Information and Computation 301 (2024) 105233

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/yinco

Systems of fixpoint equations: Abstraction, games, up-to techniques and local algorithms ☆

Paolo Baldan [a,*], Barbara König [b,*], Tommaso Padoan [c,*]

[a] *Università di Padova, Italy*
[b] *Universität Duisburg-Essen, Germany*
[c] *Università di Trieste, Italy*

A R T I C L E I N F O

Article history:
Received 30 September 2023
Received in revised form 19 October 2024
Accepted 20 October 2024
Available online 24 October 2024

Keywords:
Fixpoint equation systems
Complete lattices
Parity games
Abstract interpretation
Up-to techniques
Local algorithms
μ-calculus
Bisimilarity

A B S T R A C T

Systems of fixpoint equations over complete lattices, which combine least and greatest fixpoints, often arise from verification tasks such as model checking and behavioural equivalence checking. In this paper we develop a theory of approximation in the style of abstract interpretation, where a system over some concrete domain is abstracted into a system on a suitable abstract domain, ensuring sound and possibly complete over-approximations of the solutions. We also show how up-to techniques, commonly used to simplify coinductive proofs, fit into this framework, interpreted as abstractions. Additionally, we characterise the solution of fixpoint equation systems through parity games, extending prior work limited to continuous lattices. This game-based approach allows for local algorithms that verify system properties, such as determining whether a state satisfies a formula or two states are behaviourally equivalent. We describe a local algorithm, that can be combined with abstraction and up-to techniques to speed up the computation.

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. In fact, in order to express properties of infinite computations, specification logics almost invariably rely on some notion of recursion which leads to the use of fixpoints as key mathematical tool.

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 [38] 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. The most famous example is bisimilarity that can be seen as the greatest fixpoint of a suitable operator over the lattice of binary relations on states (see, e.g., [51]).

☆ Supported by the European Union - NextGenerationEU under the National Recovery and Resilience Plan (NRRP) - Call PRIN 2022 PNRR - Project P2022HXNSC “Resource Awareness in Programming: Algebra, Rewriting, and Analysis”; by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the project number 434050016 (SpeQt).

* Corresponding authors.
E-mail addresses: baldan@math.unipd.it (P. Baldan), barbara_koenig@uni-due.de (B. König), tommaso.padoan@units.it (T. Padoan).

https://doi.org/10.1016/j.ic.2024.105233

In the first part of the paper we propose a theory of approximation for systems of equations in the style of abstract interpretation. The general idea of abstract interpretation [15,16] 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. [26]).

Abstract interpretation has been applied also for the model checking of various kinds of μ-calculi and temporal logics (see, e.g., [29,42,17,53,20,39]). In particular, in [17] a general framework for abstract interpretation of temporal calculi and logics is devised, with the identification of conditions for soundness and completeness of the abstraction. The approach, that in [17] is devised over boolean lattices, can be naturally adapted to the setting of systems of fixpoint equations over complete lattices, 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, and suitable conditions are singled out that ensure the soundness and completeness of the approximation. The approximation theory can be used on a number of verification tasks. We show how to recover and generalise some results on property preserving abstractions for the μ-calculus [42]. We also discuss an application to fixpoint extension of Łukasiewicz logic, considered in [47] 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 $l \sqsubseteq \nu f$.

Up-to techniques have been proposed for "simplifying" proofs [45,52,50,48] and for reducing the search space in verification (e.g., in [9], 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, i.e. $l \sqsubseteq u(l)$ for all elements l) 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 [7].

We have recently shown in [3] 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 [54]. 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 [36], have already been studied in [30,3], here we focus on local algorithms, confining the attention to specific positions. For instance, in the case of the μ-calculus, rather than computing the set of states satisfying some formula φ, one could be interested in checking whether a specific state satisfies or does not satisfy φ. For probabilistic logics, rather than determining the full evaluation of φ, we could be interested in determining the value for a specific state or only in establishing a bound for such a value. Similarly, in the case of behavioural equivalences, rather than computing the full behavioural relation, one could be interested in determining whether two specific states are equivalent. Taking inspiration from backtracking methods for bisimilarity [32] and for the μ-calculus [58,57], we design a local (also called on-the-fly) algorithm for general systems of fixpoint equations (§ 7) and show how these algorithms can be enhanced with up-to techniques.

Related work Our contribution is based on the notion of approximation as formalised in abstract interpretation [15,16]. Due to the intimate connection of Galois connections and closure functions, there is a close correspondence with up-to techniques for enhancing coinduction proofs [48,50], originally developed for CCS [45]. However, as far as we know, recent research has only started to explore this connection: the paper [7] explains the relation between sound up-to techniques and complete abstract domains in the special setting where the semantic function has an adjoint. This adjunction or Galois connection plays a different role than the abstractions: its existence roughly means that the system exhibits some form of "determinism". Transported to our setting, it implies that in the game formalisation the existential player has a unique best move, making the solution of the associated game quite efficient (for a detailed discussion the reader is referred to [4].)

Systems of fixpoint equations largely derive their interest from μ-calculus model-checking [10]. 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 [36], 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 [11,37,41] and nested fixpoints [31,2] have been devised. Instead of improving the complexity bounds, our aim here is to introduce heuristics, based on a local algorithm and up-to functions that are known to achieve good efficiency in practice. In particular, we explain how up-to techniques, which have been traditionally used just for coinductive equivalences, can be naturally generalised to systems of fixpoint equations over a complete lattice with special interest for the integration with a μ-calculus model-checking algorithm.

Abstraction in the setting of μ-calculus and, more generally, temporal logic verification, is a vast topic which has been widely studied. A classical approach consists in exploiting a state-based abstraction over a transition system. This induces an abstract transition relation which overapproximates the concrete behaviour thus ensuring soundness for the verification of ACTL, the universal fragment of CTL [12]. As already mentioned, property-preserving abstraction for the μ-calculus using simulations viewed as Galois connections has been studied in [42] and we will show that this approach can be obtained as an instance of our framework.

