a state gives a lattice element, generalising the standard truth values 0, 1 (see, e.g., [Eleftheriou et al. 2012; Grumberg et al. 2005; Kupfermann and Lustig 2007]). This happens naturally also when μ-calculus formulae are evaluated over weighted transition systems or over probabilistic automata [Huth and Kwiatkowska 1997].

Summing up, our main contributions are the following:

- We propose a game-theoretical characterisation of the solution of systems of fixpoint equations over lattices and we identify continuous lattices as a general and appropriate setting for such theory.
- We develop a theory of progress measures à la Jurdziński in this general framework, with a clear recipe for their computation. This can be seen as a generalisation of the MC progress measures, proposed in [Hasuo et al. 2016] for coalgebraic logics over powerset lattices.
- We devise strategies for the computation of such progress measures based on selections and a logic for the symbolic representation of players' moves, along with a complexity analysis.
- We explicitly discuss an application scenario beyond standard μ-calculus over powerset lattices, namely the model-checking of latticed μ-calculi via progress measures.

We believe that due to the generality of our results, there is the potential for several more interesting applications for different lattices. In particular, when the lattice under consideration is infinite, it is known that, despite the functions involved in the equations of the system being continuous, due to alternation of least and greatest fixpoints, discontinuous functions may arise and we possibly have to refer to ordinals beyond ω [Fontaine 2008; Mio and Simpson 2015]. Still some preliminary investigations, detailed in [Baldan et al. 2018], reveal the possibility of adapting our framework to properly work with infinite lattices, providing a technique that is always correct, and complete under certain conditions (for instance, on well-orders without other restrictions, or on the reals, suitably restricting the functions). In particular, for systems of fixpoint equations over the reals, as considered in [Mio and Simpson 2015, 2017], the game for solving fixpoint equations over the reals can be encoded into an SMT formula of fixed size, representing the winning condition of the existential player.

The rest of the paper is structured as follows. In § 2 we recap the basics of continuous lattices and introduce some notation that will be used throughout the paper. In § 3 we introduce the systems of fixpoint equations over a lattice, we define their solution and devise a corresponding notion of approximation. In § 4 we present a game-theoretical approach to the solution of a system of equations over a continuous lattice, together with several case studies. In § 5 we introduce the notion of progress measure for (the game associated with) systems of fixpoint equations over a continuous lattice. In § 6 we discuss an application to the model-checking of latticed μ-calculi.

In § 7 we conclude the paper and outline future research. All proofs, further details on the encoding of μ-calculus formulae into fixpoint equation systems (and vice versa) and a detailed comparison to [Hasuo et al. 2016] are included in the full version [Baldan et al. 2018].

2 PRELIMINARIES ON ORDERED STRUCTURES

In this section we provide the basic order theoretic notions that will be used throughout the paper. In particular, we define continuous lattices and we provide some notation about tuples of elements that will be useful for compactly describing the solution of systems of equation.

2.1 Lattices

A preordered or partially ordered set $\langle P, \sqsubseteq \rangle$ is often denoted simply as P, omitting the (pre)order relation. It is *well-ordered* if every non-empty subset $X \subseteq P$ has a minimum. The *join* and the *meet* of a subset $X \subseteq P$ (if they exist) are denoted $\bigsqcup X$ and $\bigsqcap X$, respectively.

Definition 2.1 (complete lattice, basis, irreducibles). A *complete lattice* is a partially ordered set $(L, \sqsubseteq)$ such that each subset $X \subseteq L$ admits a join $\bigsqcup X$ and a meet $\bigsqcap X$. A complete lattice $(L, \sqsubseteq)$ always has a least element $\bot = \bigsqcup \emptyset$ and a greatest element $\top = \bigsqcap \emptyset$. Given an element $l \in L$ we define its upward-closure $\uparrow l = \{l' \mid l' \in L \wedge l \sqsubseteq l'\}$. A *basis* for a lattice is a subset $B_L \subseteq L$ such that for each $l \in L$ it holds that $l = \bigsqcup \{b \in B_L \mid b \sqsubseteq l\}$. An element $l \in L$ is *completely join-irreducible* if whenever $l = \bigsqcup X$ for some $X \subseteq L$ then $l \in X$.

Since all lattices in this paper will be complete, we will often omit the qualification "complete". Similarly, since we are only interested in completely join-irreducible elements we will often omit the qualification "completely". Note that $\bot$ is never an irreducible since $\bot = \bigsqcup \emptyset$ and $\bot \notin \emptyset$.

Example 2.2. Three simple examples of lattices, that we will refer to later, are:

- The powerset of any set X, ordered by subset inclusion $(2^X, \sqsubseteq)$. Join is union, meet is intersection, top is X and bottom is $\emptyset$. A basis is the set of singletons $B_{2^X} = \{\{x\} \mid x \in X\}$. These are also the the join-irreducible elements. Any set $Y \subseteq X$ with $|Y| > 1$ is not irreducible, since $Y = \bigsqcup_{x \in Y} \{x\}$ but clearly $Y \neq \{x\}$ for any $x \in Y$.
- The real interval $[0, 1]$ with the usual order $\leq$. Join and meet are the sup and inf over real numbers, 0 is bottom and 1 is top. The rationals $\mathbb{Q} \cap (0, 1]$ are a basis. There are no irreducible elements (in fact, for any $x \in [0, 1]$ we have that $x = \bigsqcup \{y \mid y < x\}$ and clearly $x \notin \{y \mid y < x\}$).
- Consider the partial order $W = \mathbb{N} \cup \{\omega, a\}$ depicted in Fig. 1. It is easy to see that it is a lattice. All elements are irreducible apart from the bottom 0 and the top ω. For the latter notice that, e.g., $\omega = \bigsqcup \{1, a\}$.

A function $f : L \to L$ is *monotone* if for all $l, l' \in L$, if $l \sqsubseteq l'$ then $f(l) \sqsubseteq f(l')$. By Knaster-Tarski's theorem [Tarski 1955], any monotone function on a complete lattice has a least and a greatest fixpoint, denoted respectively μf and νf, characterised as the meet of all pre-fixpoints respectively the join of all post-fixpoints:

$$\mu f = \bigsqcap \{l \mid f(l) \sqsubseteq l\} \qquad \nu f = \bigsqcup \{l \mid l \sqsubseteq f(l)\}$$

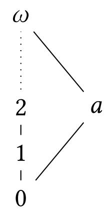


Fig. 1. A complete lattice W which is not continuous.

The least and greatest fixpoint can also be obtained by iterating the function on the bottom and top elements of the lattice. This is often referred to as Kleene's theorem (at least for continuous functions) and it is one of the pillars of abstract interpretation [Cousot and Cousot 1979]. Given a lattice L, define its *height* λ_L as the supremum of the length of any strictly ascending, possibly transfinite, chain. Then we have the following result.

Theorem 2.3 (Kleene's iteration [Cousot and Cousot 1979]). *Let L be a lattice and let $f : L \to L$ be a monotone function. Consider the (transfinite) ascending chain $(f^\beta(\bot))_\beta$ where β ranges over the ordinals, defined by $f^0(\bot) = \bot$, $f^{\alpha+1}(\bot) = f(f^\alpha(\bot))$ for any ordinal α and $f^\alpha(\bot) = \bigsqcup_{\beta<\alpha} f^\beta(\bot)$ for any limit ordinal α. Then $\mu f = f^\gamma(\bot)$ for some ordinal $\gamma \le \lambda_L$. The greatest fixpoint νf can be characterised dually, via the (transfinite) descending chain $(f^\alpha(\top))_\alpha$.*

Note also that $f^\alpha(\bot)$ is always a post-fixpoint and $f^\alpha(\top)$ is always a pre-fixpoint.

We will focus on special lattices where elements are generated by suitably defined approximations. Given a lattice L, a subset $X \subseteq L$ is *directed* if $X \neq \emptyset$ and every pair of elements in X has an upper bound in X.

Definition 2.4 (way-below relation, continuous lattices). Let L be a lattice. Given two elements $l, l' \in L$ we say that l is *way-below* l', written $l \ll l'$ when for every directed set $D \subseteq L$, if $l' \sqsubseteq \bigsqcup D$ then there exists $d \in D$ such that $l \sqsubseteq d$. We denote by $\twoheaddownarrow l$ the set of elements way-below l, i.e., $\twoheaddownarrow l = \{l' \mid l' \in L \,\wedge\, l' \ll l\}$.

The lattice L is called *continuous* if $l = \bigsqcup \twoheaddownarrow l$ for all $l \in L$.

Intuitively, the way-below relation captures a form of finitary approximation: if one imagines that $\sqsubseteq$ is an order on the information content of the elements, then $x \ll y$ means that whenever y can be "covered" by joining (possibly small) pieces of information, then x is covered by one of those pieces. Then a lattice is continuous if any element can be built by joining its approximations.

Concerning the origin of the name "continuous lattice", we can quote [Scott 1972] that says that "One of the justifications (by euphony at least) of the term *continuous lattice* is the fact that such spaces allow for so many continuous functions." For instance, one indication is the fact that meet and join are both continuous in such lattices.

When L is a continuous lattice and B_L is a basis, for all $l \in L$, it holds that $l = \bigsqcup(\twoheaddownarrow l \cap B_L)$.

Various lattices that are commonly used in semantics enjoy a property stronger than continuity, defined below.

Definition 2.5 (compact element, algebraic lattice). Let L be a lattice. An element $l \in L$ is called *compact* whenever $l \ll l$. The lattice L is *algebraic* if the set of compact elements is a basis.

Example 2.6. Some examples are as follows:

- All finite lattices are continuous (since every finite directed set has a maximum). More generally, all algebraic lattices (which include all finite lattices) are continuous. The way-below relation is $x \ll y$ if x compact and $x \sqsubseteq y$.
- Given a set X, the powerset lattice $\mathbf{2}^X$, ordered by inclusion, is an algebraic lattice. The compact elements are the finite subsets. In fact, any set Y is the union of its finite subsets, i.e., $Y = \bigcup\{F \mid F \subseteq Y \ \land \ F \text{ finite}\}$. Since $\{F \mid F \subseteq Y \ \land \ F \text{ finite}\}$ is directed set, compactness requires that $Y \subseteq F$ for some finite $F \subseteq Y$, hence $Y = F$. Therefore $Y \ll Z$ holds when Y is finite and $Y \subseteq Z$.
- The interval $[0, 1]$ with the usual order $\leq$ is a continuous lattice. For $x, y \in [0, 1]$, we have $x \ll y$ when $x < y$ or $x = 0$. In fact, each $\emptyset \neq Y \subseteq [0, 1]$ is directed. Imagine that $y \leq \bigsqcup Y$ for such a Y. Then by standard properties of the reals there always exists a $y' \in Y$ such that $x \leq y'$ if and only if $x < y$ or $x = 0$. Note that this lattice is not algebraic since the only compact element is 0.
- The lattice W in Fig. 1 is not continuous. In fact, $a \not\ll a$ since $a \sqsubseteq \bigsqcup \mathbb{N}$ but $a \not\sqsubseteq i$ for all $i \in \mathbb{N}$. Therefore $\twoheaddownarrow a = \{0\}$ and thus $a \neq \bigsqcup \twoheaddownarrow a$.

2.2 Tuples and Ordinals

We will often consider tuples of elements. Given a set A, an n-tuple in A^n will be denoted by a boldface letter $\boldsymbol{a}$. The components of a tuple $\boldsymbol{a}$ will be denoted by using the same name of the tuple, not in boldface style and with an index, i.e., $\boldsymbol{a} = (a_1, \ldots, a_n)$. For an index $n \in \mathbb{N}$ we use the notation $\underline{n}$ to denote the integer interval $\{1, \ldots, n\}$. Given $\boldsymbol{a} \in A^n$ and $i, j \in \underline{n}$ we write $\boldsymbol{a}_{i,j}$ for the subtuple $(a_i, a_{i+1}, \ldots, a_j)$.

Definition 2.7 (pointwise order). Given a lattice $(L, \sqsubseteq)$ we will denote by $(L^n, \sqsubseteq)$ the set of n-tuples endowed with the *pointwise order* defined, for $\boldsymbol{l}, \boldsymbol{l'} \in L^n$, by $\boldsymbol{l} \sqsubseteq \boldsymbol{l'}$ if $l_i \sqsubseteq l'_i$ for all $i \in \underline{n}$.

The structure $(L^n, \sqsubseteq)$ is a lattice and it is continuous if L is continuous, with the way-below relation given by $\boldsymbol{l} \ll \boldsymbol{l'}$ iff $l_i \ll l'_i$ for all $i \in \underline{n}$ [Gierz et al. 2003, Proposition I-2.1]. More generally, for any set X, the set of functions $L^X = \{f \mid f : X \to L\}$, endowed with pointwise order, is a lattice (continuous when L is).

Definition 2.8 (lexicographic order). Given a partial order $(P, \sqsubseteq)$ we will denote by $(P^n, \preceq)$ the set of n-tuples endowed with the *lexicographic order* (where the last component is the most relevant), i.e., inductively, for $\boldsymbol{l}, \boldsymbol{l'} \in P^n$, we let $\boldsymbol{l} \preceq \boldsymbol{l'}$ if either $l_n \sqsubset l'_n$ or $l_n = l'_n$ and $\boldsymbol{l}_{1,n-1} \preceq \boldsymbol{l'}_{1,n-1}$.

When $(L, \sqsubseteq)$ is a lattice also $(L^n, \preceq)$ is a lattice. Given a set $X \subseteq L^n$, the meet of X with respect to $\preceq$ can be obtained by taking the meet of the single components, from the last to the first, i.e., it is defined inductively as $\bigsqcap X = \boldsymbol{l}$ where $l_i = \bigsqcap\{l'_i \mid \boldsymbol{l'} \in X \ \land \ \boldsymbol{l'}_{i+1,n} = \boldsymbol{l}_{i+1,n}\}$. The join can be defined analogously. Similarly, one can show that $\preceq$ is a well-order whenever $\sqsubseteq$ is.

As in [Hasuo et al. 2016; Jurdziński 2000], we will also need to consider tuples with a preorder arising from the lexicographic order, when some components are considered irrelevant.

