Abstraction, Up-To Techniques and Games for Systems of Fixpoint Equations

Paolo Baldan
Università Padova, Italy
baldan@math.unipd.it

Barbara König
Universität Duisburg-Essen, Germany
barbara_koenig@uni-due.de

Tommaso Padoan
Università di Padova, Italy
padoan@math.unipd.it

Abstract

Systems of fixpoint equations over complete lattices, consisting of (mixed) least and greatest fixpoint equations, allow one to express many verification tasks such as model-checking of various kinds of specification logics or the check of coinductive behavioural equivalences. In this paper we develop a theory of approximation for systems of fixpoint equations in the style of abstract interpretation: a system over some concrete domain is abstracted to a system in a suitable abstract domain, with conditions ensuring that the abstract solution represents a sound/complete overapproximation of the concrete solution. Interestingly, up-to techniques, a classical approach used in coinductive settings to obtain easier or feasible proofs, can be interpreted as abstractions in a way that they naturally fit into our framework and extend to systems of equations. Additionally, relying on the approximation theory, we can characterise the solution of systems of fixpoint equations over complete lattices in terms of a suitable parity game, generalising some recent work that was restricted to continuous lattices. The game view opens the way for the development of local algorithms for characterising the solution of such equation systems and we explore some special cases.

2012 ACM Subject Classification Theory of computation → Verification by model checking; Software and its engineering → Model checking

Keywords and phrases fixpoint equation systems, complete lattices, parity games, abstract interpretation, up-to techniques, μ-calculus, bisimilarity

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.25

Related Version A full version of the paper is available as [3], `https://arxiv.org/abs/2003.08877`.

Funding Supported by the PRIN Project Analysis of Program Analyses (ASPRA), the University of Padova project ASTA and the DFG projects BEMEGA and SpeQt.

1 Introduction

Systems of fixpoint equations over complete lattices, consisting of (mixed) least and greatest fixpoint equations, allow one to uniformly express many verification tasks. Notable examples come from the area of model-checking. Invariant/safety properties can be characterised as greatest fixpoints, while liveness/reachability properties as least fixpoints. Using both least and greatest fixpoints leads to expressive specification logics. The μ-calculus [27] is a prototypical example, encompassing various other logics such as LTL and CTL. Another area of special interest for the present paper is that of behavioural equivalences, which typically arise as solutions of greatest fixpoint equations (see, e.g., [38]).

In the first part of the paper we develop a theory of approximation for systems of equations in the style of abstract interpretation. The general idea of abstract interpretation [13, 14] consists of extracting properties of programs by defining an approximated program semantics over a so-called abstract domain, usually a complete lattice. Concrete and abstract semantics are typically expressed in terms of (systems of) least fixpoint equations, with conditions ensuring that the approximation obtained is sound, i.e., that properties derived from the abstract semantics are also valid at the concrete level. In an ideal situation also the converse holds and the abstract interpretation is called complete (see e.g. [18]). Abstract interpretation has been applied also for the model checking of various kinds of mu-calculi and temporal logics (see, e.g., [19, 30, 15, 40, 17, 28]).

We generalise this idea to systems of fixpoint equations, where least and greatest fixpoints can coexist (§4). A system over some concrete domain C is abstracted by a system over some abstract domain A. Suitable conditions are identified that ensure the soundness and completeness of the approximation. This enables the use of the approximation theory on a number of verification tasks. We show how to recover some results on property preserving abstractions for the μ-calculus [30]. We also discuss a fixpoint extension of Łukasiewicz logic, considered in [34] as a precursor to model-checking PCTL or probabilistic μ-calculi.

When dealing with greatest fixpoints, a key proof technique relies on the coinduction principle, which uses the fact that a monotone function f over a complete lattice has a greatest fixpoint νf, which is the join of all post-fixpoints, i.e., the elements l such that $l \sqsubseteq f(l)$. As a consequence proving $l \sqsubseteq f(l)$ suffices to conclude that $l \sqsubseteq \nu f$.

In this setting, up-to techniques have been proposed for "simplifying" proofs [32, 39, 37, 35]. and for reducing the search space in verification (e.g., in [8], up-to techniques applied to language equivalence of NFAs are shown to provide in many cases an exponential speed-up). A sound up-to function is a function u on the lattice such that $\nu(f \circ u) \sqsubseteq \nu f$ and hence $l \sqsubseteq f(u(l))$ implies $l \sqsubseteq \nu(f \circ u) \sqsubseteq \nu f$. The characteristics of u (typically, extensiveness) make it easier to show that an element is a post-fixpoint of $f \circ u$ rather than a post-fixpoint of f.

We show that up-to techniques admit a natural interpretation as abstractions in our framework (§5). This allows us to generalise the theory of up-to techniques to systems of fixpoint equations and contributes to the understanding of the relation between abstract interpretation and up-to techniques, a theme that received some recent attention [6].

We have recently shown in [2] that the solution of systems of fixpoint equations can be characterised in terms of a parity game when working in a suitable subclass of complete lattices, the so-called continuous lattices [41]. Here, relying on our approximation theory, we get rid of continuity and design a game that works for general complete lattices (§6.1).

The above results open the way to the development of game-theoretical algorithms, possibly integrating abstraction and up-to techniques, for solving systems of equations over complete lattices. While global algorithms deciding the game at all positions, based on progress measures [25], have already been studied in [20, 2], here we focus on local algorithms, confining the attention to specific positions. Taking inspiration from backtracking methods for bisimilarity [21] and for the μ-calculus [45, 44], we design a local (also called on-the-fly) algorithm for the case of a single equation (§6.2) (general systems are dealt with in [3]). This also establishes a link with some recent work relating abstract interpretation and up-to techniques [6] and exploiting up-to techniques for language equivalence on NFAs [8].

2 Preliminaries and Notation

A preordered or partially ordered set $\langle P, \sqsubseteq \rangle$ is often denoted simply as P, omitting the (pre)order relation. Given $X \subseteq P$, we denote by $\downarrow X = \{p \in P \mid \exists x \in X.\ p \sqsubseteq x\}$ the *downward-closure* and by $\uparrow X = \{p \in P \mid \exists x \in X.\ x \sqsubseteq p\}$ the *upward-closure* of X. The *join* and the *meet* of a subset $X \subseteq P$ (if they exist) are denoted $\bigsqcup X$ and $\bigsqcap X$, respectively.

▶ **Definition 1** (complete lattice, basis). *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 $\top = \bigsqcap \emptyset$. A* basis *for a complete lattice is a subset $B_L \subseteq L$ such that for each $l \in L$ it holds that $l = \bigsqcup(\downarrow l \cap B_L)$.*

For instance, the powerset of any set X, ordered by subset inclusion $(\mathbf{2}^X, \subseteq)$ is a complete lattice. Join is union, meet is intersection, top ($\top$) is X and bottom ($\bot$) is $\emptyset$. A basis is the set of singletons $B_{\mathbf{2}^X} = \{\{x\} \mid x \in X\}$. Another complete lattice used in the paper is the real interval $[0, 1]$ with the usual order $\leq$. Join and meet are the sup and inf over the reals, 0 is bottom and 1 is top. Any dense subset, e.g., the set of rationals $\mathbb{Q} \cap (0, 1]$, is a basis.

A function $f \colon L \to L$ is *monotone* if for all $l, l' \in L$, if $l \sqsubseteq l'$ then $f(l) \sqsubseteq f(l')$. By Knaster-Tarski's theorem [46, Theorem 1], any monotone function f on a complete lattice has a least fixpoint arising as the meet of all pre-fixpoints $\mu f = \bigsqcap\{l \mid f(l) \sqsubseteq l\}$ and a greatest fixpoint arising as the join of all post-fixpoints $\nu f = \bigsqcup\{l \mid l \sqsubseteq f(l)\}$.

Given a complete 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. If L, L' are complete lattices, a function $f : L \to L'$ is *(directed-)continuous* if for any directed set $X \subseteq L$ it holds that $f(\bigsqcup X) = \bigsqcup f(X)$. The function f is called *strict* if $f(\bot) = \bot$. *Co-continuity* and *co-strictness* are defined dually.

▶ **Definition 2** (Galois connection). *Let $(C, \sqsubseteq)$, $(A, \leq)$ be complete lattices. A* Galois connection *(or* adjunction*) is a pair of monotone functions $\langle \alpha, \gamma \rangle$ such that $\alpha : C \to A$, $\gamma : A \to C$ and for all $a \in A$ and $c \in C$ it holds that $\alpha(c) \leq a$ iff $c \sqsubseteq \gamma(a)$.*

Equivalently, for all $a \in A$ and $c \in C$, (i) $c \sqsubseteq \gamma(\alpha(c))$ and (ii) $\alpha(\gamma(a)) \leq a$. In this case we will write $\langle \alpha, \gamma \rangle : C \to A$. The Galois connection is called an insertion *when $\alpha \circ \gamma = id_A$.*

For a Galois connection $\langle \alpha, \gamma \rangle : C \to A$, the function α is called the left (or lower) adjoint and γ the right (or upper) adjoint. The left adjoint α preserves all joins and the right adjoint γ preserves all meets. Hence, in particular, the left adjoint is strict and continuous, while the right adjoint is co-strict and co-continuous.

A function $f : L \to L$ is *idempotent* if $f \circ f = f$ and *extensive* if $l \sqsubseteq f(l)$ for all $l \in L$. When f is monotone, extensive and idempotent it is called an *(upper) closure*. In this case, $\langle f, i \rangle : L \to f(L)$, where i is the inclusion, is an insertion and $f(L) = \{f(l) \mid l \in L\}$ is a complete lattice.

We will often consider tuples of elements. Given a set A, an n-tuple in A^n is denoted by a boldface letter $\boldsymbol{a}$ and its components are denoted as $\boldsymbol{a} = (a_1, \ldots, a_n)$. For an index $n \in \mathbb{N}$ we write $\underline{n}$ for 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 $(a_i, a_{i+1}, \ldots, a_j)$. The empty tuple is denoted by $()$. Given two tuples $\boldsymbol{a} \in A^m$ and $\boldsymbol{a'} \in A^n$ we denote by $(\boldsymbol{a}, \boldsymbol{a'})$ or simply by $\boldsymbol{a}\boldsymbol{a'}$ their concatenation in A^{m+n}.