In order to overcome the limitation to universal fragments of temporal logics, various approaches have been proposed, based on mixed [20] or modal [27] transition systems, where the abstract system is endowed with two transition relations coping with the different preservation requirements arising from universal and existential operators of the logics.

The abstraction of the μ-calculus along a Galois connection and its soundness is discussed in [5]. A general framework for abstract interpretation of temporal calculi and logics is developed in [17]. 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. As we already mentioned, such results are closely related to those in § 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).

The local algorithm that we propose in § 7.1 for solving arbitrary fixpoint equation systems over general lattices is a generalisation of the algorithm for the modal μ-calculus presented in [57]. The use of assumptions as stopping conditions in the algorithm is reminiscent of parameterized coinduction [56,34], closely related to up-to-techniques, as spelled out in [49]. The presented algorithm is mostly agnostic of the specific setting. Since various other approaches to the local solution of parity games have been proposed in the literature, e.g., [25] and [40], it would be interesting to investigate the possibility of re-using these approaches in our setting and determine those which are most effective. This point is further developed in the concluding section.

Synopsis The rest of the paper is structured as follows. In § 2 we introduce some basic order-theoretical notions and notation used in the paper. In § 3 we introduce systems of equations over complete lattices and their solutions, showing how various verification problems reduce to solutions of systems of equations over suitable lattices. In § 4 we propose a theory of approximation for systems of fixpoint equations over complete lattices in the style of abstract interpretation. In § 5 we show how up-to techniques can be seen as special form of abstraction and thus generalised to systems of fixpoint equations. In § 6 we present the game-theoretical characterisation of the solution of a system of equations and discuss the idea of selections. In § 7 we outline a local algorithm for solving the game, showing how it can take advantage of the up-to techniques for systems of equations. In § 8 we conclude the paper and outline future research. For the sake of readability full proofs and some technical results used only in proofs have been moved to the appendix.

This paper is the full version of [4]. The main novelty is a local algorithm which works for general systems of fixpoint equations (while the algorithm in [4] was restricted to a single greatest fixpoint equation). Moreover, the paper has been extended by including full proofs of technical results and additional examples.

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 2.1 *(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 $\top = \bigsqcap \emptyset$, referred to as *bottom* and *top*, respectively. 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 is X and bottom 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 $f : L \to L$ is *monotone* if for all $l, l' \in L$, if $l \sqsubseteq l'$ then $f(l) \sqsubseteq f(l')$. By Knaster-Tarski's theorem [59, 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 $\nu f = \bigsqcup\{l \mid l \sqsubseteq f(l)\}$.

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 [19]. Given a complete 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 2.2 *(Kleene's iteration [19]). Let L be a complete 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* $(f^\alpha(\top))_\alpha$*.*

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 2.3 *(Galois connection).* Let $(C, \sqsubseteq)$, $(A, \le)$ 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 $\alpha(c) \le a$ iff $c \sqsubseteq \gamma(a)$.

Equivalently, for all $a \in A$ and $c \in C$, (i) $c \sqsubseteq \gamma(\alpha(c))$ and (ii) $\alpha(\gamma(a)) \le a$. 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, $\langle f, i \rangle : L \to f(L)$ is a Galois connection, 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 $i, j \in \underline{n}$, with $i \le j$, 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 $L^X = \{f \mid f : X \to L\}$, 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 decide to consider either the least or the greatest solution. We define systems, their solutions and we provide some examples that will be used as running examples.

Definition 3.1 *(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, mainly for μ-calculus model-checking (see e.g., [13,55,30,3]). In particular, [30,3] 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.

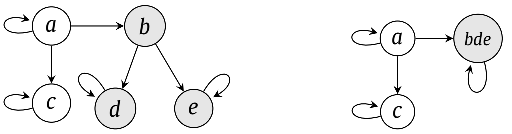
Definition 3.2 *(substitution).* Given a system E of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, an index $i \in \underline{m}$ and $l \in L$ we write $E[x_i := l]$ for the system of $m-1$ equations obtained from E by removing 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 $\boldsymbol{x}'\boldsymbol{x}'' =_{\boldsymbol{\eta}'\boldsymbol{\eta}''} \boldsymbol{f}'\boldsymbol{f}''(\boldsymbol{x}', l, \boldsymbol{x}'')$.

Definition 3.3 *(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:

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

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

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



(a) (b)

$$x_1 =_\nu p \wedge \Box x_1$$
$$x_2 =_\mu x_1 \vee \Diamond x_2$$

(c)

$$x_1 =_\nu \{b, d, e\} \cap \blacksquare_T x_1$$
$$x_2 =_\mu x_1 \cup \blacklozenge_T x_2$$

(d)

Fig. 1. Example of fixpoint equations for a μ-calculus formula interpreted over a transition system.

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

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., $\boldsymbol{s}_{1,m-1} = \boldsymbol{s}_{1,m-1}(s_m)$.

The order of equations matters: changing the order typically leads to a different solution, as shown in the example below.

Example 3.4 *(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

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

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, if we consider the system obtained from E by swapping the two equations, the solution changes and becomes $x = y = \emptyset$.

Example 3.5 *(μ-calculus formulae as fixpoint equations).* We adopt a standard μ-calculus syntax. For fixed disjoint sets *PVar* of propositional variables, ranged over by $x, y, z, \dots$ and *Prop* of propositional symbols, ranged over by $p, q, r, \dots$, each paired with the associated complement $\bar{p}$, formulae are defined by

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

where $p \in \mathit{Prop}$, $x \in \mathit{PVar}$ 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 $\rho\colon \mathit{Prop} \cup \mathit{PVar} \to \mathbf{2}^{\mathbb{S}_T}$ mapping each proposition or propositional variable to the set of states where it holds, we denote by $\|\varphi\|_\rho^T$ the semantics of φ defined as usual (see, e.g., [10]). Explicitly, define the semantic counterparts 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

$$\blacklozenge_R(Y) = \{x \in X \mid \exists y \in Y.\ (x, y) \in R\}$$
$$\blacksquare_R(Y) = \{x \in X \mid \forall y \in X. (x, y) \in R \Rightarrow y \in Y\}$$

and let us write $\blacklozenge_T$ and $\blacksquare_T$ for $\blacklozenge_{\to_T}$ and $\blacksquare_{\to_T}$