Definition 2.9 (truncated lexicographic order). Let $(P, \sqsubseteq)$ be a partial order and let $n \in \mathbb{N}$. For $i \in \underline{n}$ we define a preorder $\preceq_i$ on P^n by letting, for $\boldsymbol{x}, \boldsymbol{y} \in P^n$, $\boldsymbol{x} \preceq_i \boldsymbol{y}$ if $\boldsymbol{x}_{i,n} \preceq \boldsymbol{y}_{i,n}$. We write $=_i$ for the equivalence induced by $\preceq_i$ and $\boldsymbol{x} \prec_i \boldsymbol{y}$ for $\boldsymbol{x} \preceq_i \boldsymbol{y}$ and $\boldsymbol{x} \neq_i \boldsymbol{y}$. Whenever $\sqsubseteq$ is a well-order, given $X \subseteq P^n$ with $X \neq \emptyset$ and $i \in \underline{n}$, we write $\min_{\preceq_i} X$ for the vector $\boldsymbol{x} = (\bot, \ldots, \bot, x_i, \ldots, x_n)$ where $\boldsymbol{x}_{i,n} = \min_{\preceq}\{\boldsymbol{l}_{i,n} \mid \boldsymbol{l} \in X\}$.

In words, $\preceq_i$ is the lexicographic order restricted to the components $i, i+1, \ldots, n$. For instance, if $P = \mathbb{N}$ with the usual order, then $(6, 1, 4, 7) \prec_2 (5, 2, 4, 7)$, while $(6, 1, 4, 7) =_3 (5, 2, 4, 7)$.

We denote *ordinals* by Greek letters $\alpha, \beta, \gamma, \ldots$ and their order by $\leq$. The collection of all ordinals is well-ordered. Given any ordinal α, the collection of ordinals dominated by α is a set $[\alpha] = \{\lambda \mid \lambda \leq \alpha\}$, which, seen as an ordered structure, is a lattice. Meet and join of a set X of ordinals will be denoted by $\inf X$ (which equals $\min X$ if $X \neq \emptyset$) and $\sup X$. From the results discussed above, for a fixed $n \in \mathbb{N}$ and an ordinal α, the n-tuples of ordinals below α, referred to as *ordinal vectors*, endowed with the lexicographic order $([\alpha]^n, \preceq)$, form a lattice.

3 FIXPOINT EQUATIONS: SOLUTIONS AND APPROXIMANTS

In this section we introduce the systems of fixpoint equations we will work with in the paper. We define the solution of a system and we devise some results concerning its approximations that will play a major role later.

3.1 Systems of Fixpoint Equations

We focus on systems of (fixpoint) equations over some lattice, where, for each equation one can be interested either in the least or in the greatest solution.

Definition 3.1 (system of equations). Let L be a lattice. A system of equations E over L is a list of equations of the following form

$$\begin{array}{lll} x_1 & =_{\eta_1} & f_1(x_1, \ldots, x_m) \\ & \ldots & \\ x_m & =_{\eta_m} & f_m(x_1, \ldots, x_m) \end{array}$$

where $f_i \colon L^m \to L$ are monotone functions and $\eta_i \in \{\mu, \nu\}$. The system will often be denoted as $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{x}$, $\boldsymbol{\eta}$ and $\boldsymbol{f}$ are the obvious tuples. We denote by $\emptyset$ the system with no equations.

Systems of equations of this kind have been considered by various authors, e.g., [Cleaveland et al. 1992; Hasuo et al. 2016; Seidl 1996]. In particular, [Hasuo et al. 2016] works on general lattices.

We next define the pre-solutions of a system as tuples of lattice elements that, replacing the variables, satisfy all the equations of the system. The solution will be a suitably chosen pre-solution, taking into account also the η_i annotations that specify for each equation whether the least or greatest solution is required.

Definition 3.2 (pre-solution). Let L be a lattice and let E be a system of equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. A *pre-solution* of E is any tuple $\boldsymbol{u} \in L^m$ such that $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$.

Note that if we view $\boldsymbol{f}$ as a function $\boldsymbol{f} \colon L^m \to L^m$, then pre-solutions are the fixpoints of $\boldsymbol{f}$.

In order to define the solution of a system we need some further notation.

Definition 3.3 (substitution). Given a system E of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, an index $i \in \underline{m}$ and $l \in L$ we write $E[x_i := l]$ for the system of $m-1$ equations obtained from E by removing the i-th equation and replacing x_i by l in the other equations, i.e., if $\boldsymbol{x} = \boldsymbol{x}' x_i \boldsymbol{x}''$, $\boldsymbol{\eta} = \boldsymbol{\eta}' \eta_i \boldsymbol{\eta}''$ and $\boldsymbol{f} = \boldsymbol{f}' f_i \boldsymbol{f}''$ then $E[x_i := l]$ is $\boldsymbol{x}'\boldsymbol{x}'' =_{\boldsymbol{\eta}', \boldsymbol{\eta}''} \boldsymbol{f}'\boldsymbol{f}''(\boldsymbol{x}', l, \boldsymbol{x}'')$.

Let $f[x_i := l] \colon L^{m-1} \to L$ be defined by $f[x_i := l](\boldsymbol{x}', \boldsymbol{x}'') = f(\boldsymbol{x}', l, \boldsymbol{x}'')$. Then, explicitly, the system $E[x_i := l]$ has $m-1$ equations,

$$x_j =_{\eta_j} f_j[x_i := l](\boldsymbol{x}', \boldsymbol{x}'') \qquad j \in \{1, \ldots, i-1, i+1, \ldots, n\}$$

We can now recursively define the solution of a system of equations. The notion is the same as in [Hasuo et al. 2016], although we find it convenient to adopt a more succinct formulation (an explicit proof of the equivalence of the two notions can be found in the full version).

Definition 3.4 (solution). Let L be a lattice and let E be a system of m equations on L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. The *solution* of E, denoted $sol(E) \in L^m$, is defined inductively as follows:

$$\begin{aligned} sol(\emptyset) &= () \\ sol(E) &= (sol(E[x_m := u_m]), u_m) \text{ where } u_m = \eta_m(\lambda x.\, f_m(sol(E[x_m := x]), x)) \end{aligned}$$

The i-th component of the solution will be denoted $sol_i(E)$.

In words, for solving a system of m equations, the last variable is considered as a fixed parameter x and the system of $m-1$ equations that arises from dropping the last equation is recursively solved. This produces an $(m-1)$-tuple parametric on x, i.e., we get $\boldsymbol{u}_{1,m-1}(x) = sol(E[x_m := x])$. Inserting this parametric solution into the last equation, we get an equation in a single variable

$$x =_{\eta_m} f_m(\boldsymbol{u}_{1,m-1}(x), x)$$

that can be solved by taking for the function $\lambda x.\, f_m(\boldsymbol{u}_{1,m-1}(x), x)$, the least or greatest fixpoint, depending on whether the last equation is a μ- or ν-equation. This provides the m-th component of the solution $u_m = \eta_m(\lambda x.\, f_m(\boldsymbol{u}_{1,m-1}(x), x))$. The remaining components of the solution are obtained inserting u_m in the parametric solution $\boldsymbol{u}_{1,m-1}(x)$ previously computed, i.e., $\boldsymbol{u}_{1,m-1} = \boldsymbol{u}_{1,m-1}(u_m)$.

The next lemma will be helpful in several places. In particular, it shows that the definition above is well-given, since we are taking (least or greatest) fixpoints of monotone functions.

Lemma 3.5 (solution is monotone). *Let E be a system of $m > 0$ equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ over a lattice L. For $i \in \underline{m}$ the function $g\colon L \to L^{m-1}$ defined by $g(x) = sol(E[x_i := x])$ is monotone.*

It can be easily proved that the solution of a system is, as anticipated, a special pre-solution.

Lemma 3.6 (solution is pre-solution). *Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{u}$ be its solution. Then $\boldsymbol{u}$ is a pre-solution, i.e., $\boldsymbol{u} = \boldsymbol{f}(\boldsymbol{u})$.*

3.2 A Prototypical Example: The μ-Calculus

As a prototypical example, we discuss how μ-calculus formulae can be equivalently seen as systems of fixpoint equations. We focus on a standard μ-calculus syntax. For fixed disjoint sets *PVar* of propositional variables, ranged over by $x, y, z, \ldots$ and *Prop* of propositional symbols, ranged over by $p, q, r, \ldots$, formulae are defined by

$$\varphi ::= \mathbf{t} \mid \mathbf{f} \mid p \mid x \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box\varphi \mid \Diamond\varphi \mid \eta x.\, \varphi$$

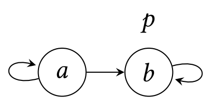
where $p \in \mathit{Prop}$, $x \in \mathit{PVar}$ and $\eta \in \{\mu, \nu\}$. Formulae of the kind $\eta x.\, \varphi$ are called fixpoint formulae.

The semantics of a formula is given with respect to an unlabelled transitions system (or Kripke structure) $(\mathbb{S}, \to)$ where $\mathbb{S}$ is the set of states and $\to \subseteq \mathbb{S} \times \mathbb{S}$ is the transition relation. Given a formula φ and an environment $\rho\colon \mathit{Prop} \cup \mathit{PVar} \to \mathbf{2}^{\mathbb{S}}$ mapping each proposition or propositional variable to the set of states where it holds, we denote by $\|\varphi\|_\rho$ the semantics of φ defined as usual (see, e.g., [Bradfield and Walukiewicz 2018]).

First note that any μ-calculus formula can be expressed in equational form, by inserting an equation for each propositional variable (see also [Cleaveland et al. 1992; Seidl 1996]). The reverse translation is also possible, hence these specification languages are equally expressive. Here, we will only depict the relation via an example, the formal treatment is given in the full version.

$$\begin{aligned} x_1 &=_\mu p \vee \Diamond x_1 \\ x_2 &=_\nu x_1 \wedge \Box x_2 \end{aligned}$$

(a)



(b)

$$\begin{aligned} x_1 &=_\mu \{b\} \cup \Diamond x_1 \\ x_2 &=_\nu x_1 \cap \Box x_2 \end{aligned}$$

(c)

Fig. 2

Example 3.7. Let $\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$ be a formula requiring that from all reachable states there exists a path that eventually reaches a state where p holds. The equational form is quite straightforward and is reported in Fig. 2a. Consider a transition system $(\mathbb{S}, \rightarrow)$ where $\mathbb{S} = \{a, b\}$ and $\rightarrow$ is as depicted in Fig. 2b, with p that holds only on state b. The resulting system of equations on the lattice $\mathbf{2}^{\mathbb{S}}$ is given in Fig. 2c, where $\Diamond, \Box \colon \mathbf{2}^{\mathbb{S}} \rightarrow \mathbf{2}^{\mathbb{S}}$ are defined as $\Diamond(S) = \{s \in \mathbb{S} \mid \exists s' \in \mathbb{S}.(s \rightarrow s' \wedge s' \in S)\}$, $\Box(S) = \{s \in \mathbb{S} \mid \forall s' \in \mathbb{S}.(s \rightarrow s' \Rightarrow s' \in S)\}$ for $S \subseteq \mathbb{S}$.

The solution is $x_1 = x_2 = \mathbb{S}$. In particular, $x_2 = \mathbb{S}$ corresponds to the fact that the formula φ holds in every state.

Example 3.8. Consider the formula $\varphi' = \nu x_2.(\Box x_2 \wedge \mu x_1.((p \wedge \Diamond x_2) \vee \Diamond x_1))$ requiring that from all reachable states there is a path along which p holds infinitely often. The equational form of φ' is:

$$\begin{aligned} x_1 &=_\mu (p \wedge \Diamond x_2) \vee \Diamond x_1 \\ x_2 &=_\nu \Box x_2 \wedge x_1 \end{aligned}$$

On the same transition system of the previous example (Fig. 2b), the solution of the corresponding system is $x_1 = x_2 = \mathbb{S}$. Notice that this time the order of the equations is relevant, while in the previous example it was not. Indeed, if we swap the two equations in the system, the solution becomes $x_1 = x_2 = \emptyset$. In general, the order of the equations is important whenever there is alternation of fixpoints (mutual dependencies between least and greatest fixpoint equations).

3.3 Data-Flow Analysis

In order to give further intuition, we revisit another area where fixpoints play a major role, namely data-flow analysis of programs. One can easily state a program analysis question in this setting as a system of fixpoint equations, based on the flow graph of the program under consideration.

We focus on the well-known *constant propagation analysis* (see, e.g., [Nielson et al. 1999]). Its aim is to show that the value of a variable is always constant at a certain program point, allowing us to optimise the program by replacing the variable by the constant. Consider for instance the while program in Fig. 3a, where variables contain integer values and blocks are numbered in order to easily reference them. The condition for the while loop (block 3) is irrelevant and is hence replaced by `*`. Note that variable `x` always has value 7 in block 4 and hence the assignment in this block could be replaced by `y:=7+y`.

```
[y:=6]1;
[x:=y+1]2;
while [*]3 do
    [y:=x+y]4
od
```

(a)

$$\begin{aligned} \rho_1 &=_\nu \bot \\ \rho_2 &=_\nu \rho_1[\mathsf{y} \mapsto 6] \\ \rho_3 &=_\nu \rho_2[\mathsf{x} \mapsto \rho_2(\mathsf{y}) + 1] \sqcap \rho_4[\mathsf{y} \mapsto \rho_4(\mathsf{x}) + \rho_4(\mathsf{y})] \\ \rho_4 &=_\nu \rho_3 \end{aligned}$$

(b)

Fig. 3. (a) A simple while program and (b) the equation system for the corresponding constant propagation analysis.

Following [Nielson et al. 1999] we analyse such programs by setting up an instance of a monotone framework. In particular we will use the following lattice to record the results of the analysis:

$$L = (\mathbb{Z} \cup \{\bot\})^{Var} \cup \{\top\}$$

where *Var* is the set of variables. That is, a lattice element is either $\top$ or a function $\rho \colon Var \rightarrow \mathbb{Z} \cup \{\bot\}$ that assigns a variable `x` to a value in $\mathbb{Z}$ (if `x` is known to have constant value $\rho(\mathsf{x})$ at this program point) or to $\bot$ (to indicate that x is possibly non-constant). As usual, we are allowed to over-approximate and $\bot$ might be assigned although the value of the variable is actually constant.

The lattice order is defined as follows: two assignments $\rho_1, \rho_2 \colon \mathit{Var} \to \mathbb{Z} \cup \{\bot\}$ are ordered, i.e. $\rho_1 \sqsubseteq \rho_2$, if for each $\mathsf{x} \in \mathit{Var}$ either $\rho_1(\mathsf{x}) = \rho_2(\mathsf{x})$ or $\rho_1(\mathsf{x}) = \bot$. That is, we consider a flat order where $\bot$ is smaller than the integers and the integers themselves are incomparable, and extend it pointwise to functions. Clearly, $\top$ is the largest lattice element and we use some overloading and denote by $\bot$ the function that maps every variable to $\bot$. Note that this order deviates from the usual convention in program analysis which states that smaller values should be more precise than larger values. We do this since our game characterises whether a lattice element is *below* the solution. Since we want to check that the solution is more precise than a given threshold, we have to reverse the order.