Given a complete 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 complete lattice. More generally, for any set X, the set of functions $L^X = \{f \mid f \colon X \to L\}$, endowed with pointwise order, is a complete lattice.

A tuple of functions $\boldsymbol{f} = (f_1, \ldots, f_m)$ with $f_i : X \to Y$, will be seen itself as a function $\boldsymbol{f} : X \to Y^m$, defined by $\boldsymbol{f}(x) = (f_1(x), \ldots, f_m(x))$. We will also need to consider the *product function* $\boldsymbol{f}^\times : X^m \to Y^m$, defined by $\boldsymbol{f}^\times(x_1, \ldots, x_m) = (f_1(x_1), \ldots, f_m(x_m))$.

3 Systems of Fixpoint Equations over Complete Lattices

We deal with systems of (fixpoint) equations over some complete lattice, where, for each equation one can be interested either in the least or in the greatest solution. We define systems, their solutions and we provide some examples that will be used as running examples.

▶ **Definition 3** (system of equations). *Let L be a complete lattice. A system of equations E over L is an ordered list of m equations of the form $x_i =_{\eta_i} f_i(x_1, \ldots, x_m)$, where $f_i \colon L^m \to L$ are monotone functions (with respect to the pointwise order on L^m) 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 this kind have been often considered in connection to verification problems (see e.g., [11, 42, 20, 2]). In particular, [20, 2] work on general classes of complete lattices.

Note that $\boldsymbol{f}$ can be seen as a function $\boldsymbol{f} \colon L^m \to L^m$. The solution of the system is a selected fixpoint of such function. We first need some auxiliary notation.

▶ **Definition 4** (substitution). *Given a system E of m equations over a complete 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 $\boldsymbol{x}'\boldsymbol{x}'' =_{\boldsymbol{\eta}'\boldsymbol{\eta}''} \boldsymbol{f}'\boldsymbol{f}''(\boldsymbol{x}', l, \boldsymbol{x}'')$.*

For solving a system of m equations $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, the last variable x_m is considered as a fixed parameter x and the system of $m-1$ equations $E[x_m := x]$ that arises from dropping the last equation is recursively solved. This produces an $(m-1)$-tuple parametric on x, i.e., we get $\boldsymbol{s}_{1,m-1}(x) = sol(E[x_m := x])$. Inserting this parametric solution into the last equation, we get an equation in a single variable $x =_{\eta_m} f_m(\boldsymbol{s}_{1,m-1}(x), x)$ that can be solved by taking for the function $\lambda x.\, f_m(\boldsymbol{s}_{1,m-1}(x), x)$, 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 $s_m = \eta_m(\lambda x.\, f_m(\boldsymbol{s}_{1,m-1}(x), x))$. The remaining components are obtained inserting s_m in the parametric solution $\boldsymbol{s}_{1,m-1}(x)$ previously computed, i.e., $\boldsymbol{s}_{1,m-1} = \boldsymbol{s}_{1,m-1}(s_m)$.

▶ **Definition 5** (solution). *Let L be a complete lattice and let E be a system of m equations over 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:*

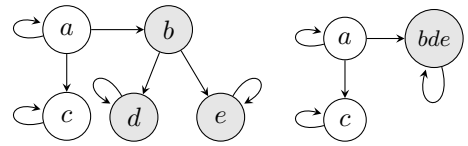
$$sol(\emptyset) = () \qquad sol(E) = (sol(E[x_m := s_m]),\, s_m)$$

where $s_m = \eta_m(\lambda x.\, f_m(sol(E[x_m := x]), x))$.

The order of equations matters: changing the order typically leads to a different solution.

▶ **Example 6** (solving a simple system of equations). Consider the powerset lattice $\mathbf{2}^S$ of any non-empty set S and the system of equations E consisting of the following two equations

$$x =_\mu x \cup y$$
$$y =_\nu x \cap y$$



(a) (b)

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

(d) $\begin{array}{lll} x_1 & =_\nu & \{b,d,e\} \cap \blacksquare_T x_1 \\ x_2 & =_\mu & x_1 \cup \blacklozenge_T x_2 \end{array}$

Figure 1 Transition systems and equational form for a μ-calculus formula.

In order to solve the system E, initially we need to compute the solution of the first equation $x =_\mu x \cup y$ parametric in y, that is, $s_x(y) = \mu(\lambda x.(x \cup y)) = y$. Now we can solve the second equation $y =_\nu x \cap y$ replacing x with the parametric solution, obtaining an equation in a single variable whose solution is $\nu(\lambda y.(s_x(y) \cap y)) = \nu(\lambda y.y) = S$. Finally, the solution of the first equation is obtained by inserting $y = S$ in the parametric solution $x = s_x(S) = S$.

Observe that even in this simple example the order of the equations matters. Indeed, if we consider the system where the two equations above are swapped the solution is $x = y = \emptyset$.

▶ **Example 7** (μ-calculus formulae as fixpoint equations)**.** We adopt 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 $p \in Prop$, $x \in PVar$ and $\eta \in \{\mu, \nu\}$.

The semantics of a formula is given with respect to an unlabelled transition system (or Kripke structure) $T = (\mathbb{S}_T, \to_T)$ where $\mathbb{S}_T$ is the set of states and $\to_T \subseteq \mathbb{S}_T \times \mathbb{S}_T$ is the transition relation. Given a formula φ and an environment ρ: $Prop \cup PVar \to \mathbf{2}^{\mathbb{S}_T}$ mapping each proposition or propositional variable to the set of states where it holds, we denote by $\|\varphi\|^T_\rho$ the semantics of φ defined as usual (see, e.g., [9]).

As observed by several authors (see, e.g., [11, 42]), a μ-calculus formula can be seen as a system of equations, with an equation for each fixpoint subformula. For instance, consider $\varphi = \mu x_2.((\nu x_1.(p \wedge \Box x_1)) \vee \Diamond x_2)$ that requires that a state is eventually reached from which p always holds. The equational form is reported in Fig. 1c. Consider a transition system $T = (\mathbb{S}_T, \to_T)$ where $\mathbb{S}_T = \{a, b, c, d, e\}$ and $\to_T$ is as depicted in Fig. 1a, with p that holds in the grey states b, d and e. Define the semantic counterpart of the modal operators as follows: given a relation $R \subseteq X \times X$ let $\blacklozenge_R, \blacksquare_R : \mathbf{2}^X \to \mathbf{2}^X$ be the functions defined, for $Y \subseteq X$, by $\blacklozenge_R(Y) = \{x \in X \mid \exists y \in Y.\ (x, y) \in R\}$, $\blacksquare_R(Y) = \{x \in X \mid \forall y \in X.(x, y) \in R \Rightarrow y \in Y\}$. Then the formula φ interpreted over the transition system T leads to the system of equations over the lattice $\mathbf{2}^{\mathbb{S}_T}$ in Fig. 1d, where we write $\blacklozenge_T$ and $\blacksquare_T$ for $\blacklozenge_{\to_T}$ and $\blacksquare_{\to_T}$.

The solution is $x_1 = \{b, d, e\}$ (states where p always holds) and $x_2 = \{a, b, d, e\}$ (states where the formula φ holds).

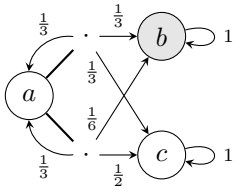
▶ **Example 8** (Łukasiewicz μ-terms)**.** Systems of equations over the real interval $[0, 1]$ have been considered in [34] as a precursor to model-checking PCTL or probabilistic μ-calculi. More precisely, the authors study a fixpoint extension of Łukasiewicz logic, referred to as Łukasiewicz μ-terms, whose syntax is as follows:

$$t \ ::= \ \mathbf{1} \mid \mathbf{0} \mid x \mid r \cdot t \mid t \sqcup t \mid t \sqcap t \mid t \oplus t \mid t \odot t \mid \eta x.t$$

where $x \in PVar$ is a variable (ranging over $[0, 1]$), $r \in [0, 1]$ and $\eta \in \{\mu, \nu\}$. The various syntactic operators have a semantic counterpart, given in Fig. 2a.

$\mathbf{0}(x) = 0,\ \mathbf{1}(x) = 1$	(constant)
$r \cdot x = rx$	(scalar mult.)
$x \sqcup y = \max(x, y)$	(weak disj.)
$x \sqcap y = \min(x, y)$	(weak conj.)
$x \oplus y = \min(x + y, 1)$	(strong disj.)
$x \odot y = \max(x + y - 1, 0)$	(strong conj.)

(a) Semantics of μ-terms ($x, y \in [0, 1]$).



(b) A PNDT.

$$\varphi \quad \begin{cases} x_1 =_\nu p \odot \Box x_1 \\ x_2 =_\mu x_1 \oplus \Diamond x_2 \end{cases}$$

$$\varphi' \quad \begin{cases} x_1 =_\nu p \odot \Box x_1 \\ x_2 =_\mu x_1 \oplus \Box x_2 \end{cases}$$

(c) Formulas as systems.

Figure 2

Then, each Łukasiewicz μ-term, in an environment $\rho : PVar \to [0, 1]$, can be assigned a semantics which is a real number in $[0, 1]$, denoted as $\|t\|_\rho$. Exactly as for the μ-calculus, a Łukasiewicz μ-term can be naturally seen as a system of fixpoint equations over the lattice $[0, 1]$. For instance, the term $\nu x_2.\ (\mu x_1.\ (\frac{5}{8} \oplus \frac{3}{8}x_2) \odot (\frac{1}{2} \sqcup (\frac{3}{8} \oplus \frac{1}{2}x_1)))$ from an example in [34], can be written as the system:

$$x_1 =_\mu (\frac{5}{8} \oplus \frac{3}{8}x_2) \odot (\frac{1}{2} \sqcup (\frac{3}{8} \oplus \frac{1}{2}x_1))$$
$$x_2 =_\nu x_1$$

▶ **Example 9** (Łukasiewicz μ-calculus)**.** The Łukasiewicz μ-calculus, as defined in [34], extends the Łukasiewicz μ-terms with propositions and modal operators. The syntax is as follows:

$$\varphi \ ::= \ p \mid \bar{p} \mid r \cdot \varphi \mid \varphi \sqcup \varphi \mid \varphi \sqcap \varphi \mid \varphi \oplus \varphi \mid \varphi \odot \varphi \mid \Diamond\varphi \mid \Box\varphi \mid \eta x.t$$

where x ranges in a set $PVar$ of propositional variables, p ranges in a set $Prop$ of propositional symbols, each paired with an associated complement $\bar{p}$, and $\eta \in \{\mu, \nu\}$.