Then

$$\|\mathbf{t}\|_\rho^T = \mathbb{S}_T \quad \|p\|_\rho^T = \rho(p) \qquad \|\varphi_1 \wedge \varphi_2\|_\rho^T = \|\varphi_1\|_\rho^T \cap \|\varphi_2\|_\rho^T \quad \|\Diamond\varphi\|_\rho^T = \blacklozenge_T \|\varphi\|_\rho^T$$

$$\|\mathbf{f}\|_\rho^T = \emptyset \quad \|\bar{p}\|_\rho^T = \mathbb{S}_T \setminus \rho(p) \quad \|\varphi_1 \vee \varphi_2\|_\rho^T = \|\varphi_1\|_\rho^T \cup \|\varphi_2\|_\rho^T \quad \|\Box\varphi\|_\rho^T = \blacksquare_T \|\varphi\|_\rho^T$$

$$\|x\|_\rho^T = \rho(x) \qquad \|\eta x.\,\varphi\|_\rho^T = \eta(\lambda S.\, \|\varphi\|_{\rho[x \mapsto S]}^T)$$

where $\rho[x \mapsto S]$ is the environment defined by $\rho[x \mapsto S](x) = S$ and $\rho[x \mapsto S](y) = \rho(x)$ for $y \neq x$.

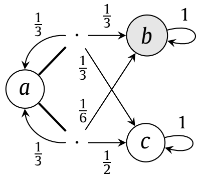
As observed by several authors (see, e.g., [13,55]), a μ-calculus formula can be seen as a system of equations, with an equation for each fixpoint subformula. For instance, consider $\varphi = \mu x_2.((\nu x_1.(p \wedge \Box x_1)) \vee \Diamond x_2)$ that requires that some path eventually reaches a state from which p always holds on all paths. The equational form is given 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, where p holds in the grey states b, d and e.

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.

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).

$\mathbf{0}(x) = 0,\ \mathbf{1}(x) = 1$	(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 PNTS

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

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

(c) Formulae as systems

Fig. 2. Examples of Łukasiewicz μ-calculus formulae for a PNTS, and semantics of μ-terms.

Example 3.6 *(Łukasiewicz μ-terms).* Systems of equations over the real interval $[0, 1]$ have been considered in [47] 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 $x \in \mathit{PVar}$ 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.

Then, each Łukasiewicz μ-term, in an environment $\rho : \mathit{PVar} \to [0, 1]$, 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 [47] 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 3.7 *(Łukasiewicz μ-calculus).* The Łukasiewicz μ-calculus, as defined in [47], extends the Łukasiewicz μ-terms with propositions and modal operators. The syntax is as follows:

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

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 [44,35] and it allows to encode PCTL [6]. 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 (PNTS)*[1] $N = (\mathbb{S}, \to)$ where $\to\ \subseteq \mathbb{S} \times \mathcal{D}(\mathbb{S})$ is the transition relation. An example of PNTS 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 $\rho : \mathit{Prop} \cup \mathit{PVar} \to (\mathbb{S} \to [0, 1])$ mapping each proposition or propositional variable to a real-valued function over the states, the semantics of φ is a function $\|\varphi\|_\rho^N : \mathbb{S} \to [0, 1]$ 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 formulae 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 formulae $\varphi = \mu x_2.(\nu x_1.(p \odot \Diamond x_1) \oplus \Diamond x_2)$ 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 $\|\varphi'\|_\rho(a) = \frac{1}{4}$.

[1] PNTS are the same as (unlabelled) Markov decision processes, as observed also in [47].

Example 3.8 *((bi)similarity over transition systems).* For defining (bi)similarity uniformly with the previous Example 3.5 on the μ-calculus, we work on unlabelled transition systems with atoms $T = (\mathbb{S}, \rightarrow, A)$ 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 $T = (\mathbb{S}, \rightarrow, A)$, consider the lattice of relations on $\mathbb{S}$, ordered by subset inclusion, 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 $sim_T : \mathsf{Rel}(\mathbb{S}) \rightarrow \mathsf{Rel}(\mathbb{S})$, defined by

$$sim_T(R) = \left\{(x, y) \in R \mid \forall a \in A. (x \in a \Rightarrow y \in a) \wedge \forall x \rightarrow x'. \exists y \rightarrow y'. (x', y') \in R\right\}$$

In other words $\precsim_T$ 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 the relation $\{(c, a), (a, b), (b, d), (d, e), (e, b)\}$.

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 least equivalence such that $b \sim_T d \sim_T e$.

4. Approximation for systems of fixpoint equations

In this section we propose a theory of approximation for systems of fixpoint equations over complete lattices. The general setup is borrowed from abstract interpretation [15–17], 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. In particular [17] deals with temporal specification logics in the style of the μ-calculus, interpreted over boolean lattices, where least and greatest fixpoints are naturally nested.

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, where the intuition is that a larger element is less precise than a smaller one. Let $f^C : C \rightarrow C$ and $f^A : A \rightarrow A$ 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 $\gamma : A \rightarrow C$, 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 in 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 [15,16,46]) 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., [17, Proposition 15], which states the dual result). For clarity we state this result explicitly in the appendix (see Lemma A.1(1)).

Then we can suitably combine the conditions for least and greatest fixpoints, similarly to what is done in [17, Lemma 36]. We will allow a different concretisation function for each equation.

Theorem 4.1 *(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 $\gamma_i : A \rightarrow C$ 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 [19] 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 fact, already exploited in [17, Theorem 40] for boolean lattices, leads to the following result on general complete lattices, where, besides the soundness conditions, we also make explicit the completeness conditions.