Let us write $\rho' = \rho[\mathsf{x} \mapsto z]$ for $z \in \mathbb{Z}$ to denote function update, that is $\rho'(\mathsf{x}) = z$ and $\rho'(\mathsf{y}) = \rho(\mathsf{y})$ for $\mathsf{y} \neq \mathsf{x}$. When $\rho = \top$ we define $\rho[\mathsf{x} \mapsto z] = \top$.

Observe that L is algebraic (and hence continuous). The compact elements are $\top$ and those functions which have finite support, i.e., functions of the kind $\bot[\mathsf{x1} \mapsto z_1, \dots, \mathsf{xn} \mapsto z_n]$ where only finitely many variables are not mapped to $\top$. In particular we can use as a basis the functions $\bot[\mathsf{x} \mapsto z]$ for some $\mathsf{x} \in \mathit{Var}$ and $z \in \mathbb{Z}$. Note also that L is not distributive. For instance if $\rho_i = \bot[\mathsf{x} \mapsto i]$ for $i \in \{1, 2, 3\}$ then $(\rho_1 \sqcap \rho_2) \sqcup \rho_3 = \bot \sqcup \rho_3 = \rho_3$ while $(\rho_1 \sqcup \rho_3) \sqcap (\rho_2 \sqcup \rho_3) = \top \sqcap \top = \top$.

From the program in Fig. 3a we can easily derive the system of fixpoint equations in Fig. 3b, where we use ρ_i to record the lattice value for the entry of block i.

At the beginning, no variable is constant. Then the equation system mimics the control flow of the program. In block 3 we have to take the meet to obtain an analysis result that is less precise than the results coming from block 2 respectively block 4. Furthermore, since precision increases with the order, we are interested in the greatest solution, which means that we have only ν-equations.

The expected solution is $\rho_1 = \bot$, $\rho_2 = \bot[\mathsf{y} \mapsto 6]$, $\rho_3 = \rho_4 = \bot[\mathsf{x} \mapsto 7]$ witnessing that at block 2 variable y has constant value 6 and at blocks 3 and 4 variable x has constant value 7.

3.4 Approximating the Solution

The game theoretical characterisation of the solution of a system of fixpoint equations discussed later will rely on a notion of approximation of the solution that is reminiscent of the lattice progress measure in [Hasuo et al. 2016].

Definition 3.9 (approximants). Let E be a system of m equations over the lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given any tuple $\boldsymbol{l} \in L^m$, let $f_{i,\boldsymbol{l}} \colon L \to L$ be the function defined by

$$f_{i,\boldsymbol{l}}(x) = f_i(sol(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}][x_i := x]), x, \boldsymbol{l}_{i+1,m}).$$

We say that a tuple $\boldsymbol{l} \in L^m$ is a μ-*approximant* when for all $i \in \underline{m}$, if $\eta_i = \nu$ then $l_i = \nu(f_{i,\boldsymbol{l}})$, else if $\eta_i = \mu$ then $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α. Dually, $\boldsymbol{l} \in L^m$ is a ν-*approximant* when for all $i \in \underline{m}$, if $\eta_i = \nu$ then $l_i = f^{\alpha}_{i,\boldsymbol{l}}(\top)$ for some ordinal α, else if $\eta_i = \mu$ then $l_i = \mu(f_{i,\boldsymbol{l}})$.

Whenever $\boldsymbol{l}$ is a μ-approximant we write $ord(\boldsymbol{l})$ to denote the least m-tuple of ordinals $\boldsymbol{\alpha}$ such that for any $i \in \underline{m}$, if $\eta_i = \mu$ then $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\bot)$ else, if $\eta_i = \nu$, $l_i = f^{\alpha_i}_{i,\boldsymbol{l}}(\top) = \nu(f_{i,\boldsymbol{l}})$.

Observe that, spelling out the definition of the solution of a system of equations, it can be easily seen that $sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}]) = \eta_i(f_{i,\boldsymbol{l}})$. Then a μ-approximant is obtained by taking under-approximations for the least fixpoints and the exact value for greatest fixpoints. In fact, in the case of μ-approximants, for each $i \in \underline{m}$, if $\eta_i = \nu$, the i-th component is set to $\nu(f_{i,\boldsymbol{l}})$ which is i-th component $sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$ of the solution. Instead, if $\eta_i = \mu$ the component l_i is set to $f^{\alpha}_{i,\boldsymbol{l}}(\bot)$ for some ordinal α, which is an underapproximation of $\mu(f_{i,\boldsymbol{l}}) = sol_i(E[\boldsymbol{x}_{i+1,m} := \boldsymbol{l}_{i+1,m}])$, obtained by iterating $f_{i,\boldsymbol{l}}$ over $\bot$ up to ordinal α. For ν-approximants the situation is dual.

We remark that the function $f_{i,\boldsymbol{l}}$ depends only on the subvector $\boldsymbol{l}_{i+1,m}$. In particular $f_{m,\boldsymbol{l}}$ does not depend on $\boldsymbol{l}$. In fact, $f_{m,\boldsymbol{l}} = \lambda x.\, f_m(sol(E[x_m := x]), x)$. Using $\boldsymbol{l}$ as subscript instead of the subvector is a slight abuse of notation that makes the notation lighter.

Approximants can be given an inductive characterisation. Besides shedding some light on the notion of approximant, the following easy result will be useful at a technical level.

Lemma 3.10 (inductive characterisation of approximants). *Let E be a system of $m > 0$ equations over the lattice L, of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $g_m \colon L \to L$ be the function $g_m(x) = f_m(sol(E[x_m := x]), x)$. A tuple $\boldsymbol{l} \in L^m$ is a μ-approximant iff the following conditions hold*

(1) either $\eta_m = \mu$ and $l_m = g_m^{\alpha}(\bot)$ for some ordinal α, or $\eta_m = \nu$ and $l_m = \nu g_m$
(2) $\boldsymbol{l}_{1,m-1}$ is a μ-approximant of $E[x_m := l_m]$.

As mentioned above, μ-approximants are closely related to lattice progress measures in the sense of [Hasuo et al. 2016]. In fact, from Lemma 3.10 we can infer that, given a vector $\boldsymbol{\alpha}$ of ordinals, the μ- or ν-approximant $\boldsymbol{l} \in L^m$ with $ord(\boldsymbol{l}) = \boldsymbol{\alpha}$ is uniquely determined. More precisely, a μ-approximant $\boldsymbol{l}$ is determined by the subvector of $ord(\boldsymbol{l})$ consisting only of the m-tuple of components of $ord(\boldsymbol{l})$ corresponding to μ-indices. Hence we can define a function that maps each such m-tuple of ordinals to the corresponding μ-approximant and this turns out to be a lattice progress measures in the sense of [Hasuo et al. 2016]. Actually, as proved in the full version, it is the greatest one. It can be shown to coincide with the measure used in [Hasuo et al. 2016, Theorem 2.13] (completeness part).

We next observe that the name approximant is appropriate, i.e., μ-approximants provide an approximation of the solution from below, while ν-approximants from above. The solution is thus the only pre-solution which is both a μ- and a ν-approximant.

Lemma 3.11 (solution and approximants). *Let E be a system of m equations over the lattice L, of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. The solution of E is the greatest μ-approximant and the least ν-approximant.*

4 FIXPOINT GAMES

In this section we present a game-theoretical approach to the solution of a system of fixpoint equations over a continuous lattice. More precisely, given a lattice with a fixed basis, the game allows us to check whether an element of the basis is smaller (with respect to $\sqsubseteq$) than the solution of a selected equation. This corresponds to solving the associated verification problem. For instance, when model-checking the μ-calculus, one is interested in establishing whether a system satisfies a formula φ, which amounts to check whether $\{s_0\} \subseteq u_\varphi$ where s_0 is the initial state and u_φ is the solution of the system of equations associated with φ.

4.1 Definition of the Game

The fixpoint game that we introduce has been inspired by the unfolding game described in [Venema 2008], that works for a single fixpoint equation over the powerset lattice. We adopted the name *fixpoint game*, analogously to [Hansen et al. 2017].

Definition 4.1 (fixpoint game). Let L be a continuous lattice and let B_L be a basis of L such that $\bot \notin B_L$. Given a system E of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, the corresponding fixpoint game is a parity game, with an existential player $\exists$ and a universal player $\forall$, defined as follows:

- The positions of $\exists$ are pairs (b, i) where $b \in B_L$ and $i \in \underline{m}$ and those of $\forall$ are tuples $\boldsymbol{l} \in L^m$.
- From (b, i) the possible moves of $\exists$ are $\mathbf{E}(b, i) = \{\boldsymbol{l} \mid \boldsymbol{l} \in L^m \ \wedge\ b \sqsubseteq f_i(\boldsymbol{l})\}$.
- From $\boldsymbol{l} \in L^m$ the possible moves of $\forall$ are $\mathbf{A}(\boldsymbol{l}) = \{(b, i) \mid i \in \underline{m} \ \wedge\ b \in B_L \ \wedge\ b \ll l_i\}$.

The game is schematised in Table 1. For a finite play, the winner is the player whose opponent is unable to move. For an infinite play, let h be the highest index that occurs infinitely often in a pair (b, i). If $\eta_h = \nu$ then $\exists$ wins, else $\forall$ wins.

Table 1. The fixpoint game

Position	Player	Moves
(b, i)	$\exists$	$(l_1, \ldots, l_m)$ such that $b \sqsubseteq f_i(l_1, \ldots, l_m)$
$(l_1, \ldots, l_m)$	$\forall$	(b', j) such that $b' \ll l_j$

Observe that the fixpoint game is a parity game [Emerson and Jutla 1991; Zielonka 1998] (on an infinite graph) and the winning condition is the natural formulation of the standard winning condition in this setting.

Hereafter, whenever we consider a continuous lattice L, we assume that a basis B_L is fixed such that $\bot \notin B_L$. Elements of the basis will be denoted by letters b with super or subscripts.

We will prove correctness and completeness of the game, i.e., we will show that if $\boldsymbol{u}$ is the solution of the system, given a basis element $b \in B_L$ and $i \in \underline{m}$, if $b \sqsubseteq u_i$ then starting from (b, i) the existential player has a winning strategy, otherwise the universal player has a winning strategy.

Example 4.2. As an example, consider the equation system of Example 3.7, as depicted in Fig. 2c, corresponding to the μ-calculus formula $\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$. Recall that the lattice is $(\mathbf{2}^{\mathbb{S}}, \subseteq)$ and let us fix as a basis the set of singletons $B_{\mathbf{2}^{\mathbb{S}}} = \{\{a\}, \{b\}\}$.

A portion of the fixpoint game is graphically represented as a parity game in Fig. 4. Diamond nodes correspond to positions of player $\exists$ and the box nodes to positions of player $\forall$. Only a subset of the possible positions for $\forall$ are represented. The positions which are missing, such as $(\{a, b\}, \{a, b\})$, can be shown to be redundant, in a sense formalised later (see § 5.3.1), so that the subgame is equivalent to the full game. Numbers in the diamond nodes correspond to priorities. Box nodes do not have priorities (or we can assume priority 0). Since index 1 and 2 corresponds to a μ and a ν equation, respectively, in this specific case the winning condition for player $\exists$ is exactly the same as for parity games: either the play is finite and $\exists$ plays last or the play is infinite and the highest priority that occurs infinitely often is even (in this case 2).

Let (u_1, u_2) be the solution of the system. We can check if $a \in u_2$, i.e., if a satisfies φ, by playing the game from the position $(\{a\}, 2)$. In fact, $\{a\} \sqsubseteq u_2$ amounts to $a \in u_2$. Indeed player $\exists$ has a winning strategy that we can represent as a function ς from the positions of the game (for any play) to the corresponding moves of player $\exists$, i.e., $\varsigma\colon B_{\mathbf{2}^{\mathbb{S}}} \times \underline{2} \to \mathbf{2}^{\mathbb{S}} \times \mathbf{2}^{\mathbb{S}}$. A winning strategy for $\exists$ is given by $\varsigma(\{a\}, 1) = (\{b\}, \emptyset)$, $\varsigma(\{a\}, 2) = (\{a\}, \{a, b\})$, $\varsigma(\{b\}, 1) = (\emptyset, \emptyset)$ and $\varsigma(\{b\}, 2) = (\{b\}, \{b\})$. In Fig. 4 we depict by bold arrows the choices prescribed by the strategy.

A possible play of the game could be the following, where $\overset{x}{\rightsquigarrow}$ denotes a move of $x \in \{\exists, \forall\}$:

$$(\{a\}, 2) \overset{\exists}{\rightsquigarrow} (\{a\}, \{a, b\}) \overset{\forall}{\rightsquigarrow} (\{a\}, 1) \overset{\exists}{\rightsquigarrow} (\{b\}, \emptyset) \overset{\forall}{\rightsquigarrow} (\{b\}, 1) \overset{\exists}{\rightsquigarrow} (\emptyset, \emptyset) \overset{\forall}{\not\rightsquigarrow},$$

hence $\exists$ wins. Another (infinite) play is the following. It is also won by $\exists$ since the highest index that occurs infinitely often is 2, which is a ν-index:

$$(\{a\}, 2) \overset{\exists}{\rightsquigarrow} (\{a\}, \{a, b\}) \overset{\forall}{\rightsquigarrow} (\{a\}, 2) \overset{\exists}{\rightsquigarrow} (\{a\}, \{a, b\}) \overset{\forall}{\rightsquigarrow} \ldots$$

Note that if $\exists$ always plays as specified by ς, she will always win.

4.2 Correctness and Completeness

Before proving correctness and completeness of the game in the general case, as a warm up, we give some intuition and outline the proof for the case of a single equation. Let $f\colon L \to L$ be a monotone function on a continuous lattice L and consider the equation $x =_\eta f(x)$, where $\eta \in \{\nu, \mu\}$, with solution $u = \eta f$. In this case the positions for $\exists$ are simply basis elements $b \in B_L$ and $\exists$ must

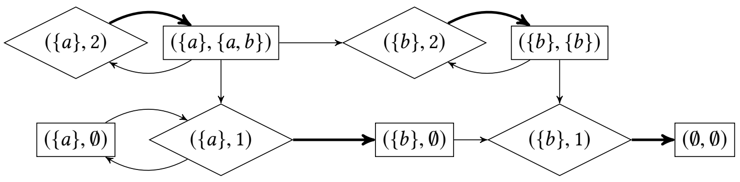


Fig. 4. Graphical representation of a fixpoint game

choose $l \in L$ such that $b \sqsubseteq f(l)$. Positions of $\forall$ are lattice elements $l \in L$ and moves are elements of the basis $b \in B_L$, with $b \ll l$. In the case of $\eta = \mu$, player $\forall$ wins infinite plays and in the case of $\eta = \nu$, player $\exists$ wins infinite plays.