The Łukasiewicz μ-calculus can be seen as a logic for probabilistic transition systems. It extends the quantitative modal μ-calculus of [31, 24] and it allows to encode PCTL [5]. For a finite set $\mathbb{S}$, the set of (discrete) probability distributions over $\mathbb{S}$ is defined as $\mathcal{D}(\mathbb{S}) = \{d : \mathbb{S} \to [0, 1] \mid \sum_{s \in \mathbb{S}} d(s) = 1\}$. A formula is interpreted over a *probabilistic non-deterministic transition system (PNDT)* $N = (\mathbb{S}, \to)$ where $\to \subseteq \mathbb{S} \times \mathcal{D}(\mathbb{S})$ is the transition relation. An example of PNDT can be found in Fig. 2b. Imagine that the aim is to reach state b. State a has two transitions. A "lucky" one where the probability to get to b is $\frac{1}{3}$ and an "unlucky" one where b is reached with probability $\frac{1}{6}$. For both transitions, with probability $\frac{1}{3}$ one gets back to a and then, with the residual probability, one moves to c. Once in states b or c, the system remains in the same state with probability 1.

Given a formula φ and an environment $\rho : Prop \cup PVar \to (\mathbb{S} \to [0, 1])$ mapping each proposition or propositional variable to a real-valued function over the states, the semantics of φ is a function $\|\varphi\|^N_\rho : \mathbb{S} \to [0, 1]$ defined as expected using the semantic operators. In addition to those already discussed, we have the semantic operators for the complement and the modalities: for $v : \mathbb{S} \to [0, 1]$

$$\bar{v}(x) = 1 - v(x) \qquad \blacklozenge_N(v)(x) = \max_{x \to d} \sum_{y \in \mathbb{S}} d(y) \cdot v(y) \qquad \blacksquare_N(v)(x) = \min_{x \to d} \sum_{y \in \mathbb{S}} d(y) \cdot v(y)$$

As it happens for the propositional μ-calculus, also formulas of the Łukasiewicz μ-calculus can be seen as systems of equations, but on a different complete lattice, i.e., $[0, 1]^{\mathbb{S}}$. For instance, consider the formulas $\varphi = \mu x_2.(\nu x_1.(p \odot \Box x_1) \oplus \Diamond x_2)$ and $\varphi' = \mu x_2.(\nu x_1.(p \odot \Box x_1) \oplus \Box x_2)$, rendered as (syntactic) equations in Fig. 2c. Roughly speaking, they capture the probability of eventually satisfying forever p, with an angelic scheduler and a daemonic one, choosing at each step the best or worst transition, respectively. Assuming that p holds with probability 1 on b and 0 on a and c, we have $\|\varphi\|_\rho(a) = \frac{1}{2}$ and $\|\varphi'\|_\rho(a) = \frac{1}{4}$.

▶ **Example 10** ((bi)similarity over transition systems). For defining (bi)similarity uniformly with the example on μ-calculus, we work on unlabelled transition systems with atoms $T = (\mathbb{S}, \to, A)$ where $A \subseteq \mathbf{2}^{\mathbb{S}}$ is a fixed set of atomic properties over the states. Everything can be easily adapted to labelled transition systems.

Given $T = (\mathbb{S}, \to, A)$, consider the lattice of relations on $\mathbb{S}$, namely $\mathsf{Rel}(\mathbb{S}) = (\mathbf{2}^{\mathbb{S}\times\mathbb{S}}, \subseteq)$. We take as basis the set of singletons $B_{\mathsf{Rel}(\mathbb{S})} = \{\{(x, y)\} \mid x, y \in \mathbb{S}\}$. The *similarity relation* on T, denoted $\precsim_T$, is the greatest fixpoint of the function $sim_T : \mathsf{Rel}(\mathbb{S}) \to \mathsf{Rel}(\mathbb{S})$, defined by

$$sim_T(R) = \left\{ (x, y) \in R \;\middle|\; \forall a \in A.\, (x \in a \Rightarrow y \in a) \wedge \forall x \to x'.\, \exists y \to y'.\, (x', y') \in R \right\}$$

In other words it can be seen as the solution of a system consisting of a single greatest fixpoint equation $x =_\nu sim_T(x)$.

For instance, consider the transition system T in Fig. 1a and take $p = \{b, d, e\}$ as the only atom. Then similarity $\precsim_T$ is the transitive reflexive closure of $\{(c, a), (a, b), (b, d), (d, e), (e, b)\}$.

Bisimilarity $\sim_T$ can be obtained analogously as the greatest fixpoint of $bis_T(R) = sim_T(R) \cap sim_T(R^{-1})$. In the transition system T above, bisimilarity $\sim_T$ is the equivalence such that $b \sim_T d \sim_T e$.

4 Approximation for Systems of Fixpoint Equations

In this section we design a theory of approximation for systems of fixpoint equations over complete lattices. The general setup is borrowed from abstract interpretation [13, 14], where a concrete domain C and an abstract domain A are fixed. Semantic operators on the concrete domain C have a counterpart in the abstract domain A, and suitable conditions can be imposed on such operators to ensure that the least fixpoints of the abstract operators are sound and/or complete approximations of the fixpoints of their concrete counterparts.

Similarly, here we will have a system of equations $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}^C(\boldsymbol{x})$ over a concrete domain C and its abstract counterpart $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}^A(\boldsymbol{x})$ over an abstract domain A, and we want that the solution of the latter provides an approximation of the solution of the former.

Let us first focus on the case of a single equation. Let $(C, \sqsubseteq)$ and $(A, \leq)$ be complete lattices and let $f^C : C \to C$ and $f^A : A \to A$ be monotone functions. The fact that f^A is a sound (over)approximation of f^C can be formulated in terms of a concretisation function $\gamma : A \to C$, that maps each abstract element $a \in A$ to a concrete element $\gamma(a) \in C$, for which, intuitively, a is an overapproximation. In the setting of abstract interpretation, where the interest is for program semantics, typically expressed in terms of least fixpoints, the desired *soundness* property is $\mu f^C \sqsubseteq \gamma(\mu f^A)$. A standard sufficient condition for soundness (see [13, 14, 33]) is $f^C \circ \gamma \sqsubseteq \gamma \circ f^A$. The same condition ensures soundness also for greatest fixpoints, i.e., $\nu f^C \sqsubseteq \gamma(\nu f^A)$, provided that γ is co-continuous and co-strict (see, e.g., [15, Proposition 15], which states the dual result).

Then we can suitably combine the conditions for least and greatest fixpoints. We will allow a different concretisation function for each equation.

▶ **Theorem 11** (sound concretisation for systems). *Let $(C, \sqsubseteq)$ and $(A, \leq)$ be complete lattices, let E_C of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}^C(\boldsymbol{x})$ and E_A of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}^A(\boldsymbol{x})$ be systems of m equations over C and A, with solutions $\boldsymbol{s}^C \in C^m$ and $\boldsymbol{s}^A \in A^m$, respectively. Let $\boldsymbol{\gamma}$ be an m-tuple of monotone functions, with $\gamma_i : A \to C$ for $i \in \underline{m}$. If $\boldsymbol{\gamma}$ satisfies $\boldsymbol{f}^C \circ \boldsymbol{\gamma}^\times \sqsubseteq \boldsymbol{\gamma}^\times \circ \boldsymbol{f}^A$ with γ_i co-continuous and co-strict for each $i \in \underline{m}$ such that $\eta_i = \nu$, then $\boldsymbol{s}^C \sqsubseteq \boldsymbol{\gamma}^\times(\boldsymbol{s}^A)$.*

The standard abstract interpretation framework of [16] relies on Galois connections: concretisation functions γ are right adjoints, whose left adjoint, the abstraction function α, intuitively maps each concrete element in C to its "best" overapproximation in A. When

$\langle \alpha, \gamma \rangle$ is a Galois connection, α is automatically continuous and strict, while γ is co-continuous and co-strict. This leads to the following result, where, besides the soundness conditions, we also make explicit the completeness conditions.

▶ **Theorem 12** (abstraction via Galois connections). *Let $(C, \sqsubseteq)$ and $(A, \leq)$ be complete lattices, let E_C of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}^C(\boldsymbol{x})$ and E_A of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}^A(\boldsymbol{x})$ be systems of m equations over C and A, with solutions $\boldsymbol{s}^C \in C^m$ and $\boldsymbol{s}^A \in A^m$, respectively. Let $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ be m-tuples of monotone functions, with $\langle \alpha_i, \gamma_i \rangle : C \to A$ a Galois connection for each $i \in \underline{m}$.*

1. Soundness*: If $\boldsymbol{\gamma}$ satisfies $\boldsymbol{f}^C \circ \boldsymbol{\gamma}^\times \sqsubseteq \boldsymbol{\gamma}^\times \circ \boldsymbol{f}^A$ or equivalently $\boldsymbol{\alpha}$ satisfies $\boldsymbol{\alpha}^\times \circ \boldsymbol{f}^C \leq \boldsymbol{f}^A \circ \boldsymbol{\alpha}^\times$, then $\boldsymbol{\alpha}^\times(\boldsymbol{s}^C) \leq \boldsymbol{s}^A$ (equivalent to $\boldsymbol{s}^C \sqsubseteq \boldsymbol{\gamma}^\times(\boldsymbol{s}^A)$).*
2. Completeness (for abstraction)*: If $\boldsymbol{\alpha}$ satisfies $\boldsymbol{f}^A \circ \boldsymbol{\alpha}^\times \leq \boldsymbol{\alpha}^\times \circ \boldsymbol{f}^C$ with α_i co-continuous and co-strict for each $i \in \underline{m}$ such that $\eta_i = \nu$, then $\boldsymbol{s}^A \leq \boldsymbol{\alpha}^\times(\boldsymbol{s}^C)$.*
3. Completeness (for concretisation)*: If $\boldsymbol{\gamma}$ satisfies $\boldsymbol{\gamma}^\times \circ \boldsymbol{f}^A \sqsubseteq \boldsymbol{f}^C \circ \boldsymbol{\gamma}^\times$ with γ_i continuous and strict for each $i \in \underline{m}$ such that $\eta_i = \mu$, then $\boldsymbol{\gamma}^\times(\boldsymbol{s}^A) \sqsubseteq \boldsymbol{s}^C$.*

Completeness for the abstraction, i.e., $\boldsymbol{s}^A \leq \boldsymbol{\alpha}^\times(\boldsymbol{s}^C)$, together with soundness, leads to $\boldsymbol{\alpha}^\times(\boldsymbol{s}^C) = \boldsymbol{s}^A$. This is a rare but very pleasant situation in which the abstraction does not lose any information as far as the abstract properties are concerned. We remark that here the notion of "completeness" slightly deviates from the standard abstract interpretation terminology where soundness is normally indispensable, and thus complete abstractions (see, e.g., [18]) are, by default, also sound.