Theorem 4.2 *(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 $\langle \alpha_i, \gamma_i \rangle : C \rightarrow A$ 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., [26]) 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 in overapproximations. Together with soundness, it leads to $\boldsymbol{s}^{\boldsymbol{C}} = \boldsymbol{\gamma}^\times(\boldsymbol{s}^{\boldsymbol{A}})$, a very strong property which is not meaningful in program analysis. 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.

As in the standard abstract interpretation framework, dealing with Galois connections, we can consider the best (smallest) sound abstraction of the concrete system in the abstract domain.

Definition 4.3 *(best abstraction).* Let $(C, \sqsubseteq)$ and $(A, \leq)$ be complete lattices, let E_C be a system of m equations over C of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$. 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 $i \in \underline{m}$. The *best abstraction* of E_C is the system over A defined by $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}^\#(\boldsymbol{x})$, where $\boldsymbol{f}^\# = \boldsymbol{\alpha}^\times \circ \boldsymbol{f} \circ \boldsymbol{\gamma}^\times$.

Standard arguments show that $\boldsymbol{f}^\#$ is a sound abstraction of $\boldsymbol{f}$ over A, and it is the smallest one. Moreover, sound abstract operators can be obtained compositionally out of basic ones, preserving soundness, but not necessarily completeness.

Example 4.4 *(abstraction for μ-calculus).* The paper [42] 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 [42], 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 [42, 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 [14]) 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 [42, 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 $\alpha(\|\varphi\|_\rho^{T_C}) \subseteq \|\varphi\|_{\alpha \circ \rho}^{T_A}$. 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 4.2, 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 4.2 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 by definition.

In order to extend the logic by including negation on propositions, in [42], 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 4.2 for negated propositions.

Our theory naturally suggests generalisations of [42]. E.g., by (the dual of) Theorem 4.1, continuity and strictness of the abstraction α are sufficient to retain the results, hence one can deal with an abstraction not being an adjoint, thus going beyond ordinary simulations.

Example 4.5 *(abstraction for Łukasiewicz μ-terms).* For Łukasiewicz μ-terms, as introduced in Example 3.6, 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 **0**, **1**, $\sqcup$, $\sqcap$, while it is just sound for the remaining operators.

Also observe that, when moving from $[0,1]_{/n}$ to $[0,1]_{/m}$ with m a multiple of n, the abstraction "improves", i.e., the abstract solution gets closer to the concrete one. This immediately follows by noticing that the inclusion of $[0,1]_{/m}$ into $[0,1]_{/n}$ is a Galois insertion.

For instance, the system in Example 3.6 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 4.6 *(abstraction for Łukasiewicz μ-calculus).* Observe that when dealing with Łukasiewicz μ-calculus over some probabilistic transition system $N = (\mathbb{S}, \rightarrow)$, we can lift the Galois insertion above to $[0,1]^{\mathbb{S}}$. Define $\alpha_n^{\rightarrow} : [0,1]^{\mathbb{S}} \to [0,1]^{\mathbb{S}}_{/n}$ by letting, $\alpha_n^{\rightarrow}(v) = \alpha_n \circ v$ for $v \in [0,1]^{\mathbb{S}}$. Then $\langle \alpha_n^{\rightarrow}, \gamma_n^{\rightarrow} \rangle : [0,1]^{\mathbb{S}} \to [0,1]^{\mathbb{S}}_{/n}$ (where $\gamma_n^{\rightarrow}$ 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 3.7. Recall that the exact solution is $x_2(a) = 0.25$. With abstraction α_{10} we get $x_2(a) = 0.3$, with α_{30} we get a better (over-) approximation, i.e., $x_2(a) = 0.2\bar{6}$.

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 [45,52], they have been later studied in the setting of complete lattices [48,49]. Some recent work [7] started the exploration of the relation between up-to techniques and abstract interpretation. Roughly, they work in a setting where the semantic function 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, which is discussed in [4, Appendix A].

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 3.8. 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 $\{(s_1, s_2)\} \subseteq R$. The use of up-to functions is meant to ease this task.

Definition 5.1 *(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 in 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 [48].

Definition 5.2 *(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. We first consider the case in which u is a closure (i.e., extensive and idempotent). Then $u(L)$ 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 do not only deal with greatest but also with least fixpoints.

Lemma 5.3 *(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 $\mu f = i(\mu f_{|u(L)}) = \mu(f \circ u)$.*

$$\begin{tikzcd} L \arrow[loop above, "f"] \arrow[loop left, "f\circ u"] \arrow[r, shift right, "u"'] & u(L) \arrow[l, shift right, "i"'] \arrow[loop right, "f_{|u(L)}"] \end{tikzcd}$$

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 (hence sound and complete) by taking the least closure $\bar{u}$ above u.

Definition 5.4 *(least upper closure).* Let L be a complete lattice and let $u : L \to L$ be a monotone function. We let $\bar{u} : L \to L$ be the function defined by $\bar{u}(x) = \mu(\hat{u}_x)$ where $\hat{u}_x(y) = u(y) \sqcup x$.

Lemma 5.5 *(properties of $\bar{u}$). Let $u : L \to L$ be a monotone function. Then*

1. *$\bar{u}$ is the least closure larger than u;*
2. *if u is f-compatible then $\bar{u}$ is;*
3. *if u is continuous and strict then $\bar{u}$ is.*

The least upper closure above a given function has been considered already in [18], with a slightly different construction.

Using Lemmas 5.3 and 5.5, whenever u is a compatible up-to function for f, we have that $\bar{u}$ is a sound and complete up-to function for f. The soundness of u then immediately follows.

Corollary 5.6 *(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 [48] 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 [49] 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, as discussed in § 7.2, 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, we can easily move from a single equation to systems of equations. As in the case of abstractions, a different up-to function is allowed for each equation.

Definition 5.7 *(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 $\boldsymbol{u}$, such that each $u_i : L \to L$ satisfies compatibility ($\boldsymbol{u}^\times \circ \boldsymbol{f} \sqsubseteq \boldsymbol{f} \circ \boldsymbol{u}^\times$) and u_i is continuous and strict for each $i \in \underline{m}$ such that $\eta_i = \mu$.