When $\eta = \mu$, if $b \sqsubseteq u$, then $b \sqsubseteq f^{\alpha}(\bot)$ for some ordinal α. The idea is that $\exists$ can win by descending the chain $f^{\beta}(\bot)$. E.g., if $\beta = \gamma + 1$ is a successor ordinal, then $\exists$ can play $f^{\gamma}(\bot)$. If instead, $\eta = \nu$, then the existential player can win just by identifying some post-fixpoint l such that $b \sqsubseteq l$. In fact, if l is a post-fixpoint, i.e., $l \sqsubseteq f(l)$ we know that $l \sqsubseteq u$. Moreover, if $b \sqsubseteq l$ then $b \sqsubseteq f(l)$ and thus $\exists$ can cycle on l and win. More formally:

(Case $\eta = \mu$). In this case $u = f^{\alpha}(\bot)$ for some ordinal α.

- *Completeness:* We show that whenever $b \sqsubseteq f^{\beta}(\bot)$, for some ordinal β (i.e., b is below some μ-approximant), then $\exists$ has a winning strategy, by transfinite induction on β. First observe that $\beta > 0$. In fact, otherwise $b \sqsubseteq f^{0}(\bot) = \bot$, hence $b = \bot$, while $\bot \notin B_L$ by hypothesis. Hence we have two possibilities:
  - If β is a limit ordinal, player $\exists$ plays $l = f^{\beta}(\bot)$, which is a post-fixpoint and hence $b \sqsubseteq f^{\beta}(\bot) \sqsubseteq f(f^{\beta}(\bot))$. Then $\forall$ chooses $b' \ll f^{\beta}(\bot) = \bigsqcup_{\gamma<\beta} f^{\gamma}(\bot)$. Since this is a directed join, by definition of the way-below relation there exists $\gamma < \beta$ with $b' \sqsubseteq f^{\gamma}(\bot)$.
  - If $\beta = \gamma + 1$, $\exists$ plays $l = f^{\gamma}(\bot)$ and $\forall$ chooses $b' \ll f^{\gamma}(\bot)$, hence $b' \sqsubseteq f^{\gamma}(\bot)$.

  Note that $\exists$ always has a move and the answer of $\forall$ is some $b' \sqsubseteq f^{\gamma}(\bot)$, with $\gamma < \beta$, from which there exists a winning strategy for $\exists$ by the inductive hypothesis.
- *Correctness:* We show that whenever $b \not\sqsubseteq u$, player $\forall$ has a winning strategy.
  Observe that a move of $\exists$ will be some l such that $b \sqsubseteq f(l)$. Note that there must be a $b' \ll l$ with $b' \not\sqsubseteq u$. In fact, otherwise, if for all $b' \ll l$ it holds that $b' \sqsubseteq u$, since L is a continuous lattice, we would have $l = \bigsqcup\{b' \mid b' \ll l\} \sqsubseteq u$ and furthermore $b \sqsubseteq f(l) \sqsubseteq f(u) = u$, which is a contradiction.
  Hence $\forall$ can choose such a $b' \ll l$ with $b' \not\sqsubseteq u$ and the game can continue. Then either $\exists$ runs out of moves at some point or we end up in an infinite play. In both cases $\forall$ wins.

(Case $\eta = \nu$). In this case $u = f^{\alpha}(\top)$ for some ordinal α.

- *Completeness:* We show that when $b \sqsubseteq u$, then $\exists$ has a winning strategy. In fact, in this case $\exists$ simply plays $l = u$, which satisfies $b \sqsubseteq u = f(u)$ and $\forall$ answers with some $b \ll u$, hence $b \sqsubseteq u$. The game can thus continue forever, leading to an infinite play which is won by $\exists$.
- *Correctness:* We show that whenever $b \not\sqsubseteq f^{\beta}(\top)$, for some ordinal β (i.e., b is not below some ν-approximant), then $\forall$ has a winning strategy, by transfinite induction on β. First observe that $\beta > 0$. In fact, otherwise $b \not\sqsubseteq f^{0}(\top) = \top$ would be a contradiction. Hence we distinguish two cases:
  - If β is a limit ordinal $b \not\sqsubseteq f^{\beta}(\top) = \bigsqcap_{\gamma<\beta} f^{\gamma}(\top)$, which means that there exists $\gamma < \beta$ such that $b \not\sqsubseteq f^{\gamma}(\top)$.

Now any move of $\exists$ is some l with $b \sqsubseteq f(l)$. Therefore $l \not\sqsubseteq f^{\gamma}(\top)$, since otherwise $b \sqsubseteq f(l) \sqsubseteq f(f^{\gamma}(\top)) = f^{\gamma+1}(\top) \sqsubseteq f^{\beta}(\top)$ (since $\gamma + 1 < \beta$). Hence there must be $b' \ll l$ with $b' \not\sqsubseteq f^{\gamma}(\top)$. Otherwise, as above, if for all $b' \ll l$ we had $b' \sqsubseteq f^{\gamma}(\top)$, then by continuity of the lattice, we would conclude $l = \bigsqcup\{b' \mid b' \ll l\} \sqsubseteq f^{\gamma}(\top)$. Such a b' can be chosen by $\forall$, and the game continues.

– If $\beta = \gamma + 1$ we know that $b \not\sqsubseteq f^{\beta}(\top) = f(f^{\gamma}(\top))$.
Any move of $\exists$ is l with $b \sqsubseteq f(l)$, which as above implies that $l \not\sqsubseteq f^{\gamma}(\top)$ and thus the existence of $b' \ll l$ with $b' \not\sqsubseteq f^{\gamma}(\top)$. The basis element b' is chosen by $\forall$ and the game continues.

Hence $\forall$ always has a move, ending up in $b' \not\sqsubseteq f^{\gamma}(\top)$, from which there exists a winning strategy for $\forall$ by the induction hypothesis.

Observe that cases of a μ- and a ν-equation are not completely symmetric. In the completeness part, for showing that $l \sqsubseteq \nu f$ we use the fact that νf is the greatest post-fixpoint. Instead, for showing that $l \sqsubseteq \mu f$ we use the fact that $l \sqsubseteq f^{\alpha}(\bot)$ for some α and provide a proof that we can descend to $\bot$, similarly to what happens for ranking functions in termination analysis. Note that in order to guarantee that we truly descend, also below limit ordinals, we require that $\forall$ plays b with $b \ll l$. Then we can use the fact that whenever b is way-below a directed join, then it is smaller than one of the elements over which the join is taken. We remark that choosing b with $b \sqsubseteq l$ instead would not be sufficient (see Proposition 4.10). In the correctness part, despite the asymmetry, both proofs use the fact that each element is the join of all elements way-below it, for which it is essential to be in a continuous lattice (see Proposition 4.9). Instead, for completeness, the continuity hypothesis does not play a role.

For the general case, correctness and completeness of the game are proved by relying on the notions of μ- and ν-approximant. We prove the two properties separately. Completeness exploits a result that shows how $\exists$ can play descending along a chain of μ-approximants and, as in the case of a single equation, it can be proved for general lattices, without assuming the continuity hypothesis.

Lemma 4.3 (descending on μ-approximants). *Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For each μ-approximant $\boldsymbol{l} \in L^m$ and $(b, i) \in \mathbf{A}(\boldsymbol{l})$ there exists a μ-approximant $\boldsymbol{l}' \in \mathbf{E}(b, i)$ such that $ord(\boldsymbol{l}) \succeq_i ord(\boldsymbol{l}')$. Moreover, if $\eta_i = \mu$, the i-th component strictly decreases and thus the inequality is strict.*

The previous result allows us to prove that player $\exists$ can always win starting from a μ-approximant. Roughly, relying on Lemma 4.3, we can prove that player $\exists$ can play on μ-approximants in a way that each time the i-th equation is chosen, the ordinal vector associated to the approximant decreases with respect to $\preceq_i$, and it strictly decreases when the i-th equation is a μ-equation. This, together with the fact that the order on ordinals is well-founded, allows one to conclude that either the play is finite and $\exists$ plays last or the highest index on which one can cycle is necessarily the index of a ν-equation. In both cases player $\exists$ wins.

Lemma 4.4 ($\exists$ wins on μ-approximants). *Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{l} \in L^m$ be a μ-approximant. Then in a game starting from $\boldsymbol{l}$ (which is a position of $\forall$) player $\exists$ has a winning strategy.*

Since the solution of a system of equation is a μ-approximant (the greatest one), completeness is an easy corollary of Lemma 4.4.

Corollary 4.5 (completeness). *Let E be a system of m equations over a lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given any μ-approximant $\boldsymbol{l} \in L^m$, $b \in B_L$ and $i \in \underline{m}$, if $b \sqsubseteq l_i$ then $\exists$ has a winning strategy from position (b, i).*

For correctness we rely on a result, dual to Lemma 4.3, that allows to ascend along ν-approximants. However, in this case, the fact of working in a continuous lattice is crucial (see Proposition 4.9).

Lemma 4.6 (ascending on ν-approximants). *Let E be a system of m equations over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given a ν-approximant $\boldsymbol{l} \in L^m$, an element $b \in B_L$ and an index $i \in \underline{m}$ with $b \not\sqsubseteq l_i$, for all tuples $\boldsymbol{l}' \in \mathbf{E}(b, i)$ there are a ν-approximant $\boldsymbol{l}''$ and $(b'', j) \in \mathbf{A}(\boldsymbol{l}')$ such that (1) $b'' \not\sqsubseteq l''_j$ and (2) $ord(\boldsymbol{l}) \succeq_i ord(\boldsymbol{l}'')$. Moreover, if $\eta_i = \nu$, the i-th component strictly decreases and thus the inequality in item 2 above is strict.*

As in the dual case, correctness is an easy corollary of the above lemma, recalling that the solution is the least ν-approximant.

Lemma 4.7 (correctness). *Let E be a system of m equations over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For a ν-approximant $\boldsymbol{l} \in L^m$, $b \in B_L$ and $i \in \underline{m}$, if $b \not\sqsubseteq l_i$ then $\forall$ has a winning strategy from position (b, i).*

Combining Corollary 4.5 and Lemma 4.7 we reach the desired result.

Theorem 4.8 (correctness and completeness). *Given a system of m equations E over a continuous lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$, then for all $b \in B_L$ and $i \in \underline{m}$,*

$$b \sqsubseteq u_i \quad \text{iff} \quad \exists \text{ has a winning strategy from position } (b, i).$$

Note that even when the fixpoint is reached in more than ω steps, thanks to the fact that the order on the ordinals is well-founded and players "descend" over the order, ordinals do not play an explicit role in the game. In particular plays are not transfinite and whenever $\forall$ or $\exists$ win due to the fact that the other player cannot make a move, this happens after a *finite* number of steps. This can be a bit surprising at first since the game works for general continuous lattices, including, for instance, intervals over the reals.

We close this subsection by proving two results that, in a sense, show that the choice of continuous lattices and the design of the game based on the way-below relation are "the right ones". We first observe that the restriction to continuous lattices is not only sufficient but also necessary for the correctness of the game.

Proposition 4.9 (correctness holds exactly in continuous lattices). *Let L be a lattice and let B_L be a fixed basis with $\bot \notin B_L$. The game is correct for every system of equations over L if and only if L is continuous.*

As a counterexample, consider the lattice W in Fig. 1, which is not continuous and let B_W be any basis such that $0 \notin B_W$. First note that necessarily $a \in B_W$, otherwise $a \neq \bigsqcup\{x \in B_W \mid x \sqsubseteq a\} = \bigsqcup \emptyset = 0$. Secondly, $\twoheaddownarrow a = \{0\}$ since $a \not\ll a$. Then, consider the equation $x =_\mu f(x)$, where the function $f \colon W \to W$ is defined by $f(0) = 0$, and $f(x) = \omega$ for $x \neq 0$. Clearly f is monotone and its least fixpoint is $\mu f = 0$. However, the player $\exists$ can win any play of the game from position a, despite the fact that $a \not\sqsubseteq \mu f = 0$. In fact, the first move of $\exists$ can be a, since $a \sqsubseteq f(a) = \omega$. But then player $\forall$ has no moves since $\twoheaddownarrow a \cap B_W = \emptyset$. And so player $\exists$ always wins while she should not.

The second observation is that using the lattice order instead of the way-below relation may break completeness. More precisely, consider the natural variant of the game where the way-below relation is replaced by the lattice order. Let us call it *weak game*. Since the set of possible moves of player $\forall$ is enlarged, correctness clearly continues to hold. Instead, as we hinted before, completeness could fail. We show that it is exactly on algebraic lattices that completeness still holds for the weak game.

Proposition 4.10 (way-below is needed in non-algebraic lattices). *Let L be a lattice. The weak game is complete on every system of equations over L if and only if B_L consists of compact elements (which in turn means that L is algebraic).*

Note that when the elements of the basis are compact, the way-below relation with respect to elements of the basis is the lattice order. Hence the result above essentially states that the weak game is complete exactly when it coincides with the original game, thus further supporting the appropriateness of our formulation of the game.

As a counterexample, consider the continuous lattice $[0, 1]$ with the usual order and basis $B_{[0,1]} = \mathbb{Q} \cap (0, 1]$. Recall that $[0, 1]$ is not algebraic (the only compact element is 0) and way-below relation is the strict order $<$. Let $g\colon [0, 1] \to [0, 1]$ be the function defined by $g(x) = \frac{x+1}{2}$. The fixpoint equation $x =_\mu g(x)$ has solution $\mu g = 1$.

In the weak game, from position $l \in [0, 1]$, player $\forall$ can play any $b \leq l$ (instead, of $b < l$). Then player $\exists$ loses any play starting from position 1, despite the fact that $1 \leq \mu g = 1$. In fact, the only possible move of player $\exists$ is 1, and $\forall$ can play any $x \leq 1$. In particular, playing 1 the game will continue forever and will thus be won by $\forall$.

Notice that, instead, in the original game, from position 1, player $\forall$ has to play an element $1-\epsilon$ for some $\epsilon > 0$. Then, it is easy to see that at each step i player $\exists$ will be able to play some $z_i \leq 1 - 2^i\epsilon$. This means that after finitely many steps $\exists$ will be allowed to play 0, thus leaving no possible answer to $\forall$ and winning the game.