Moreover, completeness for the concretisation is normally of limited interest in abstract interpretation. Alone, it states that the abstract solution is an underapproximation of the concrete one, while typically the interest is for overapproximations. Together with soundness, it leads to $\boldsymbol{s^C} = \boldsymbol{\gamma}^\times(\boldsymbol{s^A})$, a very strong property which is not meaningful in program analysis. In our case, keeping the concepts of soundness and completeness separated and considering also completeness for the concretisation is helpful in some cases, especially when dealing with up-to functions, which are designed to provide underapproximations of fixpoints.

Standard arguments also show that abstract operators can be obtained compositionally out of basic ones, preserving soundness.

▶ **Example 13** (abstraction for the μ-calculus). The paper [30] observes that (bi)simulations over transition systems can be seen as Galois connections and interpreted as abstractions. Then it characterises fragments of the μ-calculus which are preserved and strongly preserved by the abstraction. We next discuss how this can be derived as an instance of our framework.

Let $T_C = (\mathbb{S}_C, \to_C)$ and $T_A = (\mathbb{S}_A, \to_A)$ be transition systems and let $\langle \alpha, \gamma \rangle : \mathbf{2}^{\mathbb{S}_C} \to \mathbf{2}^{\mathbb{S}_A}$ be a Galois connection. It is a *simulation*, according to [30], if it satisfies the following condition: $\alpha \circ \blacklozenge_{T_C} \circ \gamma \subseteq \blacklozenge_{T_A}$. In this case T_A is called a $\langle \alpha, \gamma \rangle$-*abstraction* of T_C, written $T_C \sqsubseteq_{\langle \alpha, \gamma \rangle} T_A$. This can be shown to be equivalent to the ordinary notion of simulation between transition systems [30, Propositions 9 and 10]. In particular, if $R \subseteq \mathbb{S}_C \times \mathbb{S}_A$ is a simulation in the ordinary sense then one can consider $\langle \blacklozenge_{R^{-1}}, \blacksquare_R \rangle : \mathbf{2}^{\mathbb{S}_C} \to \mathbf{2}^{\mathbb{S}_A}$, where $\blacklozenge_{R^{-1}}$ is the function $\blacklozenge_{R^{-1}}(X) = \{y \in \mathbb{S}_A \mid \exists x \in X.\ (x, y) \in R\}$. This is a Galois connection (in the abstract interpretation setting $\blacklozenge_{R^{-1}}$ and $\blacksquare_R$ are often denoted $\widetilde{pre}_R$ and $post_R$, respectively [12]) inducing a simulation in the above sense, i.e., $\blacklozenge_{R^{-1}} \circ \blacklozenge_{T_C} \circ \blacksquare_R \subseteq \blacklozenge_{T_A}$.

When $T_C \sqsubseteq_{\langle \alpha, \gamma \rangle} T_A$, by [30, Theorem 2], one has that α "preserves" the $\mu\Diamond$-calculus, i.e., the fragment of the μ-calculus without $\Box$ operators. More precisely, for any formula φ of the $\mu\Diamond$-calculus, we have $\alpha(\|\varphi\|^{T_C}_{\rho}) \subseteq \|\varphi\|^{T_A}_{\alpha \circ \rho}$. This means that for each $s_C \in \mathbb{S}_C$, if s_C satisfies φ in the concrete system, then all the states in $\alpha(\{s_C\})$ satisfy φ in the abstract system, provided that each proposition p is interpreted in A with $\alpha(\rho(p))$, the abstraction of its interpretation in C.

This can be obtained as an easy consequence of Theorem 12, where we use the same function α as an abstraction for all equations. The condition $\alpha \circ \blacklozenge_{T_C} \circ \gamma \subseteq \blacklozenge_{T_A}$ above can be rewritten as $\alpha \circ \blacklozenge_{T_C} \subseteq \blacklozenge_{T_A} \circ \alpha$ which is the soundness condition $(\boldsymbol{\alpha}^{\times} \circ \boldsymbol{f}^{C} \leq \boldsymbol{f}^{A} \circ \boldsymbol{\alpha}^{\times})$ in Theorem 12 for the semantics of the diamond operator. For the other operators the soundness condition is trivially shown to hold. In fact,

- for $\mathbf{t}$ and $\mathbf{f}$ we have $\alpha(\emptyset) = \emptyset$ and $\alpha(\mathbb{S}_C) \subseteq \mathbb{S}_A$;
- for $\wedge$ and $\vee$ we have $\alpha(X \cup Y) = \alpha(X) \cup \alpha(Y)$ and $\alpha(X \cap Y) \subseteq \alpha(X) \cap \alpha(Y)$;
- a proposition p represents the constant function $\rho(p)$ in T_C and $\alpha(\rho(p))$ in T_A.

In order to extend the logic by including negation on propositions, in [30], an additional condition is required, called *consistency* of the abstraction with respect to the interpretation: $\alpha(\rho(p)) \cap \alpha(\overline{\rho(p)}) = \emptyset$, for all p. This is easily seen to be equivalent to $\alpha(\overline{\rho(p)}) \subseteq \overline{\alpha(\rho(p))}$ which is the soundness condition $(\boldsymbol{\alpha}^{\times} \circ \boldsymbol{f}^{C} \leq \boldsymbol{f}^{A} \circ \boldsymbol{\alpha}^{\times})$ in Theorem 12 for negated propositions.

Our theory naturally suggests generalisations of [30]. E.g., by (the dual of) Theorem 11, continuity and strictness of the abstraction α are sufficient to retain the results, hence one could deal with an abstraction not being an adjoint, thus going beyond ordinary simulations.

▶ **Example 14** (abstraction for Łukasiewicz μ-terms)**.** For Łukasiewicz μ-terms, as introduced in Example 8, leading to systems of fixpoint equations over the reals, we can consider as an abstraction a form of discretisation: for some fixed n define the abstract domain $[0,1]_{/n} = \{0\} \cup \{k/n \mid k \in \underline{n}\}$ and the insertion $\langle \alpha_n, \gamma_n \rangle : [0,1] \to [0,1]_{/n}$ with α_n defined by $\alpha_n(x) = \lceil n \cdot x \rceil / n$ and γ_n the inclusion. We can consider for all operators op, their best abstraction $op^{\#} = \alpha_n \circ op \circ \boldsymbol{\gamma_n}^{\times}$, thus getting a sound abstraction.

Note that for all semantic operators, $op^{\#}$ is the restriction of op to the abstract domain, with the exception of $r \cdot^{\#} x = \alpha_n(r \cdot x)$ for $x \in [0,1]_{/n}$. Moreover, for $x, y \in [0,1]$ we have

- $\alpha_n(\mathbf{0}(x)) = \mathbf{0}^{\#}(\alpha_n(x))$, $\alpha_n(\mathbf{1}(x)) = \mathbf{1}^{\#}(\alpha_n(x))$;
- $\alpha_n(r \cdot x) \leq r \cdot^{\#} \alpha_n(x)$;
- $\alpha_n(x \sqcup y) = \alpha_n(x) \sqcup^{\#} \alpha_n(y)$, $\alpha_n(x \sqcap y) = \alpha_n(x) \sqcap^{\#} \alpha_n(y)$;
- $\alpha_n(x \oplus y) \leq \alpha_n(x) \oplus^{\#} \alpha_n(y)$, $\alpha_n(x \odot y) \leq \alpha_n(x) \odot^{\#} \alpha_n(y)$ since $\alpha_n(x+y) \leq \alpha_n(x) + \alpha_n(y)$

i.e., the abstraction is complete for $\mathbf{0}$, $\mathbf{1}$, $\sqcup$, $\sqcap$, while it is just sound for the remaining operators.

For instance, the system in Example 8 can be shown to have solution $x_1 = x_2 = 0.2$. With abstraction α_{10} we get $x_1 = x_2 = 0.8$, with a more precise abstraction α_{100} we get $x_1 = x_2 = 0.22$ and with α_{1000} we get $x_1 = x_2 = 0.201$.

▶ **Example 15** (abstraction for Łukasiewicz μ-calculus)**.** Although space limitations prevent a detailed discussion, observe that when dealing with Łukasiewicz μ-calculus over some probabilistic transition system $N = (\mathbb{S}, \to)$, we can lift the Galois insertion above to $[0,1]^{\mathbb{S}}$. Define $\alpha_n^{\to} : [0,1]^{\mathbb{S}} \to [0,1]^{\mathbb{S}}_{/n}$ by letting, $\alpha_n^{\to}(v) = \alpha_n \circ v$ for $v \in [0,1]^{\mathbb{S}}$. Then $\langle \alpha_n^{\to}, \gamma_n^{\to} \rangle : [0,1]^{\mathbb{S}} \to [0,1]^{\mathbb{S}}_{/n}$, where $\gamma_n^{\to}$ is the inclusion, is a Galois insertion and, as in the previous case, we can consider the best abstraction for the operators of the Łukasiewicz μ-calculus.