We can then generalise Corollary 5.6 to systems of equations.

Theorem 5.8 *(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 5.9 *(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 4.4, namely $u_{\precsim} = \blacklozenge_{\succsim}$. 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 4.4 which has been already observed to coincide with soundness in the sense of Theorem 4.2 for the operators of the $\mu\Diamond$-calculus. Concerning propositional variables, in Example 4.4, 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 $\rho : Prop \to \mathbf{2}^{\mathbb{S}}$, we must have $\rho(p) = u_{\precsim}(\rho(p))$, i.e., $\rho(p)$ is 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 3.5 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 § 5.2) 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 5.10 *(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 [50]. 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 $bis_T : \mathsf{Rel}(\mathbb{S}) \to \mathsf{Rel}(\mathbb{S})$ of which bisimilarity is the greatest fixpoint (see Example 3.8).

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 5.6).

We conclude this section by providing an alternative view on the integration of up-to functions into systems of fixpoint equations, without the explicit need for closures nor extensiveness of the up-to functions. Let E be a system of m equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ over a complete lattice L and let $\boldsymbol{u}$ be a compatible tuple of up-to functions for E. By Theorem 5.8 we have that the system $E_{\bar{\boldsymbol{u}}}$ with equations $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\bar{\boldsymbol{u}}^{\times}(\boldsymbol{x}))$ has the same solution as E. Now, since $\bar{\boldsymbol{u}}$ is a tuple of functions obtained as least fixpoints (see Definition 5.4), the system $E_{\bar{\boldsymbol{u}}}$ can be "equivalently" written as the system of $2m$ equations that we denote by $d(E, \boldsymbol{u})$, defined as follows:

$$\boldsymbol{y} =_{\boldsymbol{\mu}} (\boldsymbol{u}^{\times}(\boldsymbol{y})) \sqcup \boldsymbol{x}$$
$$\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{y})$$

More precisely, we can show the following result.

Theorem 5.11 *(preserving solutions with up-to). Let E be a system of m equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ over a complete lattice L. Let $\boldsymbol{u}$ be an m-tuple of up-to functions compatible for E. The solution of the system $d(E, \boldsymbol{u})$ is $sol(d(E, \boldsymbol{u})) = (sol(E), sol(E))$.*

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 [3]. 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.

Then, we introduce a device that allows to improve the general efficiency of the game by restricting the number of possible moves, without any loss of information.

6.1. Game characterization

Parity games [21,61] are two-player zero-sum games of perfect information played on directed graphs. Nodes, also known as positions, of the game graph are partitioned into two sets, depending on the player controlling the node. Starting from an initial position, the game is played by moving a token from a node to another along one of the outgoing edges, chosen by

Table 1
The game on the powerset of the basis.

Position	Player	Moves
(b, i)	$\exists$	$\boldsymbol{X}$ such that $b \sqsubseteq f_i(\bigsqcup \boldsymbol{X})$
$\boldsymbol{X}$	$\forall$	(b', j) such that $b' \in X_j$

the player controlling the current position. Thus, a play in the game is a (possibly infinite) path in the graph. Each position is labelled by a priority, expressed by a natural number. A finite play is simply won by the player who moved last. Instead, the winner of an infinite play is established based on the maximal priority appearing infinitely often along the play. The name of parity games originates from the fact that usually the parity of the priority is used to determine such winner. Even priorities are associated with one player, while odd ones with the other, then the winner is decided depending on whether the maximal priority appearing infinitely often is even or odd. A strategy for a player is a function which assigns to each position controlled by that player one of the positions connected by an outgoing edge, that is, a possible move of the player. A strategy for a player is winning from a node (or a subset of nodes) if every play starting from such a node is won by that player, assuming the player follows the strategy, independently from the moves of the other player. A crucial fact is that, for every parity game, there is a unique bipartition of the game positions into those from which one player has a winning strategy and those from which the other does (see, e.g., [21]).

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

Definition 6.1 *(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 $\boldsymbol{x} =_{\eta} \boldsymbol{f}(\boldsymbol{x})$, the corresponding *powerset game* is a parity game, with two players: an existential player $\exists$ and a universal player $\forall$. It is 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 $\boldsymbol{X} = (X_1, \dots, X_m) \in (\mathbf{2}^{B_L})^m$.
- 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.

Note that, differently from what happens for standard parity games, the priority associated with the positions of player $\exists$ is not only a number, the index of the equation in the system, but also the indication of the corresponding kind of fixpoint (least or greatest). It is the latter information, rather than the parity, which determines the winner of an infinite play. An alternative, more in line with the literature on parity games would be to define priorities in a way that their parity determines the kind of fixpoint of the associated equation, even for ν and odd for μ. Concretely this would mean giving a node (b, i) priority $2i$ if $\eta_i = \nu$ and priority $2i + 1$ if $\eta_i = \mu$. One could also reduce the number of priorities using the same priority for groups of consecutive equations that have the same fixpoint operator. We decided to opt for the first notation to be coherent with previous work in [30,3].

The game is meant to be used to decide whether a specified element of the basis of the lattice is below the solution of a fixpoint equation of the system. Indeed, as shown in the theorem below, this reduces to determine which player has a winning strategy starting from the corresponding position in the game.

Theorem 6.2 *(correctness and completeness). Let E be a system of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}(\boldsymbol{x})$ 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).*

Interestingly, the correctness and completeness of the game can be proved by exploiting the soundness and completeness of the game over continuous lattices from [3] and the results about abstraction in § 4. The game for continuous lattices in [3] is very similar to the one described in Table 1. Positions of player $\exists$ are still pairs (b, i) where b is an element of the basis and $i \in \underline{m}$ an equation index, while positions of player $\forall$ are tuples of elements of the lattice $\boldsymbol{l} \in L^m$, the intuition being that a position $\boldsymbol{X} \in (\mathbf{2}^{B_L})^m$ of the powerset game corresponds to a position $\bigsqcup \boldsymbol{X} \in L^m$ of the game over continuous lattices. Moreover the order relation $\sqsubseteq$ is replaced by the way-below relation $\ll$ in the continuous lattice in a way that the condition defining the moves for player $\exists$ translates from $b \sqsubseteq f_i(\bigsqcup \boldsymbol{X})$ to $b \ll f_i(\boldsymbol{l})$, while the condition $b' \in X_j$ for $\forall$-moves becomes $b' \ll l_j$.