4.3 Relation to μ-Calculus Model-Checking

We discuss how our fixpoint game over systems arising from μ-calculus formulae relates to classical techniques for model-checking the μ-calculus, which can be presented interchangeably in terms of parity games, tableaux, and automata (see, e.g., [Emerson 1985]). Specifically, we compare our game with classical tableau systems for the μ-calculus (e.g., as in [Cleaveland 1990; Stirling and Walker 1991]) where the similarities can be presented more directly.

Recall that a tableau is a (finite) proof tree whose nodes are labelled by sequents. Usually sequents are of the kind $s \models \varphi$, where s is a state of the model and φ formula. The fact that a state s satisfies a formula φ amounts to the existence of a tableau, rooted in $s \models \varphi$ and that it is *successful*, according to a suitable definition.

Given a closed μ-calculus formula φ and a state s in a transition system $(\mathbb{S}, \to)$, let E be the corresponding system of m equations. The model-checking problem using tableaux is solved by searching for a successful tableau for the sequent $s \models \varphi$. Instead, using the fixpoint game, it is reduced to the existence of a winning strategy for player $\exists$ starting from position $(\{s\}, m)$, where m is the highest equation index.

We discuss the two approaches using Example 3.7. Recall that the formula of interest is $\varphi = \nu x_2.((\mu x_1.(p \vee \Diamond x_1)) \wedge \Box x_2)$. Let ψ denote the subformula $\mu x_1.(p \vee \Diamond x_1)$. Using the tableau rules in the style of [Cleaveland 1990] (omitting assumptions for the sake of the presentation), we can build a successful tableau for the sequent $a \models \varphi$ as in Fig. 5a. It is not difficult to see that this tableau corresponds to the winning strategy ς for $\exists$ discussed in Example 4.2. In fact, consider the reduced tree in Fig. 5b, which is obtained from the tableau by keeping only the sequents corresponding to fixpoint formulae (i.e., either φ or ψ) and replacing such formulae with the corresponding variable (φ with x_2 and ψ with x_1).

Each sequent $s \models x_i$ can be seen as a position $(\{s\}, i) \in B_{2^{\mathbb{S}}} \times \underline{2}$ of $\exists$ in the fixpoint game. The successor sequents correspond to the move prescribed on $(\{s\}, i)$ by the strategy ς. More precisely, the move should be (y_1, y_2) where $y_j = \{s' \mid s' \models x_j \text{ is a successor of } s \models x_i\}$ for $j \in \underline{2}$. For instance, the sequent $a \models x_2$ corresponds to the position $(\{a\}, 2)$. The three successors $a \models x_1$, $a \models x_2$ and

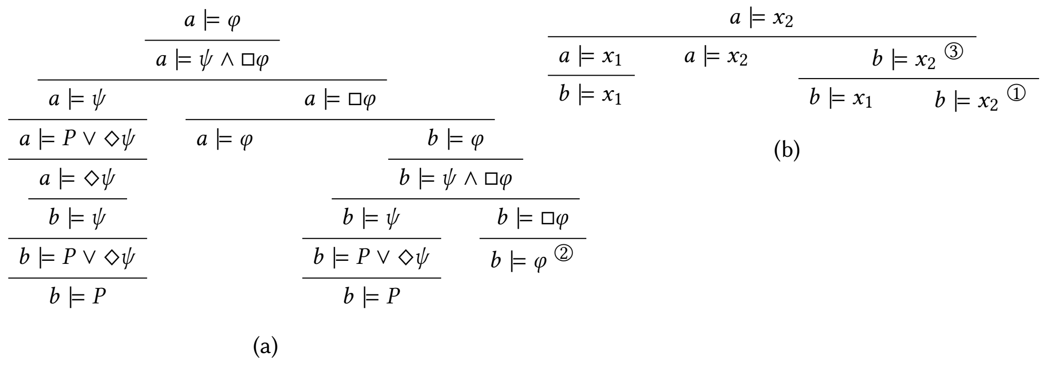


Fig. 5. μ-calculus tableaux vs strategies in the fixpoint game

$b \models x_2$ determine the move prescribed by the strategy $\varsigma(\{a\}, 2) = (\{a\}, \{a, b\})$. Instead, the sequent $a \models x_1$ has only one successor $b \models x_1$ and, correspondingly, we have $\varsigma(\{a\}, 1) = (\{b\}, \emptyset)$, since there are no successors containing variable x_2. When a sequent appears on a leaf of the reduced tree which was already a leaf in the original tableau, by definition of the tableau rules it must have an ancestor labelled by the same sequent and in this case the strategy is defined by the ancestor. For instance, in Fig. 5, the sequent $b \models x_2$ labels the leaf ① which was already a leaf in the original tableau, marked by ②. The strategy is thus defined by the ancestor ③, labelled by the same sequent $b \models x_2$, as $\varsigma(\{b\}, 2) = (\{b\}, \{b\})$.

Additionally, it can be seen that plays of the fixpoint game correspond to paths in the reduced tree. For example, the first play discussed in Example 4.2 corresponds to the leftmost path in the tree. In fact, while successors of sequents define the strategy for player $\exists$, the moves of player $\forall$ determine the path to follow.

For general, possibly non-successful tableaux, if we consider the reduced tree, then for each subtree the sequents at the leaves can be read as a set of assumptions that player $\exists$ has taken to show that the root sequent holds. Player $\forall$ chooses among such assumptions which one player $\exists$ should develop next. If there is no winning strategy for player $\exists$, the winning strategy for player $\forall$ is such that he always chooses a path in the tableau that cannot be successfully concluded at a leaf.

4.4 Fixpoint Games in Data-Flow Analysis

We get back to constant propagation example in § 3.3. Recall that the system of fixpoint equations expressing the analysis in Fig. 3b had solution $\rho_1 = \bot$, $\rho_2 = \bot[\mathsf{y} \mapsto 6]$, $\rho_3 = \rho_4 = \bot[\mathsf{x} \mapsto 7]$. We next describe a game that shows that indeed $\bot[\mathsf{x} \mapsto 7] \sqsubseteq \rho_4$ and hence x has constant value 7 at block 4. The game starts as follows:

$$(\bot[\mathsf{x} \mapsto 7], 4) \overset{\exists}{\rightsquigarrow} (\bot, \bot, \bot[\mathsf{x} \mapsto 7], \bot) \overset{\forall}{\rightsquigarrow} (\bot[\mathsf{x} \mapsto 7], 3) \overset{\exists}{\rightsquigarrow} (\bot, \bot[\mathsf{y} \mapsto 6], \bot, \bot[\mathsf{x} \mapsto 7]) \overset{\forall}{\rightsquigarrow}$$

Now the universal player has two options: either choose $(\bot[\mathsf{x} \mapsto 7], 4)$, which brings him back to an earlier game situation and might potentially lead to an infinite game. Since we are considering greatest fixpoints, this means that $\exists$ wins. If he chooses the other option, the game continues as follows, where eventually $\forall$ has no move left and $\exists$ wins as well:

$$(\bot[\mathsf{y} \mapsto 6], 2) \overset{\exists}{\rightsquigarrow} \bot \overset{\forall}{\not\rightsquigarrow}$$

5 STRATEGIES AS PROGRESS MEASURES

Along the lines of [Jurdziński 2000], influenced by [Hasuo et al. 2016], in this section we introduce a general notion of progress measure for fixpoint games over continuous lattices. We will show how a complete progress measure characterises the winning positions for the two players. The existence of a so-called small progress measure will allow us to express a complete progress measure as a least fixpoint, thus providing a technique for computing the progress measure and solving the corresponding system of equations.

5.1 General Definition

Given an ordinal α we denote by $[\alpha]^m_\star = \{\beta \mid \beta \leq \alpha\}^m \cup \{\star\}$, the set of ordinal vectors with entries smaller or equal than α, with an added bound $\star$.

Definition 5.1 (progress measure). Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given an ordinal λ, a *λ-progress measure* for E is a function $R\colon B_L \to \underline{m} \to [\lambda]^m_\star$ such that for all $b \in B_L$, $i \in \underline{m}$, either $R(b)(i) = \star$ or there exists $\boldsymbol{l} \in \mathbf{E}(b,i)$ such that for all $(b',j) \in \mathbf{A}(\boldsymbol{l})$ it holds

- if $\eta_i = \mu$ then $R(b)(i) \succ_i R(b')(j)$;
- if $\eta_i = \nu$ then $R(b)(i) \succeq_i R(b')(j)$

A progress measure maps any basis element of the lattice and index $i \in \underline{m}$ to an m-tuple of ordinals, with one component for each equation. Components relative to μ-equations roughly measure how many unfolding steps for the equation would be needed to reach an under-approximation l_i above b, and thus, for $\exists$, to win the game. Components relative to ν-equations, as in the original work of [Jurdziński 2000], are less relevant, as we will see.

Intuitively, whenever $R(b)(i) \neq \star$, the progress measure R provides an evidence of the existence of a winning strategy for $\exists$ in a play starting from (b,i). The tuple $\boldsymbol{l}$, whose existence is required by the definition, is a move of player $\exists$ such that for any possible answer of $\forall$, the progress measure will not increase with respect to $\preceq_i$, and it will strictly decrease in the case of μ-equations. Since $\prec_i$ is well-founded, this ensures that we cannot cycle on a μ-equation. Also note that whenever the current index is i, all indices lower than i are irrelevant (expressed by the orders $\succeq_i$ resp. $\succ_i$), which is related to the fact that the highest index which is visited infinitely often is the only relevant index for determining the winner of the game. This idea is formalised in the following lemma.

Lemma 5.2 (progress measures are strategies). *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$. For any $b \in B_L$ and $i \in \underline{m}$, if there exists some ordinal λ and a λ-progress measure R such that $R(b)(i) \preceq_i (\lambda, \ldots, \lambda)$, then $b \sqsubseteq u_i$.*

The above lemma, in a sense, says that progress measures provide sound characterisations of the solution. However, in general, they are not complete, since whenever $R(b)(i) = \star$ we cannot derive any information on (b,i), i.e., if $\boldsymbol{u}$ is the solution of the system, we cannot conclude that $b \not\sqsubseteq u_i$. This motivates the following definition.

Definition 5.3 (complete progress measures). Let L be a continuous lattice and let E be a system of equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{u}$. A λ-progress measure $R\colon B_L \to \underline{m} \to [\lambda]^m_\star$ is called *complete* if for all $b \in B_L$ and $i \in \underline{m}$, if $b \sqsubseteq u_i$ then $R(b)(i) \preceq_i (\lambda, \ldots, \lambda)$.

Observe that in search of a complete progress measure, in principle, we would have to try all ordinals as a bound. We next show that we can take as bound the height λ_L of the lattice L. This provides a generalisation of the *small progress measure* in [Jurdziński 2000].

Definition 5.4 (small progress measure). Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Given an m-tuple of ordinals $\boldsymbol{\alpha}$, let us denote by $z_E(\boldsymbol{\alpha})$ the m-tuple of ordinals where ν-components are set to 0, i.e., $z_E(\boldsymbol{\alpha}) = \boldsymbol{\beta}$ with $\beta_i = \alpha_i$ if $\eta_i = \mu$ and $\beta_i = 0$ otherwise. We define the *small progress measure* $R_E \colon B_L \to \underline{m} \to [\lambda_L]^m_\star$

$$R_E(b)(i) = \min_{\preceq_i}\{z_E(ord(\boldsymbol{l})) \mid \boldsymbol{l} \text{ is a } \mu\text{-approximant } \wedge\ \boldsymbol{l} \in \mathbf{E}(b,i)\}$$

where $\min_{\preceq_i}$ is the minimum on $\preceq_i$ as given in Definition 2.9, with the convention that $\min_{\preceq_i} \emptyset = \star$.

Observe that R_E is well-defined, i.e., it actually takes values in $[\lambda_L]^m_\star$. In fact, the components of $z_E(ord(\boldsymbol{l}))$ corresponding to μ-indices are ordinals expressing the number of Kleene iterations needed to reach under-approximations of the least fixpoint. These are clearly bounded by λ_L, since for a monotone function $f \colon L \to L$, the sequence $f^\alpha(\bot)$ is strictly increasing until it reaches the least fixpoint of f. For ν-indices, $z_E(ord(\boldsymbol{l}))$ is always 0.

Observe that while formally $R_E(b)(i)$ takes values in $[\lambda_L]^m_\star$, whenever $j < i$ or $\eta_j = \nu$, due to the effect of the $\min_{\preceq_i}$ and of the z_E operations, the only possible value for the j-th component is 0. Despite such components are then clearly irrelevant, we keep them for notational convenience.

The fact that R_E is indeed a progress measure follows from Lemma 4.3. Moreover, we can easily show that it is complete.

Lemma 5.5 (small progress measure). *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Then $R_E \colon B_L \to \underline{m} \to [\lambda_L]^m_\star$ is a complete progress measure.*

5.2 Progress Measures as Fixpoints

Here we show that a complete progress measure can be characterised as the least solution of a system of equations over tuples of ordinals, naturally induced by Definition 5.1.

Definition 5.6 (progress measure equations). Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Let $\boldsymbol{\delta}^\eta_i$, with $i \in \underline{m}$, be, for $\eta = \nu$, the null vector and, for $\eta = \mu$, the vector such that $\delta_j = 0$ if $j \neq i$ and $\delta_i = 1$. The *progress measure equations* for E over the lattice $[\lambda_L]^m_\star$, are defined, for $b \in B_L$, $i \in \underline{m}$, as:

$$R(b)(i) = \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b,i)\}$$

We will denote by Φ_E the corresponding endofunction on $L \to \underline{m} \to [\lambda_L]^m_\star$ which is defined, for $R \colon B_L \to \underline{m} \to [\lambda_L]^m_\star$, by $\Phi_E(R)(b)(i) = \min_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid (b',j) \in \mathbf{A}(\boldsymbol{l})\} \mid \boldsymbol{l} \in \mathbf{E}(b,i)\}$

Observe that, since $[\lambda_L]^m_\star$ is a lattice, also the corresponding set of progress measures, endowed with pointwise $\preceq$-order, is a lattice. It is immediate to see that Φ_E is monotone with respect to such order, i.e., if $R \preceq R'$ pointwise then $\Phi_E(R) \preceq \Phi_E(R')$ pointwise. This allows us to obtain a complete progress measure as a (least) fixpoint of Φ_E.

Lemma 5.7 (complete progress measure as a fixpoint). *Let L be a continuous lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. Then the least solution R_M of the progress measure equations (least fixpoint of Φ_E with respect to $\preceq$) is the least λ_L-progress measure, hence it is smaller than R_E and it is complete.*

Observe that, since $R_M \preceq R_E$, in particular, for all $b \in B_L$ and $i \in \underline{m}$, if $R_M(b)(i) \neq \star$, then all components of $R_M(b)(i)$ corresponding to ν-indices are 0.