For instance, consider the system for φ' in Example 9. Recall that the exact solution is $x_2(a) = 0.25$. With abstraction α_{10} we get $x_2(a) = 0.3$, with α_{15} we get $x_2(a) = 0.2\bar{6}$.

5 Up-To Techniques

Up-to techniques have been shown effective in easing the proof of properties of greatest fixpoints. Originally proposed for coinductive behavioural equivalences [32, 39], they have been later studied in the setting of complete lattices [35, 36]. Some recent work [6] started the exploration of the relation between up-to techniques and abstract interpretation. Roughly,

they work in a setting where the semantic functions of interest $f^* : L \to L$ admits a left adjoint $f_* : L \to L$, the intuition being that f^* and f_* are predicate transformers mapping a condition into, respectively, its strongest postcondition and weakest precondition. Then complete abstractions for f^* and sound up-to functions for f_* are shown to coincide. This has a natural interpretation in our game theoretic framework, as discussed in §6.2.

Here we take another view. We work with general semantic functions and, in §5.1, we first argue that up-to techniques can be naturally interpreted as abstractions where the concretisation is complete (and sound, if the up-to function is a closure). Then, in §5.2 we can smoothly extend up-to techniques from a single fixpoint to systems of fixpoint equations.

5.1 Up-To Techniques as Abstractions

The general idea of up-to techniques is as follows. Given a monotone function $f : L \to L$ one is interested in the greatest fixpoint νf. In general, the aim is to establish whether some given element of the lattice $l \in L$ is under the fixpoint, i.e., if $l \sqsubseteq \nu f$. In turn, since by Tarski's Theorem, $\nu f = \bigsqcup\{x \mid x \sqsubseteq f(x)\}$, this amounts to proving that l is under some post-fixpoint l', i.e., $l \sqsubseteq l' \sqsubseteq f(l')$. For instance, consider the function $bis_T : \mathsf{Rel}(\mathbb{S}) \to \mathsf{Rel}(\mathbb{S})$ for bisimilarity on a transition system T in Example 10. Given two states $s_1, s_2 \in \mathbb{S}$, proving $\{(s_1, s_2)\} \subseteq \nu bis_T$, i.e., showing the two states bisimilar, amounts to finding a post-fixpoint, i.e., a relation R such that $R \subseteq bis_T(R)$ (namely, a bisimulation) such that $\{(s_1, s_2)\} \subseteq R$.

▶ **Definition 16** (up-to function). *Let L be a complete lattice and let $f : L \to L$ be a monotone function. A* sound up-to function *for f is any monotone function $u : L \to L$ such that $\nu(f \circ u) \sqsubseteq \nu f$. It is called* complete *if also the converse inequality $\nu f \sqsubseteq \nu(f \circ u)$ holds.*

When u is sound, if l is a post-fixpoint of $f \circ u$, i.e., $l \sqsubseteq f(u(l))$ we have $l \sqsubseteq \nu(f \circ u) \sqsubseteq \nu f$. The idea is that the characteristics of u should make it easier to prove that l is a postfix-point of $f \circ u$ than proving that it is for f. This is clearly the case when u is extensive. In fact by extensiveness of u and monotonicity of f we get $f(l) \sqsubseteq f(u(l))$ and thus obtaining $l \sqsubseteq f(u(l))$ is "easier" than obtaining $l \sqsubseteq f(l)$. Note that extensiveness also implies "completeness" of the up-to function: since $f \sqsubseteq f \circ u$ clearly $\nu f \sqsubseteq \nu(f \circ u)$. We remark that for up-to functions, since the interest is for underapproximating fixpoints, the terms soundness and completeness are somehow reversed with respect to their meaning in abstract interpretation.

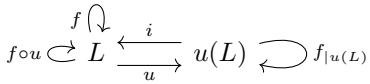
A common sufficient condition ensuring soundness of up-to functions is compatibility [35].

▶ **Definition 17** (compatibility). *Let L be a complete lattice and let $f : L \to L$ be a monotone function. A monotone function $u : L \to L$ is* f-compatible *if $u \circ f \sqsubseteq f \circ u$.*

The soundness of an f-compatible up-to function u can be proved by viewing it as an abstraction. When u is a closure (i.e., extensive and idempotent), $u(L)$ is a complete lattice that can be seen as an abstract domain in a way that $\langle u, i\rangle : L \to u(L)$, with i being the inclusion, is a Galois insertion. Moreover $f_{|u(L)}$ can be shown to provide an abstraction of both f and $f \circ u$ over L, sound and complete with respect to the inclusion i, seen as the concretisation. The formal details are given below. Since we later aim to apply up-to techniques to systems of equations, we deal with not only greatest but also least fixpoints.

▶ **Lemma 18** (compatible up-to functions as sound and complete abstractions). *Let $f : L \to L$ be a monotone function and let $u : L \to L$ be an f-compatible closure. Consider the Galois insertion $\langle u, i\rangle : L \to u(L)$ where $i : u(L) \to L$ is the inclusion. Then*

1. *f restricts to $u(L)$, i.e., $f_{|u(L)} : u(L) \to u(L)$;*
2. *$\nu f = i(\nu f_{|u(L)}) = \nu(f \circ u)$. If u is continuous and strict then $\mu f = i(\mu f_{|u(L)}) = \mu(f \circ u)$.*



When the up-to function is just f-compatible (hence sound), but possibly not a closure, we canonically turn u into an f-compatible closure by taking the least closure $\bar{u}$ above u.

▶ **Corollary 19** (soundness of compatible up-to functions). *Let $f : L \to L$ be a monotone function, let $u : L \to L$ be an f-compatible up-to function and let $\bar{u}$ be the least closure above u. Then $\nu(f \circ u) \sqsubseteq \nu(f \circ \bar{u}) = \nu f$. If u is continuous and strict, then $\mu(f \circ u) \sqsubseteq \mu(f \circ \bar{u}) = \mu f$.*

In [35] the proof of soundness of a compatible up-to technique u relies on the definition of a function u^ω defined as $u^\omega(x) = \bigsqcup\{u^n(x) \mid n \in \mathbb{N}\}$, where $u^n(x)$ is defined inductively as $u^0(x) = x$ and $u^{n+1}(x) = u(u^n(x))$. The function u^ω is extensive but not idempotent in general, and it can be easily seen that $u^\omega \sqsubseteq \bar{u}$. The paper [36] shows that for any monotone function one can consider the largest compatible up-to function, the so-called companion, which is extensive and idempotent. The companion could be used in place of $\bar{u}$ for part of the theory. However, we find it convenient to work with $\bar{u}$ since, despite not discussed in the present paper, it plays a key role for the integration of up-to techniques into the verification algorithms. Furthermore the companion is usually hard to determine.

5.2 Up-To Techniques for Systems of Equations

Exploiting the view of up-to functions as abstractions, moving to systems of equations is easy. As in the case of abstractions, a different up-to function is allowed for each equation.

▶ **Definition 20** (compatible up-to for systems of equations). *Let $(L, \sqsubseteq)$ be a complete lattice and let E be $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, a system of m equations over L. A* compatible tuple of up-to functions *for E is an m-tuple of monotone functions $\boldsymbol{u}$, with $u_i : L \to L$, satisfying compatibility ($\boldsymbol{u}^\times \circ \boldsymbol{f} \sqsubseteq \boldsymbol{f} \circ \boldsymbol{u}^\times$) with u_i continuous and strict for each $i \in \underline{m}$ such that $\eta_i = \mu$.*

We can then generalise Corollary 19 to systems of equations.

▶ **Theorem 21** (up-to for systems). *Let $(L, \sqsubseteq)$ be a complete lattice and let E be $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, a system of m equations over L, with solution $\boldsymbol{s} \in L^m$. Let $\boldsymbol{u}$ be a compatible tuple of up-to functions for E and let $\bar{\boldsymbol{u}} = (\bar{u}_1, \ldots, \bar{u}_m)$ be the corresponding tuple of least closures. Let $\boldsymbol{s}'$ and $\bar{\boldsymbol{s}}$ be the solutions of the systems $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{u}^\times(\boldsymbol{x}))$ and $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\bar{\boldsymbol{u}}^\times(\boldsymbol{x}))$, respectively. Then $\boldsymbol{s}' \sqsubseteq \bar{\boldsymbol{s}} = \boldsymbol{s}$. Moreover, if $\boldsymbol{u}$ is extensive then $\boldsymbol{s}' = \boldsymbol{s}$.*

▶ **Example 22** (μ-calculus up-to (bi)similarity). Consider the problem of model-checking the μ-calculus over some transition system with atoms $T = (\mathbb{S}, \to, A)$.

Assuming that we have an a priori knowledge about the similarity relation $\precsim$ over some of the states in T, then, restricting to a suitable fragment of the μ-calculus we can avoid checking the same formula on similar states. This intuition can be captured in the form of an up-to technique, that we refer to as up-to similarity. It is based on an up-to function $\boldsymbol{u}_{\precsim} : \mathbf{2}^{\mathbb{S}} \to \mathbf{2}^{\mathbb{S}}$ defined, for $X \in \mathbf{2}^{\mathbb{S}}$, by $u_{\precsim}(X) = \{s \in \mathbb{S} \mid \exists s' \in X.\ s' \precsim s\}$.

Function $u_{\precsim}$ is monotone, extensive, and idempotent. It is also continuous and strict.

Moreover, $u_{\precsim}$ is a compatible (and thus sound) up-to function for the $\mu\Diamond$-calculus where propositional variables are interpreted as atoms. In fact, $\precsim$ is a simulation (the largest one) and the function $u_{\precsim}$ is the associated abstraction as defined in Example 13, namely $u_{\precsim} = \blacklozenge_{\precsim}$. Therefore, compatibility $u_{\precsim} \circ f \sqsubseteq f \circ u_{\precsim}$ corresponds to condition $\alpha \circ \blacklozenge_{T_C} \circ \gamma \subseteq \blacklozenge_{T_A}$ in Example 13 which has been already observed to coincide with soundness in the sense of

Table 1 The game on the powerset of the basis.