The crucial observation is that the described correspondence between the two games can be interpreted as a Galois insertion between L and the powerset lattice of its basis (which is algebraic hence continuous). The Galois connection is $\langle \alpha, \gamma \rangle : \mathbf{2}^{B_L} \to L$ 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" along such an insertion to a system of equations over the powerset of the basis $\mathbf{2}^{B_L}$, in a way that the system in L can be seen as a sound and complete abstraction of the one in $\mathbf{2}^{B_L}$.

As discussed later in § 7 (Example 7.1), if we instantiate the game to the setting of standard μ-calculus model-checking, we obtain an alternative encoding of μ-calculus into parity games.

Example 6.3. We provide a simple example illustrating the game. Consider the infinite lattice $L = \mathbb{N} \cup \{\omega, \omega+1\}$ (where $n \leq \omega \leq \omega+1$ for every $n \in \mathbb{N}$) with basis $B_L = L$. Furthermore let $f : L \to L$ be a monotone function with $f(n) = n+1$ for $n \in \mathbb{N}$ and $f(\omega) = \omega$, $f(\omega+1) = \omega+1$. Hence $\mu f = \omega$.

We set $b = \omega$ and attempt to show via the game that $b \leq \mu f$, by exhibiting a winning strategy for $\exists$. Note that since we are dealing with a μ-equation, in order to win $\exists$ must ensure that $\forall$ eventually has no moves left. Since there is only one fixpoint equation, we omit the indices. Starting with $b = \omega$, $\exists$ plays $X = \mathbb{N}$, which is a valid move since $\omega \leq f(\bigsqcup X) = f(\omega)$. Now $\forall$ has to pick some $n \in X$. In the next move, $\exists$ can play $X = \{n-1\}$, which means that $\forall$ picks $n-1$. Hence we obtain a descending chain, leading to 1, which can be covered by $\exists$ by choosing $X = \emptyset$, since $1 \leq f(\bigsqcup \emptyset) = f(0)$. Now $\forall$ has no moves left and $\exists$ wins. Observe that in every position $n \in \mathbb{N}$, $\exists$ could have also played the set $X = \{m \in \mathbb{N} \mid m \leq n-1\}$ with the same outcome.

Instead, if we start from $b = \omega+1 \not\leq \mu f$, then $\exists$ has no winning strategy since she has to play a set X that contains $\omega+1$. Then player $\forall$ can reply by choosing $\omega+1$ and the game will continue forever. This is won by $\forall$ since we are dealing with a μ-equation.

6.2. 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. The idea is formalised by introducing a notion of selection, similarly to what has been done in [3]. The intuition is very simple: in the game for continuous lattices in [3], player $\exists$ in position (b, i) has to play a tuple $\boldsymbol{l} \in L^m$ such that $f_i(\boldsymbol{l})$ is (way-)above the basis element b. Clearly, when $\boldsymbol{l}$ is a move for player $\exists$, then, by monotonicity of f_j, every $\boldsymbol{l}'$ such that $\boldsymbol{l} \sqsubseteq \boldsymbol{l}'$ is also a move for $\exists$. However, since from position $\boldsymbol{l}$ player $\forall$ performs a move (b', j) such that b' is (way-)below l_j, taking a smaller $\boldsymbol{l}$ reduces the possible moves of player $\forall$ and thus is a preferable choice for player $\exists$. On the basis of the above considerations, the idea behind selections is to restrict to a subset of moves for player $\exists$ which are the most favourable, without altering the winner of the game.

In order to apply the same idea to the powerset game, with the aim of suitably ordering the moves of player $\exists$ which are tuples of sets, we introduce the so-called *Hoare preorder* [1]. Given a lattice L, the *Hoare preorder* $\sqsubseteq_H$ on $\mathbf{2}^{B_L}$, defined by letting, for $X, Y \in \mathbf{2}^{B_L}$, $X \sqsubseteq_H Y$ if $\forall x \in X. \exists y \in Y. x \sqsubseteq y$. Observe that $\sqsubseteq_H$ is not antisymmetric in general, e.g., if we consider the lattice $\mathbb{N}^\omega$ of natural numbers extended with a top element ω, then $\{1,3\} \sqsubseteq_H \{2,3\} \sqsubseteq_H \{1,3\}$. 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 $\exists$, which are tuples in $(\mathbf{2}^{B_L})^m$, can be ordered by the pointwise extension of $\sqsubseteq_H$, denoted $\sqsubseteq_H^\wedge$, defined by $\boldsymbol{X} \sqsubseteq_H^\wedge \boldsymbol{Y}$ when $X_i \sqsubseteq_H Y_i$ for all $i \in \underline{m}$. Note that the set of moves $\mathbf{E}(b, i)$, from a position (b, i), is always upward-closed w.r.t. $\sqsubseteq_H^\wedge$. In fact, since $X \sqsubseteq_H Y$ implies $\bigsqcup X \sqsubseteq \bigsqcup Y$, given $\boldsymbol{X} \sqsubseteq_H^\wedge \boldsymbol{Y}$ such that $\boldsymbol{X} \in \mathbf{E}(b, i)$, then by monotonicity of f_i it holds that $b \sqsubseteq f_i(\bigsqcup \boldsymbol{X}) \sqsubseteq f_i(\bigsqcup \boldsymbol{Y})$, hence $\boldsymbol{Y} \in \mathbf{E}(b, i)$.

Now, the crucial observation is that, given two moves for $\exists$, say $\boldsymbol{X}, \boldsymbol{Y} \in (\mathbf{2}^{B_L})^m$, if $\boldsymbol{X} \sqsubseteq_H^\wedge \boldsymbol{Y}$ then we can safely ignore $\boldsymbol{Y}$ without affecting the winner of the game. This fact, formalised in Theorem 6.5, justifies the notion of selection.