Example 5.8. If we consider the system of equations of Example 3.7 we obtain as least fixpoint the progress measure $R_M(\{a\})(1) = (1,0)$ while $R_M(\{a\})(2) = R_M(\{b\})(1) = R_M(\{b\})(2) = (0,0)$. Note that R_M never assumes the top value $\star$, consistently with the fact that the solution is $(u_1,u_2) = (\mathbb{S},\mathbb{S})$. We will discuss how R_M is obtained later when providing a more "efficient" way for computing it.

We next observe that the operator Φ_E creates functions $R\colon B_L \to \underline{m} \to [\lambda_L]^m_\star$ which are monotone, with respect to the pointwise order on $\underline{m} \to [\lambda_L]^m_\star$. Explicitly, $R\colon B_L \to \underline{m} \to [\lambda_L]^m_\star$ is monotone if for all $b, b' \in L$ and $i \in \underline{m}$, if $b \sqsubseteq b'$ then $R(b)(i) \preceq R(b')(i)$.

Lemma 5.9 ($\Phi_E(R)$ is monotone). *Let L be a lattice and let E be a system of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. For every function and $R\colon B_L \to \underline{m} \to [\lambda_L]^m_\star$, the function $\Phi_E(R)$ is monotone.*

5.3 Computing Progress Measures

5.3.1 Selections. In principle, at least on finite lattices, the previous results allow one to compute the progress measure and thus to prove properties of the solutions of the system of equations. However, the computation can be quite inefficient due to the fact that the existential player has a (uselessly) large number of possible moves. In fact, given a system $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ on a lattice L, from a position (b, i), given any move $\boldsymbol{l} \in \mathbf{E}(b, i)$ for player $\exists$, i.e., any tuple such that $b \sqsubseteq f_i(\boldsymbol{l})$, it is immediate to see that all $\boldsymbol{l}'$ such that $\boldsymbol{l} \sqsubseteq \boldsymbol{l}'$ are valid moves for $\exists$, since by monotonicity of f_i we have $b \sqsubseteq f_i(\boldsymbol{l}) \sqsubseteq f_i(\boldsymbol{l}')$. In other words, $\mathbf{E}(b, i)$ is upward-closed. However, player $\exists$, in order to win, has to try to descend as much as possible, hence playing large elements is inconvenient.

We next introduce some machinery that formalises the above intuition and allows us to make the calculation more efficient. The idea is discussed for a single function first, and then for a system of equations. For this we need some additional notation. Given a monotone function $f\colon L^m \to L$ and $b \in B_L$, we write $\mathbf{E}(b, f) = \{\boldsymbol{l} \mid \boldsymbol{l} \in L^m \ \wedge\ b \sqsubseteq f(\boldsymbol{l})\}$.

Definition 5.10 (selection). Let L be a lattice. Given a monotone function $f\colon L^m \to L$, a *selection for f* is a function $\sigma\colon B_L \to \mathbf{2}^{L^m}$ such that for all $b \in B_L$ it holds $\mathbf{E}(b, f) = \uparrow\sigma(b)$. Given a system E of m equations on L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, a *selection for E* is an m-tuple of functions $\boldsymbol{\sigma}$ such that, for each $i \in \underline{m}$, the function σ_i is a selection for f_i.

Intuitively, a selection provides for each element of the basis and function f_i, a subset of the moves $\mathbf{E}(b, i)$ that are sufficient to "cover" b in all possible ways. Indeed, we can show that when computing the complete progress measure R_M according to the equations in Lemma 5.7, we can restrict the moves of the existential player to a selection. Dually, since the moves of the universal player $\mathbf{A}(\boldsymbol{l})$ are downward-closed and the progress measures of interest are monotone (see Lemma 5.9), we can restrict also such moves to a subset whose downward-closure is $\mathbf{A}(\boldsymbol{l})$.

Lemma 5.11 (progress measure on a selection). *Let L be a continuous lattice, let E be a system of equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ and let $\boldsymbol{\sigma}$ be a selection for E. Moreover, for all $\boldsymbol{l} \in L^m$ let $\mathbf{A}_r(\boldsymbol{l}) \subseteq B_L \times \underline{m}$ be such that $\mathbf{A}(\boldsymbol{l}) = \{(b', i) \mid (b, i) \in \mathbf{A}_r(\boldsymbol{l}) \ \wedge\ b' \sqsubseteq b\}$. The system of equations over the lattice $[\lambda_L]^m_\star$, defined, for $b \in L$, $i \in \underline{m}$, as:*

$$R(b)(i) = \min\nolimits_{\preceq_i}\{\sup\{R(b')(j) + \boldsymbol{\delta}_i^{\eta_i} \mid (b', j) \in \mathbf{A}_r(\boldsymbol{l})\} \mid \boldsymbol{l} \in \sigma_i(b)\}$$

has the same least solution as that in Lemma 5.7.

Since the complete progress measure R_M witnesses the existence of a winning strategy for $\exists$, the above result implies that whenever $\exists$ has a winning strategy, it has one also in the game where the moves of $\exists$ are restricted to be in the selection. A similar property holds for $\forall$ and $\mathbf{A}_r(\boldsymbol{l})$.

Clearly, for computational purposes, we are interested in having the selections as small as possible. Given a monotone function $f\colon L^m \to L$, and two selections $\sigma, \sigma'\colon B_L \to \mathbf{2}^{L^m}$ for f, we write $\sigma \subseteq \sigma'$ if for all $b \in B_L$ it holds $\sigma(b) \subseteq \sigma'(b)$. We will use the same notation for the pointwise order on selections for systems of equations.

Example 5.12 (selections for μ-calculus operators). Given a transition system $(\mathbb{S}, \to)$, consider the powerset lattice $\mathbf{2}^{\mathbb{S}}$ ordered by subset inclusion, with basis $B_{\mathbf{2}^{\mathbb{S}}} = \{\{s\} \mid s \in \mathbb{S}\}$. Then standard μ-calculus operators admit a least selection, as detailed below.

- Given $f\colon (2^{\mathbb{S}})^2 \to 2^{\mathbb{S}}$ defined by $f(X_1, X_2) = X_1 \cup X_2$, then $\sigma\colon B_{2^{\mathbb{S}}} \to 2^{(2^{\mathbb{S}})^2}$ is $\sigma(\{s\}) = \{(\emptyset, \{s\}), (\{s\}, \emptyset)\}$
- Given $f\colon (2^{\mathbb{S}})^2 \to 2^{\mathbb{S}}$ defined by $f(X_1, X_2) = X_1 \cap X_2$, then $\sigma\colon B_{2^{\mathbb{S}}} \to 2^{(2^{\mathbb{S}})^2}$ is $\sigma(\{s\}) = \{(\{s\}, \{s\})\}$
- Given $f\colon 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ defined by $f(X) = \Diamond X$, then $\sigma\colon B_{2^{\mathbb{S}}} \to 2^{2^{\mathbb{S}}}$ is $\sigma(\{s\}) = \{\{s'\} \mid s \to s'\}$
- Given $f\colon 2^{\mathbb{S}} \to 2^{\mathbb{S}}$ defined by $f(X) = \Box X$, then $\sigma\colon B_{2^{\mathbb{S}}} \to 2^{2^{\mathbb{S}}}$ is $\sigma(\{s\}) = \{\{s' \mid s \to s'\}\}$

We next provide sufficient conditions for a function to admit a least selection.

Lemma 5.13 (existence of least selections). *Let L be a lattice with a basis B_L and let $f\colon L^m \to L$ be a monotone function. If f preserves the meet of descending chains, then it admits a least selection σ_m that maps each $b \in B_L$ to the set of minimal elements of $\mathbf{E}(b, f)$.*

Example 5.14. Consider our running example in Example 3.7. Minimal selections for the functions f_1 and f_2 associated with the first and second equation are given by

- $\sigma_1(\{a\}) = \{(\{a\}, \emptyset), (\{b\}, \emptyset)\}$ and $\sigma_1(\{b\}) = \{(\emptyset, \emptyset)\}$;
- $\sigma_2(\{a\}) = \{(\{a\}, \{a, b\})\}$ and $\sigma_2(\{b\}) = \{(\{b\}, \{b\})\}$.

Observe that the winning strategy for $\exists$ discussed in Example 4.2 is a subset of the selection. This is a general fact: if a winning strategy exists, we can find one that is a subset of any given selection.

Selections can be constructed "compositionally", i.e., if a function f arises as the composition of some component functions then we can derive a selection for f from selections of the components. More details can be found in the full version of the paper.

5.3.2 *A Logic for Characterising the Moves of the Existential Player.* The set of possible moves of the existential player is an upward-closed set in the lattice. Such sets can be conveniently represented and manipulated in logical form (see, e.g., [Delzanno and Raskin 2000]). Intuitively, (minimal) selections describe a disjunctive normal form, but more compact representations can be obtained using arbitrary nesting of conjunction and disjunction. For instance, the minimal selection for the monotone function $f(X_1, \ldots, X_{2n}) = (X_1 \cup X_2) \cap (X_3 \cup X_4) \cap \cdots \cap (X_{2n-1} \cup X_{2n})$ would be of exponential size (think of the corresponding disjunctive normal form), but we can easily give a formula of linear size.

This motivates the introduction of a propositional logic for expressing the set of moves of the existential player along with a technique for deriving the fixpoint equations for computing the progress measure, avoiding the potential exponential explosion.

Definition 5.15 (logic for upward-closed sets). Let L be a continuous lattice and let B_L be a basis for L. Given $m \in \mathbb{N}$, the logic $\mathcal{L}_m(B_L)$ has formulae defined as follows, where $b \in B_L$ and $j \in \underline{m}$:

$$\varphi ::= [b, j] \mid \bigvee_{k \in K} \varphi_k \mid \bigwedge_{k \in K} \varphi_k$$

We will write *true* for the empty conjunction. The semantics of a formula φ is an upward-closed set $[\![\varphi]\!] \subseteq L^m$, defined as follows:

$$[\![[b, j]]\!] = \{\boldsymbol{l} \in L^m \mid b \sqsubseteq l_j\} \qquad [\![\bigvee_{k \in K} \varphi_k]\!] = \bigcup_{k \in K} [\![\varphi_k]\!] \qquad [\![\bigwedge_{k \in K} \varphi_k]\!] = \bigcap_{k \in K} [\![\varphi_k]\!]$$

It is easy to see that indeed each upward-closed set is denoted by a formula, showing that the logic is sufficiently expressive.

Lemma 5.16 (formulae for upward-closed sets). *Let L be a continuous lattice with basis B_L and let $X \subseteq L^m$ be upward-closed. Then $X = [\![\varphi]\!]$ where φ is the formula in $\mathcal{L}_m(B_L)$ defined as follows:*

$$\varphi = \bigvee_{\boldsymbol{l} \in X} \bigwedge \{[b, j] \mid j \in \underline{m} \ \wedge \ b \sqsubseteq l_j\}.$$

For practical purposes we should restrict to finite formulae. This can surely be done in the case of finite lattices, but also for well-quasi orders (see, e.g., [Delzanno and Raskin 2000]).

Definition 5.17 (symbolic $\exists$-moves). Let L be a continuous lattice and let $f \colon L^m \to L$ be a monotone function. A *symbolic $\exists$-move* for f is a family $(\varphi_b)_{b \in B_L}$ of formulae in $\mathcal{L}_m(B_L)$ such that $[\![\varphi_b]\!] = \mathbf{E}(b, f)$ for all $b \in B_L$.

If E is a system of m equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ over a continuous lattice L, a *symbolic $\exists$-move* for E is a family of formulae $(\varphi^i_b)_{b \in B_L, i \in \underline{m}}$ such that for all $i \in \underline{m}$, the family $(\varphi^i_b)_{b \in B_L}$ is a symbolic $\exists$-move for f_i.

Interestingly, symbolic $\exists$-moves can be obtained compositionally, namely, the formulae corresponding to a functions arising as a composition can be obtained from those of the components.

Lemma 5.18 (symbolic $\exists$-moves, compositionally). *Let L be a continuous lattice with a basis B_L, and let $f \colon L^n \to L$, $f_j \colon L^m \to L$ for $j \in \underline{n}$ be monotone functions and let $(\varphi_b)_{b \in B_L}$, $(\varphi^j_b)_{b \in B_L}$, $j \in \underline{n}$ be symbolic $\exists$-moves for $f, f_1, \ldots, f_n$. Consider the function $h \colon L^m \to L$ obtained as the composition $h(\boldsymbol{l}) = f(f_1(\boldsymbol{l}), \ldots, f_n(\boldsymbol{l}))$. Define $(\varphi'_b)_{b \in B_L}$ as follows. For all $b \in B_L$, the formula φ'_b is obtained from φ_b by replacing each occurrence of $[b', j]$ by $\varphi^j_{b'}$. Then $(\varphi'_b)_{b \in B_L}$ is a symbolic $\exists$-move for h.*

Example 5.19. Consider again our running example in Example 3.7. The selections specified in Example 5.14 can be expressed in the logic as follows:

$$\varphi^1_{\{a\}} = [\{a\}, 1] \vee [\{b\}, 1] \qquad \varphi^1_{\{b\}} = \mathit{true}$$
$$\varphi^2_{\{a\}} = [\{a\}, 1] \wedge [\{a\}, 2] \wedge [\{b\}, 2] \qquad \varphi^2_{\{b\}} = [\{b\}, 1] \wedge [\{b\}, 2]$$

These formulae can be obtained compositionally. For instance the formula $\varphi^2_{\{a\}}$ for the equation $x_2 =_\nu x_1 \cap \Box x_2$ is obtained by combining a logical formula for x_1 (namely $[\{a\}, 1]$) via conjunction with a logical formula for $\Box x_2$ (namely $[\{a\}, 2] \wedge [\{b\}, 2]$).

A symbolic $\exists$-move for a system can be directly converted into a recipe for evaluating the fixpoint expressions for progress measures. Essentially, every disjunction simply has to be replaced by a minimum and every conjunction by a supremum (although the proof, which relies on complete distributivity of the lattice $[\lambda_L]^m_\star$ is not trivial). Furthermore, in the case of an algebraic lattice, where we can ensure that the elements of the basis are compact, an atom translates to a straightforward lookup of the progress measure without additional computation.