Position	Player	Moves
(b, i)	$\exists$	$\boldsymbol{X}$ s.t. $b \sqsubseteq f_i(\bigsqcup \boldsymbol{X})$
$\boldsymbol{X}$	$\forall$	(b', j) s.t. $b' \in X_j$

Theorem 12 for the operators of the $\mu\Diamond$-calculus. Concerning propositional variables, in Example 13, they were interpreted, in the target transition system, by the abstraction of their interpretation in the source transition system. Since here we have a single transition system and a single interpretation $\rho : \mathit{Prop} \to \mathbf{2}^{\mathbb{S}}$, we must have $\rho(p) = u_{\precsim}(\rho(p))$, i.e., $\rho(p)$ upward-closed with respect to $\precsim$. This automatically holds by the fact that $\precsim$ is a simulation.

Similarly, we can define up-to bisimilarity via the up-to function $u_{\sim}(X) = \{s \in \mathbb{S} \mid \exists s' \in X.\ s \sim s'\}$. As above, one can see that compatibility $u_{\sim} \circ f \sqsubseteq f \circ u_{\sim}$ holds for the full μ-calculus with propositional variables interpreted as atoms. For instance, consider the formula φ in Example 7 and the transition system in Fig. 1a. Using the up-to function $u_{\sim}$ corresponds to working in the bisimilarity quotient in Fig. 1b. Note, however, that when using a local algorithm (see §6.2) the quotient does not need to be actually computed. Rather, only the bisimilarity over the states explored by the searching procedure is possibly exploited.

▶ **Example 23** (bisimilarity up-to transitivity)**.** Consider the problem of checking bisimilarity on a transition system $T = \langle \mathbb{S}, \to \rangle$. A number of well-known sound up-to techniques have been introduced in the literature [37]. As an example, we consider the up-to function $u_{tr} : \mathsf{Rel}(\mathbb{S}) \to \mathsf{Rel}(\mathbb{S})$ performing a single step of transitive closure. It is defined as:

$$u_{tr}(R) = R \circ R = \{(x, y) \mid \exists\, z \in \mathbb{S}.\ (x, z) \in R \ \wedge\ (z, y) \in R\}.$$

It is easy to see that u_{tr} is monotone and compatible with respect to the function $\mathit{bis}_T : \mathsf{Rel}(\mathbb{S}) \to \mathsf{Rel}(\mathbb{S})$ of which bisimilarity is the greatest fixpoint (see Example 10). Since A is deterministic, bisimilarity coincides with language equivalence.

Note that u_{tr} is neither idempotent nor extensive. The corresponding closure $\bar{u}_{tr}$ maps a relation to its (full) transitive closure (this is known to be itself a sound up-to technique, a fact that we can also derive from the compatibility of u_{tr} and Corollary 19).

6 Solving Systems of Equations via Games

In this section, we first provide a characterisation of the solution of a system of fixpoint equations over a complete lattice in terms of a parity game. This generalises a result in [2]. While the original result was limited to continuous lattices, here, exploiting the results on abstraction in §4, we devise a game working for any complete lattice.

The game characterisation opens the way to the development of algorithms for solving the game and thus the associated verification problem. A proper treatment of these aspects is beyond the scope of the present paper, but covered in [3]. Here, in §6.2, we hint at the algorithmic potentials of our theory focusing on the case of a single equation.

6.1 Game Characterization

We show that the solution of a system of equations over a complete lattice can be characterised using a parity game.

▶ **Definition 24** (powerset game). *Let L be a complete lattice with a basis B_L. Given a system E of m equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, the corresponding* powerset 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 $b \in B_L$, $i \in \underline{m}$. Those of $\forall$ are tuples of subsets of the basis $\boldsymbol{X} = (X_1, \ldots, X_m) \in (\mathbf{2}^{B_L})^m$.*
- *From position (b, i) the moves of $\exists$ are $\mathbf{E}(b, i) = \{\boldsymbol{X} \mid \boldsymbol{X} \in (\mathbf{2}^{B_L})^m \ \wedge \ b \sqsubseteq f_i(\bigsqcup \boldsymbol{X})\}$.*
- *From position $\boldsymbol{X} \in (\mathbf{2}^{B_L})^m$ the moves of $\forall$ are $\mathbf{A}(\boldsymbol{X}) = \{(b, i) \mid i \in \underline{m} \ \wedge \ b \in X_i\}$.*

The game is schematised in Table 1. For a finite play, the winner is the player who moved last. 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.

Interestingly, the correctness and completeness of the game can be proved by exploiting the results in §4. The crucial observation is that there is a Galois insertion between L and the powerset lattice of its basis (which is algebraic hence continuous) $\langle \alpha, \gamma \rangle : \mathbf{2}^{B_L} \to L$ where abstraction α is the join $\alpha(X) = \bigsqcup X$ and concretisation γ takes the lower cone $\gamma(l) = {\downarrow} l \cap B_L$. Then a system of equations over a complete lattice L can be "transferred" to a system of equations over the powerset of the basis $\mathbf{2}^{B_L}$ along such insertion, in a way that the system in L can be seen as a sound and complete abstraction of the one in $\mathbf{2}^{B_L}$.

▶ **Theorem 25** (correctness and completeness). *Let E be a system of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{s}$. For all $b \in B_L$ and $i \in \underline{m}$, $b \sqsubseteq s_i$ iff $\exists$ has a winning strategy from position (b, i).*

6.2 An Algorithmic View

The game theoretical characterisation can be the basis for the development of algorithms, possibly integrating abstraction and up-to techniques, for solving systems of equations. Here we consider local algorithms for the case of a single equation. Our main focus is to provide a general procedure which transcends the verification problem at hand, and also takes advantage of heuristics based on abstractions and up-to techniques. This allows us also to establish a link with some recent work relating abstract interpretation and up-to techniques [6] and exploiting up-to techniques for computing language equivalence on NFAs [8]. While not improving the complexity bounds, our algorithm is still in line with other local algorithms designed for specific settings, such as [8, 21, 22], as they arise as proper instantiations.

An algorithm for general systems is considerably more difficult and cannot be described here due to lack of space (it can be found in the full version of this paper [3]). Here we focus on the special case of a single (greatest) fixpoint equation $x =_\nu f(x)$.

6.2.1 Selections

For a practical use of the game it can be useful to observe that the set of moves of the existential player can be suitably restricted without affecting the completeness of the game, by introducing a notion of selection, similarly to what is done in [2].

Given a lattice L, define a preorder $\sqsubseteq_H$ on $\mathbf{2}^{B_L}$ by letting, for $X, Y \in \mathbf{2}^{B_L}$, $X \sqsubseteq_H Y$ if $\bigsqcup X \sqsubseteq \bigsqcup Y$. (The subscript H comes from the fact that for completely distributive lattices, if B_L is the set of irreducible elements, then $\sqsubseteq_H$ is the "Hoare preorder" [1], requiring that $\forall x \in X. \exists y \in Y. x \sqsubseteq y$.) Observe that $\sqsubseteq_H$ is not antisymmetric. We write $\equiv_H$ for the corresponding equivalence, i.e., $X \equiv_H Y$ when $X \sqsubseteq_H Y \sqsubseteq_H X$.

The moves of player $\exists$ can be ordered by the pointwise extension of $\sqsubseteq_H$, thus leading to the following definition. Since we deal with a single equation, we will omit the indices from the positions of player $\exists$ and write b instead of $(b, 1)$.

▶ **Definition 26** (selection). *Let $x =_\nu f(x)$ be an equation over a complete lattice L, with basis B_L. A* selection *is a function $\sigma : B_L \to \mathbf{2}^{\mathbf{2}^{B_L}}$ such that for all $b \in B_L$ it holds $\uparrow_H \sigma(b) = \mathbf{E}(b)$, i.e. the set of moves of $\exists$ from position b, where $\uparrow_H$ is the upward-closure with respect to $\sqsubseteq_H$.*

This is equivalent to requiring that $\sigma(b) \subseteq \mathbf{E}(b)$ and for each $X \in \mathbf{E}(b)$ there exists $Y \in \sigma(b)$ such that $\bigsqcup Y \sqsubseteq \bigsqcup X$.

For the case of a single fixpoint equation it is easy to see that Theorem 25 continues to hold if we restrict the moves of player $\exists$ to those prescribed by a selection.

▶ **Theorem 27** (game with selections). *Let $x =_\nu f(x)$ be an equation over a complete lattice L with solution s. For all $b \in B_L$, it holds that $b \sqsubseteq s$ iff $\exists$ has a winning strategy from position b in the game restricted to selections.*

6.2.2 Local Algorithm for a Special Case

In this section we assume that $f : L \to L$ is some fixed function that preserves non-empty meets, i.e., for $X \neq \emptyset$, $f(\bigsqcap X) = \bigsqcap f(X)$. This is equivalent to asking $f(x) = f^*(x) \sqcap c$ for some $c \in L$ (just take $c = f(\top)$), with f^* being a right adjoint of a map f_*, a setting that has been studied also in [6]. We will call a function satisfying this assumption a *deterministic function*. Note that the adjunction $\langle f_*, f^* \rangle$ is completely orthogonal to the adjunctions (Galois connections) studied so far.

▶ **Example 28.** For a simple example adopted from [8], consider a deterministic finite automaton $A = (Q, \Sigma, \delta, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function and $F \subseteq Q$ is the set of final states. Since A is deterministic, language equivalence coincides with bisimilarity. Consider the lattice of relations $L = (\mathbf{2}^{Q \times Q}, \sqsubseteq)$ with basis $B_L = \{\{(q_1, q_2)\} \mid q_1, q_2 \in Q\}$. The behaviour map, having bisimilarity as largest fixpoint, is $f : \mathbf{2}^{Q \times Q} \to \mathbf{2}^{Q \times Q}$ defined as $f(R) = f^*(R) \cap C$ where $f^*(R) = \{(q_1, q_2) \mid \forall a \in \Sigma.\, (\delta(q_1, a), \delta(q_2, a)) \in R\}$ with $C = \{(q_1, q_2) \mid q_1 \in F \iff q_2 \in F\}$. The left adjoint is $f_*(R) = \{(\delta(q_1, a), \delta(q_2, a)) \mid (q_1, q_2) \in R, a \in \Sigma\}$.