Definition 6.4 *(selection).* Let E be a system of m equations over a complete lattice L, with basis B_L. A *selection* for E is a function $\sigma : (B_L \times \underline{m}) \to \mathbf{2}^{(\mathbf{2}^{B_L})^m}$ such that, for all $b \in B_L$ and $i \in \underline{m}$, it holds $\uparrow_H \sigma(b, i) = \mathbf{E}(b, i)$, where $\uparrow_H$ is the upward-closure with respect to $\sqsubseteq_H^\wedge$.

Observe that requiring the set $\mathbf{E}(b, i)$ of moves of $\exists$ from position (b, i) to be the upward-closure of $\sigma(b, i)$ with respect to $\sqsubseteq_H^\wedge$ is equivalent to requiring that $\sigma(b, i) \subseteq \mathbf{E}(b, i)$ and for each $\boldsymbol{X} \in \mathbf{E}(b, i)$ there exists $\boldsymbol{Y} \in \sigma(b, i)$ such that $\boldsymbol{Y} \sqsubseteq_H^\wedge \boldsymbol{X}$. Since $\mathbf{E}(b, i)$ is upward-closed wrt. $\sqsubseteq_H^\wedge$, selections always exist.

The crucial fact is that Theorem 6.2 continues to hold, even if we restrict the moves of player $\exists$ to those prescribed by a selection.

Theorem 6.5 *(game with selections). 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 $\boldsymbol{s}$, and let σ be a selection for E. 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) in the game restricted to the selection σ.*

Example 6.6. Selections are of great importance for pruning the state space of the game, in many cases avoiding the exponential blow-up suggested by the game rules, where $\exists$ can play any tuple of subsets of the basis.

As an example, consider a game played on a formula of the μ-calculus as defined in Example 3.5. Assuming that variables are $x_1, \ldots, x_m$ and we have a standard encoding of fixpoint operators into an equation system, the remaining relevant operators are $\Diamond$, $\Box$, conjunction and disjunction.

Remember that we are working with unlabelled transition systems $T = (\mathbb{S}_T, \to_T)$, where the lattice corresponds to $\mathbf{2}^{\mathbb{S}_T}$ and basis elements (positions of $\exists$) are the singleton sets $\{s\}$, $s \in \mathbb{S}_T$.

Assume for simplicity that the equation system has been flattened, with one equation for each operator. Hence for the $\Diamond$-operator we have a semantic equation $x_i =_{\eta_i} \blacklozenge x_j$ and we define the corresponding selection

$$\sigma(\{s\}, i) = \{(\emptyset, \dots, \emptyset, \underbrace{\{t\}}_{\text{pos. } j}, \emptyset, \dots, \emptyset) \mid s \to_T t\}.$$

After this move by $\exists$, the player $\forall$ is forced to take the chosen singleton $\{t\}$.

Instead, for the $\Box$-operator we have a semantic equation $x_i =_{\eta_i} \blacksquare x_j$ and we define the corresponding selection

$$\sigma(\{s\}, i) = \{(\emptyset, \dots, \emptyset, \underbrace{\{t \mid s \to_T t\}}_{\text{pos. } j}, \emptyset, \dots, \emptyset)\}.$$

After this move by $\exists$, the player $\forall$ can choose any singleton $\{t\}$ such that $s \to_T t$.

In the case of conjunction the equation becomes $x_i =_{\eta_i} x_j \cap x_k$ with the selection function is defined as

$$\sigma(\{s\}, i) = \{(\emptyset, \dots, \emptyset, \underbrace{\{s\}}_{\text{pos. } j}, \emptyset, \dots, \emptyset, \underbrace{\{s\}}_{\text{pos. } k}, \emptyset, \dots, \emptyset)\}$$

and $\exists$ is forced to play this tuple.

In the case of disjunction the equation becomes $x_i =_{\eta_i} x_j \cup x_k$. The selection function is defined as

$$\sigma(\{s\}, i) = \{(\emptyset, \dots, \emptyset, \underbrace{\{s\}}_{\text{pos. } j}, \emptyset, \dots, \emptyset), (\emptyset, \dots, \emptyset, \underbrace{\{s\}}_{\text{pos. } k}, \emptyset, \dots, \emptyset)\}$$

and $\exists$ can play either the first or second tuple.

Hence there is no blowup in the intermediate steps and this results in a game similar and comparable in size to the standard parity game obtained by encoding a μ-calculus formula (cf. [10]).

Remark 6.7. Observe that whenever L satisfies the infinite distributive law (i.e., $\forall l \in L. \forall X \subseteq L. l \sqcap \bigsqcup X = \bigsqcup\{l \sqcap x \mid x \in X\}$), and the basis B_L comprises only completely join-irreducible elements, then $X \sqsubseteq_H Y$ is the same as $\bigsqcup X \sqsubseteq \bigsqcup Y$ (while, in general, it would only hold that $X \sqsubseteq_H Y$ implies $\bigsqcup X \sqsubseteq \bigsqcup Y$.)

The assumptions above are satisfied, in particular, when L is a powerset $\mathbf{2}^X$ and the basis consists of the singletons $B_L = \{\{x\} \mid x \in X\}$ (see Example 7.1 on the μ-calculus). In this case $\sqsubseteq_H$ is simply subset inclusion.

Interestingly enough, even if for general lattices $\sqsubseteq_H$ and the preorder based on suprema differ, the latter can safely replace the former whenever the system of equations contains only ν-equations (this is the special case explored in [4]).

The rest of the section is devoted to singling out situations in which there is an optimal choice for selections, which restricts the moves of player $\exists$ as much as possible. This is made rigorous by introducing an order on selections.

Definition 6.8 *(order on selections).* Let E be a system of m equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, over a complete lattice L, with basis B_L. Given two selections σ, σ', we write $\sigma \subseteq_H \sigma'$ if for all $b \in B_L$, $i \in \underline{m}$, and $\boldsymbol{X} \in \sigma(b, i)$, there exists $\boldsymbol{Y} \in \sigma'(b, i)$ such that $\boldsymbol{X} \subseteq^{\wedge} \boldsymbol{Y}$.