Proposition 5.20 (progress measure from symbolic $\exists$-moves). *Let E be a system of m equations over a continuous lattice L and let B_L be a basis for L. Let $(\varphi^i_b)_{b \in B_L, i \in \underline{m}}$ be a symbolic $\exists$-move for E.*

Then the system of fixpoint equations for computing the progress measure can be written, for all $b \in B_L$ and $i \in \underline{m}$, as $R(b)(i) = R^i_{\varphi^i_b}$ where R^i_ψ is defined inductively as follows:

$$R^i_{[b,j]} = \min\nolimits_{\preceq_i} \{\sup\{R(b')(j) + \boldsymbol{\delta}^{\eta_i}_i \mid b' \ll b\}\} \qquad R^i_{\bigvee_{k \in K} \varphi_k} = \min_{k \in K} R^i_{\varphi_k} \qquad R^i_{\bigwedge_{k \in K} \varphi_k} = \sup_{k \in K} R^i_{\varphi_k}$$

Whenever the basis element b is compact it holds that $R^i_{[b,j]} = \min_{\preceq_i}\{R(b)(j) + \boldsymbol{\delta}^{\eta_i}_i\}$.

Note that the operator $\min_{\preceq_i}$ in the definition of $R^i_{[b,j]}$ above is just there to ensure that all entries in positions smaller than i are set to 0.

Example 5.21. Using the logical formulae from Example 5.19, we obtain the following equations for the progress measure (where $\max_{\leq_i}$ works analogously to $\min_{\leq_i}$: it sets all vector entries in positions smaller than i to 0):

$$R(\{a\})(1) = \min_{\leq_1}\{R(\{a\})(1) + (1,0), R(\{b\})(1) + (1,0)\} \quad R(\{b\})(1) = (0,0)$$
$$R(\{a\})(2) = \max_{\leq_2}\{R(\{a\})(1), R(\{a\})(2), R(\{b\})(2)\} \qquad R(\{b\})(2) = \max_{\leq_2}\{R(\{b\})(1), R(\{b\})(2)\}$$

The solution for the progress measure equations has already been given in Example 5.8.

5.3.3 Complexity Analysis. The benefit of the progress measures introduced in [Jurdziński 2000] is to ensure that model-checking is polynomial in the number of states and exponential in (half of) the alternation depth. We will now perform a corresponding complexity analysis for our setting, based on symbolic $\exists$-moves and by assuming that we are working on a finite lattice.

Let E be a fixed system of m equations over a finite lattice L, let k be the number of μ-equations and let B_L be a basis for L. Let $(\varphi_b^i)_{b \in B_L, i \in \underline{m}}$ be a symbolic $\exists$-move for E and assume that the size of every such formula is bounded by s. Note that the formulae are typically of moderate size. For instance, μ-calculus model-checking, the branching of a transition system (i.e., the number of successors of a single state) is a determining factor. In fact, as it can be grasped from our running example (see Example 5.19), the size of the symbolic $\exists$-move φ_b^i will be linear in the number of propositional operators and, in the presence of modal operators, linear in the branching degree of the transition system. For arbitrary monotone functions it is more difficult to give a general rule.

The shape of the formulae in the symbolic $\exists$-move determine how the values of the progress measure at various positions (b, i) of the games are interrelated. These dependencies clearly play a role in the computation and thus are made explicit by following definition.

Definition 5.22 (dependency graph). Given two game positions $(b, i), (b', j) \in B_L \times \underline{m}$ of $\exists$ we say that (b, i) is a *predecessor* of (b', j) if $[b', j]$ occurs in φ_b^i. We will write $pred(b', j)$ for the set of predecessors of (b', j). In this situation we will also call the pair $((b, i), (b', j))$ an *edge* and refer to corresponding graph as the *dependency graph* for E.

As a first step we provide a bound to the number of edges in the dependency graph.

Proposition 5.23 (edges in the dependency graph). *The number e of edges in the dependency graph for system E is such that $e \leq \min\{|B_L| \cdot m \cdot s, (|B_L| \cdot m)^2\}$, where m is the number of equations and s is the bound on the size of symbolic $\exists$-moves.*

In order to bound the complexity of the overall computation of the progress measure, first note that the lattice $[\lambda_L]_\star^m$ contains $(\lambda_L + 1)^m + 1$ elements. However only $h = (\lambda_L + 1)^k + 1$ are relevant, since the entries of ν-indices are always set to 0. As an example, when model-checking a μ-calculus formula over a finite state system, λ_L is the size of the state space of the Kripke structure. In fact, the lattice is $(\mathbf{2}^{\mathbb{S}}, \subseteq)$ where $\mathbb{S} = \{s_0, \ldots, s_n\}$ is the state space, then the longest ascending chain is $\emptyset \subseteq \{s_0\} \subseteq \{s_0, s_1\} \subseteq \ldots \subseteq \mathbb{S}$.

This fact and the observation that we can perform the fixpoint iteration for the progress measure using a worklist algorithm on the dependency graph, lead to the following result.

Theorem 5.24 (computing progress measures). *The time complexity for computing the least fixpoint progress measure for system E is $O(s \cdot k \cdot e \cdot h)$, where s is the bound on the size of symbolic $\exists$-moves, k is the number of μ-equations, e the number of edges in the dependecy graph, and $h = (\lambda_L + 1)^k + 1$.*

We compare the above with the runtime in [Jurdziński 2000], which is $O(dg(\frac{n}{d})^{\lceil \frac{d}{2} \rceil})$, where d is the alternation depth of the formula, g the number of edges and n the number of nodes of the parity game.

The correspondence is as follows: g corresponds to our number e and n to λ_L (where we cannot exploit the optimisation by Jurdziński which uses the fact that every node in the parity game is associated with a single parity, leading to the division by d). Furthermore s is a new factor, which is due to the fact that we are working with arbitrary functions. But this is mitigated by the fact that we often obtain smaller parity games than in the standard μ-calculus case (see for instance Example 4.2, Figure 4). The number $\frac{d}{2}$ corresponds to our k. However $\frac{d}{2}$ could be strictly lower than k, since we did not take into account the fact that some equations might not be dependent on other equations.

To incorporate this and possibly further optimisations into the complexity analysis we need a notion of alternation depth for equation systems. This can be easily obtained by extending the one introduced in [Cleaveland et al. 1992; Schneider 2004]. A system of equations can be split into closed subsystems corresponding to the strongly connected components of the dependency graph for the system. Then the alternation depth of the system is defined as the length of the longest chain of mutually dependent μ and ν-equations within a closed subsystem. By solving every component separately we obtain a more efficient algorithm.

In particular, systems of fixpoint equations that consist only of μ-equations or ν-equations can be solved by a single fixpoint iteration on L^m, where m is the number of equations [Venema 2008]. Similarly, equations with indices $i, i+1$ where $\eta_i = \eta_{i+1}$ can be merged. This results in an equation system where subsequent equations alternate between μ and ν. (Notice that this transformation means that the equations are over L^j instead of L, but this can be easily adapted in our setting.)

Note also that the runtime might be substantially improved by finding a good strategy for computing the progress measure, as spelled out in [Jurdziński 2000], in the same way as efficient ways can be found for implementing the worklist algorithm in program analysis.

6 MODEL-CHECKING LATTICED μ-CALCULI

As explained earlier, model-checking for μ-calculus formulae can be reduced to solving fixpoint equations over the powerset lattice $2^{\mathbb{S}}$ where $\mathbb{S}$ is the state space of the system under consideration. A state $x \in \mathbb{S}$ can either satisfy or not satisfy a formula, meaning it either belongs to the solution or not. However, there are also multi-valued logics for modelling uncertainty, disagreement or relative importance in which it is natural to have "non-binary" truth values (see, e.g., [Eleftheriou et al. 2012; Fitting 1991; Grumberg et al. 2005; Kupfermann and Lustig 2007]). Such a setting, as detailed later, can also be used to model and verify conditional (or featured) transition systems with upgrades. Here we discuss latticed μ-calculi, inspired by the work cited above, and discuss a corresponding model checking procedure.

A lattice of truth values L is fixed, which is typically finite. and then formulae are evaluated over the lattice $L^{\mathbb{S}}$, endowed with the pointwise order. Also transitions are associated with an element in the lattice of truth values.

Definition 6.1 (multi-valued transition system). A *multi-valued transition system* over L is a function $R\colon \mathbb{S} \times \mathbb{S} \to L$, where $\mathbb{S}$ is the set of states.

Since L can be non-boolean, multi-valued modal logics express forms of negation or implication by relying on *residuation* or *relative pseudo-complement* operation which is well defined for all complete lattices L.

Definition 6.2 (residuation). Let L be a lattice. Given $l, m \in L$, we define $(l \Rightarrow m) = \bigsqcup\{l' \in L \mid l \sqcap l' \sqsubseteq m\}$.

Latticed μ-calculi use atoms, conjunction, disjunction and residuation. The modal operators $\Diamond$ and $\Box$ are interpreted as follows. Given $u \in L^{\mathbb{S}}$ we define $\Diamond u, \Box u \in L^{\mathbb{S}}$ as

$$(\Diamond u)(x) = \bigsqcup_{y \in \mathbb{S}} (R(x,y) \sqcap u(y)) \qquad\qquad (\Box u)(x) = \bigsqcap_{y \in \mathbb{S}} (R(x,y) \Rightarrow u(y))$$

The approach discussed in § 3.2 for model-checking the μ-calculus can be easily adapted to this setting. Instead of the powerset lattice we now have $L^{\mathbb{S}}$ and, as a basis $B_{L^{\mathbb{S}}}$ we can take the functions $b_x \in B_{L^{\mathbb{S}}}$, with $x \in \mathbb{S}$, $b \in B_L$, defined by $b_x(x) = b$ and $b_x(y) = \bot$ for all $y \neq x$.

In order to perform the calculation of the progress measure efficiently, we use symbolic $\exists$-moves as defined in § 5.3.2. Here we assume that L is a finite distributive lattice. In this case $\ll$ and $\sqsubseteq$ coincide. Moreover, for finite distributive lattice it is is well-known from the Birkhoff duality (see also [Davey and Priestley 2002]) that every element can be uniquely represented as the join of a downward-closed set of join-irreducibles. Note that if B_L is the set of join-irreducibles in L, then the basis $B_{L^{\mathbb{S}}} = \{b_x \mid x \in \mathbb{S}, b \in B_L\}$, given above is the set of join-irreducibles of $L^{\mathbb{S}}$.

Proposition 6.3 (symbolic $\exists$-moves in latticed μ-calculi). *Let L be a finite distributive lattice, let B_L be the set of its join-irreducibles. The following are symbolic $\exists$-moves for the semantic functions:*

- *For $\sqcup \colon L^{\mathbb{S}} \times L^{\mathbb{S}} \to L^{\mathbb{S}}$, we let $\psi^{\sqcup}_{b_x} = [b_x, 1] \vee [b_x, 2]$.*
- *For $\sqcap \colon L^{\mathbb{S}} \times L^{\mathbb{S}} \to L^{\mathbb{S}}$, we let $\psi^{\sqcap}_{b_x} = [b_x, 1] \wedge [b_x, 2]$.*
- *For $l \Rightarrow \_ \colon L^{\mathbb{S}} \to L^{\mathbb{S}}$ (where $l \in L$ is fixed and seen as a constant function $\mathbb{S} \to L$), we let $\psi^{\Rightarrow}_{b_x} = \bigwedge\{[b'_x, 1] \mid b' \sqsubseteq l \ \wedge \ b' \sqsubseteq b\}$.*
- *For $\Diamond \colon L^{\mathbb{S}} \to L^{\mathbb{S}}$ we let $\psi^{\Diamond}_{b_x} = \bigvee\{[b_y, 1] \mid y \in Y \ \wedge b \sqsubseteq R(x,y)\}$*
- *For $\Box \colon L^{\mathbb{S}} \to L^{\mathbb{S}}$ we let $\psi^{\Box}_{b_x} = \bigwedge\{[b'_y, 1] \mid y \in Y \ \wedge \ b' \sqsubseteq R(x,y) \ \wedge b' \sqsubseteq b\}$.*

Note that residuation is only monotone in the second argument and that distributivity is essential for this definition of symbolic $\exists$-moves. For instance, if b is not a join-irreducible then $b \sqsubseteq l_1 \sqcup l_2$ is not equivalent to $b \sqsubseteq l_1 \vee b \sqsubseteq l_2$.

Example 6.4 (conditional transition systems with upgrades). An interesting special case are conditional transition systems with upgrades [Beohar et al. 2017] for which a logic satisfying the Hennessy-Milner property has been studied in [Poltermann 2017]. This logic uses the operators given above, enriched with constants. This kind of systems extend the well-known featured transition systems for modelling software product lines [Cordy et al. 2012] by upgrades.

Let $(P, \leq)$ be a given partial order where P is the set of products and $\leq$ is the upgrade relation. If $p \leq q$, it is possible to make an upgrade from q to p during the runtime of the system, i.e., p is the more advanced product compared to q. We consider the lattice $L = (\mathcal{O}(P), \sqsubseteq)$, where $\mathcal{O}(P)$ is the set of all downward-closed subsets of P. (In fact the sets $\downarrow p$, for $p \in P$, where $\downarrow$ denotes downward-closure, are the join-irreducibles of L.) A transition system that compactly specifies the system behaviour for all possible products is given by $R \colon \mathbb{S} \times \mathbb{S} \to \mathcal{O}(P)$ where $p \in R(x,y)$ means that there exists a transition from x to y if one is in possession of product p. More advanced products lead to additional transitions, due to the downward-closure. It is possible to spontaneously perform upgrades during runtime.

Now one can study the latticed modal logic or latticed μ-calculus arising in such a setting. Evaluating a formula φ yields a function $\|\varphi\| \colon \mathbb{S} \to \mathcal{O}(P)$ which intuitively gives us for every state those products on which φ holds (taking upgrades into account).

The approach outlined in the first part of the section can be directly used for model checking the Hennessy-Milner logic on product lines. Note that, as it happens in this case, the lattice L of truth values can have a considerable size and thus the availability of general approaches for handling latticed μ-calculi can be of great help.

7 CONCLUSION

Related work. Our work is based on lattice theory and in particular on continuous lattices. The use of lattices in program analysis and verification has been pioneered by the work [Cousot and Cousot 1977]. Continuous lattices, which received this name due to their intimate connection with continuous functions, have originally been studied by Scott as a semantic domain for the λ-calculus [Scott 1972] and have since found many further applications in the semantics of programming languages [Abramsky and Jung 1994; Gierz et al. 2003].