Given two states $q_1, q_2 \in R$, we want to decide whether $(q_1, q_2) \in S$, where S is bisimilarity, the solution of the greatest fixpoint equation $R =_\nu f(R)$.

We first observe that for deterministic functions we can take a very simple selection.

▶ **Lemma 29** (selection). *Let L be a complete lattice with basis B_L, and let $f : L \to L$ be a deterministic function, i.e., $f(x) = f^*(x) \sqcap c$ for some $c \in L$ and $\langle f_*, f^* \rangle : L \to L$ a Galois connection. A selection $\sigma : B_L \to \mathbf{2}^{\mathbf{2}^{B_L}}$ for $x =_\nu f(x)$ can be defined, for $b \in B_L$, as:*

$$\sigma(b) = \begin{cases} \{X\} \text{ with } X \subseteq B_L \text{ s.t. } X \equiv_H {\downarrow} f_*(b) \cap B_L & \text{when } b \sqsubseteq c \\ \emptyset & \text{otherwise} \end{cases}$$

Observe that there might be several choices for $X \subseteq B_L$: one that always works is $X = {\downarrow} f_*(b) \cap B_L$, but subsets $X \subseteq {\downarrow} f_*(b) \cap B_L$ are also feasible, as long as $\bigsqcup X = f_*(b)$. In Example 28, given $\{(q_1, q_2)\} \in B_L$, we can define $\sigma(\{(q_1, q_2)\}) = \{\{\{(q_1', q_2')\} \mid (q_1', q_2') \in f_*(\{(q_1, q_2)\})\}\} = \{\{\{(\delta(q_1, a), \delta(q_2, a))\} \mid a \in \Sigma\}\}$.

By Lemma 29, in the game for $x =_\nu f(x)$, either the existential player is stuck or she has a best move. As a consequence, the game in §6.1 can be simplified. Let B_L be any basis for L such that $\bot \notin B_L$. The moves of player $\exists$ are deterministic, governed by σ, and only player $\forall$ has choices when exploring the elements included in such moves.

For checking whether $b \sqsubseteq \nu f$, for some $b \in B_L$, the game starts from position b. Then, at a generic position b', we do the following:

1. if $b' \not\sqsubseteq c$ then $\sigma(b') = \emptyset$ and $\exists$ loses;
2. otherwise, $\exists$ has to play the only element in $\sigma(b') = \{X\}$
   a. if $f_*(b') = \bot$ then take $X = \emptyset$; hence $\exists$ wins since $\forall$ has no moves;
   b. if instead $f_*(b') \neq \bot$, we can take $X \equiv_H {\downarrow} f_*(b') \cap B_L$ and thus player $\forall$ can play any $b'' \in X$ and the game continues.

Player $\exists$ wins the game iff no losing position for her ($b' \not\sqsubseteq c$) is encountered in the exploration. When a losing position for $\exists$ is encountered we immediately know that $\forall$ wins.

The game can be further simplified by observing that, if W denotes the set of positions already visited during the exploration, whenever, at a position b', we have $b' \sqsubseteq \bigsqcup W$ then $\exists$ wins from b' as long as she wins from all the positions in W. This leads to the local algorithm outlined in List. 1, whose proof of correctness formalises the arguments above. The procedure `Explore` allows to check if $b \sqsubseteq \nu f = \nu(f^* \sqcap c)$ by invoking `Explore(`b`,`$\emptyset$`)`, which returns `true` if and only if player $\exists$ wins in the simplified game.

Listing 1 Local algorithm for the simplified game.

```
Explore(b',W):
  if b' ⋢ c then return false;
  else if b' ⊑ ⨆W then return true;
  else take X ⊆ B_L s.t. X ≡_H ↓f_*(b') ∩ B_L;
       return ∧_{b''∈X} Explore(b'',W ∪ {b'});
```

▶ **Theorem 30** (correctness and completeness of the simplified game). *Let L be a complete lattice with basis $B_L \subseteq L \setminus \{\bot\}$, and let $f : L \to L$ be a deterministic function, i.e., $f(x) = f^*(x) \sqcap c$ for some $c \in L$ and $\langle f_*, f^* \rangle : L \to L$ a Galois connection. Then, for all $b \in B_L$, $b \sqsubseteq \nu f$ iff the invocation* `Explore(`b`,`$\emptyset$`)` *returns* `true`.

For instance, for Example 28, the local algorithm of List. 1 works as follows: for checking whether $\{(q_1, q_2)\}$ is dominated by the solution, i.e., states q_1 and q_2 are bisimilar, one starts from $\{(q_1, q_2)\}$. At position $\{(q_1', q_2')\}$, if one state is final and the other is not, $\exists$ loses. If the pair has been already explored, the branch is not considered. Otherwise, the pairs arising as a-successors $\{(q_1', q_2')\}$ are explored. If no losing position is found, the exploration finishes (recall that there are finitely many states) and $\bigcup W$ is a bisimulation including (q_1, q_2).

Observe that when the basis is $B_L = L \setminus \{\bot\}$, the game becomes deterministic also for player $\forall$: in List. 1, when $f_*(b') \neq \bot$ one can take $X = \{\{f_*(b')\}\}$, otherwise $X = \emptyset$. Therefore, since f_* is a left adjoint and thus continuous, if we take the set S of all the positions generated during the exploration (i.e., W with the addition of the last position, for finite games) then $\bigsqcup S = \bigsqcup_i f_*^i(b)$ is the least fixpoint of f_* above b, which in turn coincides with the least fixpoint of $f^* \sqcup b$. This establishes a direct link with [6] which shows that for $b \in L$ it holds that $\mu(f^* \sqcup b) \sqsubseteq c$ iff $b \sqsubseteq \nu(f^* \sqcap c) = \nu f$.

Furthermore, we can bring up-to techniques into the picture: given an up-to function u we can modify the procedure in List. 1 by replacing the winning condition for $\exists$, that is, $b' \sqsubseteq \bigsqcup W$, by $b' \sqsubseteq u(\bigsqcup W)$. The procedure remains clearly complete and it is also correct due to Theorem 21. This allows us to cover the algorithm in [8] which checks language equivalence for non-deterministic automata. It performs on-the-fly determinization and constructs a bisimulation up-to congruence on the determinized automaton. More concretely, it tries to construct a bisimulation relation for the determinized automaton (along the lines of Example 28) and remembers pairs (X_1, X_2) of sets of states seen so far in a relation W (as explained in the algorithm in List. 1). Once a pair (Y_1, Y_2) is encountered that is contained in the congruence closure of W (the least equivalence, closed under union, that contains W), one can stop exploring this branch. A more detailed comparison can be found in Appendix A.

7 Conclusion

Our contribution is based on the notion of approximation as formalised in abstract interpretation [13, 14]. Due to the intimate connection of Galois connections and closure functions, there is a close correspondence with up-to techniques for enhancing coinduction proofs [35, 37], originally developed for CCS [32]. However, as far as we know, recent research has only started to explore this connection: [6] explains the relation between sound up-to techniques and complete abstract domains in the setting where the semantic function has an adjoint. This adjunction or Galois connection plays a different role than the abstractions: it gives the existential player a unique best move, a concept explored in §6.2.2.

Fixpoint equation systems largely derive their interest from μ-calculus model-checking [9]. Evaluating μ-calculus formulae on a transition system can be reduced to solving a parity game and the exact complexity of this task is still open. Progress measures, introduced in [25], allow one to solve parity games with a complexity which is polynomial in the number of states and exponential in (half of) the alternation depth of the formula. Recently quasi-polynomial algorithms for parity games [10, 26, 29] have been devised. Instead of improving the complexity bounds, our aim here is to introduce heuristics, based on an on-the-fly algorithm and up-to functions that are known to achieve good efficiency in practice.

Many papers deal with abstraction in the setting of μ-calculus model checking. We noted that the results on simulation-based abstraction in [30] can be obtained as an instance of our framework. The abstraction of the μ-calculus along a Galois connection and its soundness is discussed in [4]. A general framework for abstract interpretation of temporal calculi and logics is developed in [15]. In particular, an abstract calculus for expressing nested fixpoint expressions is studied, parametric with respect to the basic operators. The calculus is interpreted over complete boolean lattices and conditions ensuring the soundness and the completeness of the abstraction along a Galois connection are singled out. Such results are closely related to those in Section 4. The main differences reside in the fact that we work with general complete lattices, rather than with boolean lattices. In addition, we treat separately soundness and completeness, and, in order to establish a connection with up-to techniques, we distinguish two forms of completeness (for the abstraction and for the concretisation).

We showed – for a special case – how on-the-fly algorithms inspired by [8, 6, 21, 22] for a single (greatest) fixpoint equation can be adapted to the case of general lattices. For the general case of arbitrary fixpoint equation systems a considerably more complex generalisation along the lines of [44] is possible, but omitted due to lack of space.

The use of assumptions as stopping conditions in the algorithm is reminiscent of parameterized coinduction [43, 23], closely related to up-to-techniques, as spelled out in [36].

The notion of progress measures that has been studied in [2] can be adapted to the game for arbitrary complete (rather than just continuous) lattices, introduced in this paper. A first natural question is whether the on-the-fly algorithm arises as an instance of the single equation algorithm instantiated with the progress measure fixpoint equation.

With respect to the applications, we believe that our case study on abstractions respectively simulations for μ-calculus model-checking can also be generalised to modal respectively mixed transition systems [40, 17, 28] or to abstraction for the full μ-calculus as studied in [19] by combining both under- and over-approximations. Furthermore, we plan to further study over-approximations for fixpoint equations over the reals, closely connected to probabilistic logics. In particular, we will investigate under which circumstances one can obtain guarantees to be close to the exact solution or to compute the exact solution directly. Another interesting area is the use of up-to techniques for behavioural metrics [7].

References

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

2 Paolo Baldan, Barbara König, Christina Mika-Michalski, and Tommaso Padoan. Fixpoint games in continuous lattices. In Stephanie Weirich, editor, *Proc. of POPL'19*, volume 3, pages 26:1–26:29. ACM, 2019.

3 Paolo Baldan, Barbara König, and Tommaso Padoan. Abstraction, up-to techniques and games for systems of fixpoint equations, 2020. `arXiv:2003.08877`.

4 Gourinath Banda and John P. Gallagher. Constraint-based abstract semantics for temporal logic: A direct approach to design and implementation. In Edmund M. Clarke and Andrei Voronkov, editors, *LPAR 2010*, volume 6355 of *Lecture Notes in Computer Science*, pages 27–45. Springer, 2010.

5 Andrea Bianco and Luca de Alfaro. Model checking of probabalistic and nondeterministic systems. In *Proc. of FSTTCS '95*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer, 1995.

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

7 Filippo Bonchi, Barbara König, and Daniela Petrişan. Up-to techniques for behavioural metrics via fibrations. In *Proc. of CONCUR '18*, volume 118 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl – Leibniz Center for Informatics, 2018.

8 Filippo Bonchi and Damien Pous. Checking NFA equivalence with bisimulations up to congruence. In *Proc. of POPL '13*, pages 457–468. ACM, 2013.

9 Julian Bradfield and Igor Walukiewicz. The mu-calculus and model checking. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 871–919. Springer, 2018.

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

11 Rance Cleaveland, Marion Klein, and Bernhard Steffen. Faster model checking for the modal mu-calculus. In *Proc. of CAV 1992*, volume 663 of *Lecture Notes in Computer Science*, pages 410–422. Springer, 1992.

12 Patrick Cousot. Partial completeness of abstract fixpoint checking. In Berthe Y. Choueiry and Toby Walsh, editors, *Proceedings of the Fourth International Symposium on Abstraction, Reformulations and Approximation, SARA'2000*, volume 1864 of *Lecture Notes in Computer Science*, pages 1–25. Springer, 26–29 July 2000.

13 Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. of POPL '77*, pages 238–252. ACM, 1977.

14 Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *Proc. of POPL '79*, pages 269–282. ACM, 1979.

15 Patrick Cousot and Radhia Cousot. Temporal abstract interpretation. In Mark N. Wegman and Thomas W. Reps, editors, *Proc. of POPL '00*, pages 12–25. ACM, 2000.

16 Radhia Cousot and Patrick Cousot. Constructive versions of tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43–57, 1979.

17 Dennis Dams, Rob Gerth, and Orna Grumberg. Abstract interpretation of reactive systems. *ACM Trans. Program. Lang. Syst.*, 19(2):253–291, 1997.

18 Roberto Giacobazzi, Francesco Ranzato, and Francesca Scozzari. Making abstract interpretations complete. *Journal of the ACM*, 47(2):361–416, 2000.

19 Orna Grumberg, Martin Lange, Martin Leucker, and Sharon Shoham. When not losing is better than winning: Abstraction and refinement for the full mu-calculus. *Information and Computation*, 205(8):1130–1148, 2007.

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

21 Daniel Hirschkoff. Automatically proving up to bisimulation. In *Proc. of MFCS '98 Workshop on Concurrency*, number 18 in Electronic Notes in Theoretical Computer Science, pages 75–89. Elsevier, 1998.

22 Daniel Hirschkoff. *Mise en oeuvre de preuves de bisimulation.* PhD thesis, Ecole Nationale des Ponts et Chaussées, 1999.

23 Chung-Kil Hur, Georg Neis, Derek Dreyer, and Viktor Vafeiadis. The power of parameterization in coinductive proof. In *Proc. of POPL '13*, pages 193–206. ACM, 2013.

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

25 Marcin Jurdziński. Small progress measures for solving parity games. In *Proc. of STACS '00*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000.

26 Marcin Jurdzinski and Ranko Lazic. Succinct progress measures for solving parity games. In *Proc. of LICS '17*, pages 1–9. ACM/IEEE, 2017.

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

28 Kim Guldstrand Larsen and Bent Thomsen. A modal process logic. In *Proc. of LICS '88*, pages 203–210. IEEE, 1988.

29 Karoliina Lehtinen. A modal μ perspective on solving parity games in quasi-polynomial time. In *Proc. of LICS '18*, pages 639–648. ACM/IEEE, 2018.

30 Claire Loiseaux, Susanne Graf, Joseph Sifakis, Ahmed Bouajjani, and Saddek Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6:1–35, 1995.

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

32 Robin Milner. *Communication and Concurrency.* Prentice Hall, 1989.

33 Antoine Miné. Tutorial on static inference of numeric invariants by abstract interpretation. *Foundations and Trends in Programming Languages*, 4(3-4):120–372, 2017.

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

35 Damien Pous. Complete lattices and up-to techniques. In *Proc. of APLAS '07*, pages 351–366. Springer, 2007. LNCS 4807.

36 Damien Pous. Coinduction all the way up. In *Proc. of LICS'16*, pages 307–316. ACM, 2016.

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

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

39 Davide Sangiorgi and Robin Milner. The problem of "weak bisimulation up to". In W.R. Cleaveland, editor, *Proc. of CONCUR'92*, pages 32–46. Springer, 1992.

40 David A. Schmidt. Binary relations for abstraction and refinement. In *Workshop on Refinement and Abstraction.* Elsevier Electronic, 2000.

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

42 Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(6):303–308, 1996.

43 David Sprunger and Lawrence S. Moss. Precongruences and parametrized coinduction for logics for behavioral equivalence. In *Proc. of CALCO '17*, volume 72 of *LIPIcs*, pages 23:1–23:15. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017.

44 Perdita Stevens and Colin Stirling. Practical model-checking using games. In *Proc. of TACAS '98*, volume 1384 of *Lecture Notes in Computer Science*, pages 85–101. Springer, 1998.

45 Colin Stirling. Local model checking games. In *Proc. of CONCUR '95*, volume 962 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1995.

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

A Comparison to the Bonchi/Pous Algorithm

In a seminal paper [8] Bonchi and Pous revisited the question of checking language equivalence for non-deterministic automata and presented an algorithm based on an up-to congruence technique that behaves very well in practice.

We will here give a short description of this algorithm and then explain how it arises as a special case of the algorithm developed in §6.2.2.

We are given a non-deterministic finite automaton (Q, Σ, δ, F), where Q is the finite set of states, Σ is the finite alphabet, $\delta\colon Q \times \Sigma \to \mathbf{2}^Q$ is the transition function and $F \subseteq Q$ is the set of final states. Note that we omit initial states. Given $a \in \Sigma$, $X \subseteq Q$ we define $\delta_a(X) = \bigcup_{q\in X} \delta(q, a)$.

Given $q_1, q_2 \in Q$, the aim is to show whether q_1, q_2 accept the same language (in the standard sense).

In order to do this, the algorithm performs an on-the-fly determinization and constructs a bisimulation relation $R \subseteq \mathbf{2}^Q \times \mathbf{2}^Q$ on the determinized automaton. This relation has to satisfy the following properties:

- $\{q_1\}\, R\, \{q_2\}$
- Whenever $X_1\, R\, X_2$, then
  - $\delta_a(X_1)\, R\, \delta_a(X_2)$ for all $a \in \Sigma$ (*transfer property*)
  - and $X_1 \cap F \neq \emptyset \iff X_2 \cap F \neq \emptyset$ (*one set is accepting iff the other is accepting*)

Due to the up-to technique there is no need to fully enumerate R. Instead in the second item above, it suffices to show that $\delta_a(X_1)\, c(R)\, \delta_a(X_2)$ where $c(R)$ is the congruence closure of R, i.e., the least relation R' containing R that is an equivalence and satisfies that $X_1\, R\, X_2$ implies $X_1 \cup X\, R\, X_2 \cup X$ (for $X_1, X_2, X \subseteq Q$). A major contribution of [8] is an algorithm for efficiently checking whether two given sets are in the congruence closure of a given relation. Here we will simply assume that this procedure is given and use it as a black box.

We will now translate this into our setting: the lattice is $L = \mathbf{2}^{\mathbf{2}^Q \times \mathbf{2}^Q}$ (the lattice of all relations over the powerset of states) with inclusion as partial order. The basis B consists of all singletons $\{(X_1, X_2)\}$ where $X_1, X_2 \subseteq Q$. That is, we consider the setting of of §6.2.2.

The behaviour map f is given as follows: $f(R) = f^*(R) \cap C$ where

$$\begin{aligned} f^*(R) &= \{(X_1, X_2) \mid (\delta_a(X_1), \delta_a(X_2)) \in R \text{ for all } a \in \Sigma\} \\ C &= \{(X_1, X_2) \mid X_1 \cap F = \emptyset \iff X_2 \cap F = \emptyset\} \end{aligned}$$

We want to solve a single fixpoint equation $R =_\nu f(R)$ where we are interested in the greatest fixpoint. In particular, we want to check whether $(Q_1, Q_2) \in R$ (where $Q_1 = \{q_1\}$, $Q_2 = \{q_2\}$) or alternatively $I = \{(Q_1, Q_2)\} \subseteq R$.

Since we have determinized the automaton, f^* has a left adjoint f_*, given as

$$f_*(R) = \{(\delta_a(X_1), \delta_a(X_2)) \mid (X_1, X_2) \in R, a \in \Sigma\}.$$

Now we can start exploring the game positions. Starting with $I = \{(Q_1, Q_2)\} \subseteq F$, the only move of $\exists$ is to play $\{\{(X_1, X_2)\} \mid (X_1, X_2) \in f_*(I)\}$, then it is the turn of $\forall$ who can choose any singleton set $\{(X_1, X_2)\}$ and one has to explore all those singletons. This continues until one encounters a singleton $\{(X_1, X_2)\} \not\subseteq C$ (which implies that $\exists$ has no move and loses) or

one finds a set $\{(X_1, X_2)\}$ where one can cut off a branch due to the up-to technique – more concretely $(X_1, X_2) \in c(W)$ where W is the collection of all pairs visited so far on all paths and $c(W)$ is its congruence closure. One can conclude that $\exists$ wins if all encountered pairs are in C. This is a straightforward instance of the more general algorithm, enriched with an up-to technique, as explained in §6.2.2.