The modal μ-calculus is an expressive temporal logics, which originated in an unpublished manuscript by Scott and de Bakker and was further developed by Kozen [Kozen 1983]. For a good overview see [Bradfield and Walukiewicz 2018]. Its introduction posed the problem of efficient model-checking, which involves the solution of nested fixpoint equations, see, e.g., [Browne et al. 1997; Cleaveland et al. 1992; Seidl 1996]. The paper [Cleaveland et al. 1992] introduced the notion of a hierarchical system of fixpoint equations, on which our paper is based as well. One way to tackle the model-checking problem is to translate it into the question of finding winning strategies for parity games as first described in [Emerson and Jutla 1991]. A very satisfying technique for solving parity games was proposed in [Jurdziński 2000] resulting in an algorithm which is exponential only in half of the alternation depth. The approach crucially relies on the notion of progress measure, that can be seen as generalising both invariants and ranking functions. The complexity of computing progress measures has recently been improved to quasi-polynomial [Calude et al. 2017].

An extension to general lattices has been given in [Hasuo et al. 2016], which was very inspiring for our development. Compared to [Hasuo et al. 2016] we brought games back into the picture by introducing a game that generalises both parity games and the unfolding games in [Venema 2008]. This allowed us to define a notion of progress measures which is closer to the original definition of Jurdziński and, as such, admits a constructive characterisation as a least fixed point. This works in the general context of continuous lattices, providing a way of solving systems of fixpoint equations in settings that are beyond powerset lattices and were not covered by previous work. We devised the notion of selection and a logics for specifying the moves of the existential player, with the aim of making the computation of progress measures more efficient. We view as a valuable contribution the identification of continuous lattices as the right setting where these general results can be stated.

Usually, μ-calculus formulae are evaluated over the state space of a transition system, i.e., over a powerset lattice. This changes if the μ-calculus is not a classical logic, but lattice-valued as in [Kupfermann and Lustig 2007] or real-valued as in [Huth and Kwiatkowska 1997], which presents an algorithm based on the simplex method for the non-nested case. Solving equation systems over the reals was considered in [Gawlitza and Seidl 2011] and in [Mio and Simpson 2015, 2017]. In particular, [Mio and Simpson 2017] presents an algorithm for solving nested fixpoint equation systems over the interval $[0, 1]$ by a direct algorithm which represents and manipulates piecewise linear functions as conditioned linear expressions. Our results can offer an alternative way to solve such equation systems.

Games for quantitative or probabilistic μ-calculi have been studied in [McIver and Morgan 2007; Mio 2012]. As opposed to our game, such games closely follow the structure of the μ-calculus formula on which the game is based (e.g., $\exists$ makes a choice at an $\vee$-node, $\forall$ at an $\wedge$-node). It is an interesting question whether the conceptual simplicity of our game can lead to a new perspective on existing games.

Future work. A parity game over a finite graph can be easily converted into a system of boolean equations whose solution characterises the winning positions for the players. Since our game is a standard parity game, possibly played on an infinite graph, the standard conversion would lead to infinitely many equations. Systems of equations of this kind are considered, e.g., in [Mader 1997]. An interesting question is under which conditions an infinite parity game can be converted into finitely many equations on an (infinite) powerset lattice.

When solving systems of fixpoint equations over infinite lattices, in particular lattices of infinite height, one typically faces the problem that the progress measure could not be computed in finite time. This is due to the fact that the fixpoints might only be reached after at least ω steps. Hence a symbolic representation is required in order to solve a fixpoint equation system over such lattices. We already took some steps in this direction. The starting observation is that the existence of a winning strategy for player $\exists$ in the game can be expressed as a first-order formula with nested quantifiers (existential quantifiers for the $\exists$ player, universal quantifiers for the $\forall$ player). Usually such a formula is infinite in size, but the main idea is that it can be made finite by adding a stopping condition if an equation index is visited for the second time (without any higher index in between). We have to make sure that the lattice value reached at this point is smaller or equal than the one obtained at the earlier corresponding position. For μ-indices one additionally has to add a so-called *decrease*-predicate in order to ensure that we will truly decrease and eventually reach $\perp$. Such *decrease*-predicates – which have to satisfy a well-foundedness condition – are straightforward for well-founded orders (one checks for strict inequality), but more elaborate for non-well-founded orders such as the one on the reals. Using well-founded predicates one can easily show correctness, but so far we have only partial results for completeness.

Of particular interest are equations over the real interval $[0, 1]$, as considered also in [Mio and Simpson 2017] as a precursor to model-checking PCTL or probabilistic μ-calculi. We have successfully experimented with simple equation systems, involving both linear and non-linear arithmetic, and used the resulting formulas, stating the existence of a winning condition for the existential player, as input for an SMT solver (such as z3 or cvc4). These case studies are contained in the complete fragment and the SMT solver computes the correct solution.

We refer the reader to [Baldan et al. 2018] for more details on the treatment of systems of fixpoint equations over infinite lattices and, in particular, over the reals.

We also plan to study fixpoint equations on the (non-distributive, but continuous) lattices of equivalence relations and pseudo-metrics. As explained in the introduction, the computation of fixpoints for equivalence relations is essential for behavioural equivalences, and the same holds for pseudo-metrics and behavioural distances [van Breugel and Worrell 2005].

We would also like to determine whether we can handle quantitative logics whose modalities interact with (lattice) truth values in a non-trivial way, such as logics with discounted modalities as studied in [Almagor et al. 2014]. Expressing such logics as systems of fixed point equations over suitable continuous lattices and thus obtaining a game theoretical characterisation of the model checking problem seems reasonably easy. However, turning such characterisation into an effective technique requires some non-trivial symbolic approach due to the fact that the lattice is infinite.

Furthermore we would like to study situations in which local (or on-the-fly) algorithms rather than global fixpoint iteration can be used to check whether a lattice element is below the solution. Examples of such local algorithms are backtracking methods studied in [Hirschkoff 1998; Stevens and Stirling 1998]. In particular we are interested in the integration of local methods with up-to techniques for general lattices, see for instance [Bonchi et al. 2018; Pous and Sangiorgi 2011].

ACKNOWLEDGMENTS

Research is partially supported by DFG project BEMEGA and University of Padova project ASTA.

We want to thank the reviewers for their insightful and valuable comments. Furthermore we are grateful to Clemens Kupke for bringing fixpoint games to our attention.

REFERENCES

Samson Abramsky and Achim Jung. 1994. Domain Theory. In *Handbook of Logic in Computer Science*, Samson Abramsky, Dov Gabbay, and Thomas Stephen Edward Maibaum (Eds.). Oxford University Press, 1–168.

Shaull Almagor, Udi Boker, and Orna Kupferman. 2014. Discounting in LTL. In *Proc. of TACAS '14 (Lecture Notes in Computer Science)*, Vol. 8413. Springer, 424–439.

Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. 2018. Fixpoint Games on Continuous Lattices. https://arxiv.org/abs/1810.11404 arXiv:1810.11404.

Harsh Beohar, Barbara König, Sebastian Küpper, and Alexandra Silva. 2017. Conditional Transition Systems with Upgrades. In *Proc. of TASE '17*. IEEE Xplore, 1–8.

Filippo Bonchi, Pierre Ganty, Roberto Giacobazzi, and Dusko Pavlovic. 2018. Sound up-to techniques and Complete abstract domains. In *Proc. of LICS '18*. ACM, 175–184.

Julian Bradfield and Igor Walukiewicz. 2018. The mu-Calculus and Model Checking. In *Handbook of Model Checking*, Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). Springer, 871–919.

Anca Browne, Edmund M. Clarke, Somesh Jha, David E. Long, and Wilfredo R. Marrero. 1997. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science* 178, 1–2 (1997), 237–255.

Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. 2017. Deciding parity games in quasipolynomial time. In *Proc. of STOC '17*. ACM, 252–263.

Rance Cleaveland. 1990. Tableau-based model checking in the propositional mu-calculus. *Acta Informatica* 27, 8 (1990), 725–747.

Rance Cleaveland, Marion Klein, and Bernhard Steffen. 1992. Faster model checking for the modal Mu-Calculus. In *Proc. of CAV 1992 (Lecture Notes in Computer Science)*, Vol. 663. Springer, 410–422.

Maxime Cordy, Andreas Classen, Gilles Perrouin, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. 2012. Simulation-based abstractions for software product-line model checking. In *Proc. of ICSE '12*. IEEE, 672–682.

Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proc. of POPL '77 (Los Angeles, California)*. ACM, 238–252.

Radhia Cousot and Patrick Cousot. 1979. Constructive versions of Tarski's fixed point theorems. *Pacific J. Math.* 82, 1 (1979), 43–57.

Brian A. Davey and Hilary A. Priestley. 2002. *Introduction to lattices and order.* Cambridge University Press.

Giorgio Delzanno and Jean-François Raskin. 2000. Symbolic Representation of Upward-Closed Sets. In *Proc. of TACAS '00 (Lecture Notes in Computer Science)*, Susanne Graf and Michael I. Schwartzbach (Eds.), Vol. 1785. Springer, 426–440.

Pantelis E. Eleftheriou, Costas D. Koutras, and Christos Nomikos. 2012. Notions of bisimulation for Heyting-valued modal languages. *Journal of Logic and Computation* 22 (2012), 213–235.

E. Allen Emerson. 1985. Automata, tableaux, and temporal logics. In *Proceedings of Logics of Programs 1985 (Lecture Notes in Computer Science)*, R. Parikh (Ed.), Vol. 193. Springer, 79–88.

E. Allen Emerson and Charanjit S. Jutla. 1991. Tree automata, Mu-Calculus and determinacy. In *Proc. of SFCS '91*. IEEE, 368–377.

Melvin Fitting. 1991. Many-valued modal logics. *Fundamenta Informaticae* 15 (1991), 235–254.

Gaëlle Fontaine. 2008. Continuous Fragment of the μ-Calculus. In *Proc. of CSL '08 (Lecture Notes in Computer Science)*, Vol. 5213. Springer, 139–153.

Thomas Martin Gawlitza and Helmut Seidl. 2011. Solving systems of rational equations through strategy iteration. *ACM Trans. Program. Lang. Syst.* 33, 3 (2011), 11:1–11:48.

Gerhard Gierz, Karl H. Hofmann, Klaus Keimel, Jimmie D. Lawson, Michael W. Mislove, and Dana S. Scott. 2003. *Continuous Lattices and Domains.* Cambridge University Press.

Orna Grumberg, Martin Lange, Martin Leucker, and Sharon Shoham. 2005. Don't Know in the μ-Calculus. In *Proc. of VMCAI '05 (Lecture Notes in Computer Science)*, Radhia Cousot (Ed.), Vol. 3385. Springer, 233–249.

Helle Hvid Hansen, Clemens Kupke, Johannes Marti, and Yde Venema. 2017. Parity Games and Automata for Game Logic. In *Proc. of DALI '17 (Lecture Notes in Computer Science)*, Vol. 10669. Springer, 115–132.

Ichiro Hasuo, Shunsuke Shimizu, and Corina Cîrstea. 2016. Lattice-theoretic progress measures and coalgebraic model checking. In *Proc. of POPL '16*. ACM, 718–732.

Daniel Hirschkoff. 1998. Automatically Proving Up To Bisimulation. In *Proc. of MFCS '98 Workshop on Concurrency (Electronic Notes in Theoretical Computer Science)*. Elsevier, 75–89.

Michael Huth and Marta Kwiatkowska. 1997. Quantitative analysis and model checking. In *Proc. of LICS '97*. IEEE, 111–122.

Marcin Jurdziński. 2000. Small Progress Measures for Solving Parity Games. In *Proc. of STACS '00 (Lecture Notes in Computer Science)*, Vol. 1770. Springer, 290–301.

Dexter Kozen. 1983. Results on the Propositional μ-Calculus. *Theoretical Computer Science* 27, 3 (1983), 333–354.

Orna Kupfermann and Yoad Lustig. 2007. Latticed Simulation Relations and Games. In *Proc. of ATVA '07 (Lecture Notes in Computer Science)*, Vol. 4672. Springer, 316–330.

Angelika Mader. 1997. *Verification of Modal Properties Using Boolean Equation Systems.* Ph.D. Dissertation. TU München.

Annabelle McIver and Carroll Morgan. 2007. Results on the quantitative μ-calculus qMμ. *ACM Trans. Comp. Log.* 8, 1:3 (2007), 43.

Matteo Mio. 2012. On the Equivalence of Game and Denotational Semantics for the Probabilistic μ-Calculus. *Logical Methods in Computer Science* 8, 2:07 (2012), 1–21.

Matteo Mio and Alex Simpson. 2015. Łukasiewicz μ-calculus. https://arxiv.org/abs/1510.00797 arXiv:1510.00797.

Matteo Mio and Alex Simpson. 2017. Łukasiewicz μ-calculus. *Fundamenta Informaticae* 150, 3-4 (2017), 317–346.

Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. 1999. *Principles of Program Analysis.* Springer.

Katja Poltermann. 2017. *A Modal Logic for Conditional Transition Systems.* Bachelor's thesis. Universität Duisburg-Essen.

Damien Pous and Davide Sangiorgi. 2011. Enhancements of the bisimulation proof method. In *Advanced Topics in Bisimulation and Coinduction*, Davide Sangiorgi and Jan Rutten (Eds.). Cambridge University Press.

Davide Sangiorgi. 2011. *Introduction to Bisimulation and Coinduction.* Cambridge University Press.

Klaus Schneider. 2004. *Verification of Reactive Systems: Formal Methods and Algorithms.* Springer.

Dana Scott. 1972. Continuous lattices. In *Toposes, Algebraic Geometry and Logic (Lecture Notes in Mathematics)*, F. W. Lawvere (Ed.). Springer, 97–136.

Helmut Seidl. 1996. Fast and simple nested fixpoints. *Inform. Process. Lett.* 59, 6 (1996), 303–308.

Perdita Stevens and Colin Stirling. 1998. Practical Model-Checking Using Games. In *Proc. of TACAS '98 (Lecture Notes in Computer Science)*, Vol. 1384. Springer, 85–101.

Colin Stirling. 1995. Local Model Checking Games. In *Proc. of CONCUR '95 (Lecture Notes in Computer Science)*, Vol. 962. Springer, 1–11.

Colin Stirling and David Walker. 1991. Local Model Checking in the Modal mu-Calculus. *Theoretical Computer Science* 89, 1 (1991), 161–177.

Alfred Tarski. 1955. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.* 5 (1955), 285–309.

Franck van Breugel and James Worrell. 2005. A behavioural pseudometric for probabilistic transition systems. *Theoretical Computer Science* 331 (2005), 115–142.

Yde Venema. 2008. Lectures on the modal μ-calculus. Lecture notes, Institute for Logic, Language and Computation, University of Amsterdam.

Wieslaw Zielonka. 1998. Infinite Games on Finitely Coloured Graphs with Applications to Automata on Infinite Trees. *Theoretical Computer Science* 200, 1-2 (1998), 135–183.