Unfortunately the existence of a minimal selection w.r.t. $\subseteq_H$ is not always guaranteed. We will see that under the assumption that the lattice has no infinite ascending or descending chains, not only a minimal, but even a least selection is guaranteed to exist. Such selection is the one producing both the fewest and the smallest moves. We start proving a result which guarantees that every move in the least selection is finite.

Lemma 6.9 *(finite moves). Let E be a system of m equations over a complete lattice L without infinite ascending chains. For every position $(b, i) \in B_L \times \underline{m}$ and move $\boldsymbol{X} \in \mathbf{E}(b, i)$, there exists a finite move $\boldsymbol{Y} \in \mathbf{E}(b, i)$ such that $\boldsymbol{Y} \subseteq^{\wedge} \boldsymbol{X}$.*

Proposition 6.10 *(least selection). Let E be a system of m equations over a complete lattice L with finite height. Then, there exists a unique selection σ such that $\sigma \subseteq_H \sigma'$ for all selections σ'.*

Intuitively, the least selection is obtained by progressively refining the set of all possible moves $\mathbf{E}(b, i)$. First, by Lemma 6.9 we can safely restrict only to finite moves, whose upward-closure w.r.t. $\sqsubseteq_H^{\wedge}$ is again $\mathbf{E}(b, i)$, since $\subseteq^{\wedge}$ implies $\sqsubseteq_H^{\wedge}$. Then, we restrict to only the moves $\boldsymbol{X}$ whose downward-closure $\downarrow \boldsymbol{X}$ is minimal w.r.t. $\subseteq^{\wedge}$, which exist because we assumed also the absence of infinite descending chains (hence $\downarrow \boldsymbol{X}$ is finite). The upward-closure w.r.t. $\sqsubseteq_H^{\wedge}$ is preserved after this reduction for the same reason of the previous step, observing that $\downarrow \boldsymbol{X} \equiv_H \boldsymbol{X}$. Finally, in each component of each remaining move it is enough to consider only the maximal elements w.r.t. $\sqsubseteq$, in fact the upward-closure w.r.t. $\sqsubseteq_H^{\wedge}$ is again preserved. Note that the maximal elements exist because the moves are now all finite. From the steps above we can deduce that the resulting selection is the only minimal one w.r.t. $\subseteq_H$, hence the least.

Example 6.11. Consider the powerset game that we presented in Example 6.3 for the lattice $L = \mathbb{N} \cup \{\omega, \omega+1\}$, with basis $B_L = L$, and the function $f(n) = n+1$ for $n \in \mathbb{N}$, $f(\omega) = \omega$, $f(\omega+1) = \omega+1$. In the example the moves played by $\exists$ were as follows: $\{n-1\}$ from position $n \in \mathbb{N}$, $\mathbb{N}$ from position ω, and $\{\omega+1\}$ from position $\omega+1$. There we also mentioned that, from $n \in \mathbb{N}$, $\exists$ could have equivalently played the set $\{m \in \mathbb{N} \mid m \leq n-1\}$. We now show that both kinds of moves correspond to a selection, however none of them is minimal w.r.t. the order $\subseteq_H$. Indeed, in this case L has infinite height and there is actually no least selection. Let σ be the first selection, defined as $\sigma(n) = \{\{n-1\}\}$ for $n \in \mathbb{N}$, $\sigma(\omega) = \{\mathbb{N}\}$ and $\sigma(\omega+1) = \{\{\omega+1\}\}$.[2] The other selection σ' is defined as $\sigma'(n) = \{\{m \in \mathbb{N} \mid m \leq n-1\}\}$ for $n \in \mathbb{N}$, $\sigma'(\omega) = \{\mathbb{N}\}$ and $\sigma'(\omega+1) = \{\{\omega+1\}\}$. It is easy to see that both σ and σ' are valid selections, i.e., $\uparrow_H \sigma(b) = \uparrow_H \sigma'(b) = \mathbf{E}(b)$ for all $b \in B_L$. Moreover, the two selections differ only on natural numbers $n \in \mathbb{N}$, and, in particular, the unique $X \in \sigma(n)$ is always included in the unique $X' \in \sigma'(n)$, hence $\sigma \subseteq_H \sigma'$. However, σ is not minimal since there exist smaller selections, infinitely many in fact. For instance, take σ'' defined as σ except for $\sigma''(\omega) = \{2\mathbb{N}\}$ the set of even numbers. Clearly σ'' is still a valid selection and $\sigma'' \subseteq_H \sigma$, and there are even smaller ones, however none of them is minimal w.r.t. $\subseteq_H$.

7. Local algorithm for solving the game

The game characterisation is exploited for developing an algorithm for solving the game and thus the associated verification problem. The algorithm has a "local flavour", i.e., it tries to identify the winner of the game at a specific position limiting as much as possible the exploration of the system. We first present the local algorithm and then discuss how it can take advantage of the up-to techniques for systems of equations that we devised in § 5.2.

Local algorithms, also referred to as lazy, on-the-fly, or global caching, have in practical cases proved to be a successful and efficient way to solve the corresponding problems. Indeed, also for the solution of parity games, even though their global counterparts have generally lower theoretical runtime upper bounds, local techniques have shown to be especially useful in practical use cases, which usually admit small solutions, where local algorithms are able to achieve faster runtime with less computation. This has also been demonstrated by the implementations of local procedures in tools such as PGSolver [23,24] (https://github.com/tcsprojects/pgsolver), a community toolsuite of choice for the solution of parity games.

The algorithm presented herein extends the one that was introduced in [4] to the general case of a system of equations. This gives us a technique for determining whether a lattice element is below a component of the solution. The idea consists in computing only the information needed for the local problem of interest, along the lines of other local algorithms developed for bisimilarity [33] and for μ-calculus model checking [57]. In particular, our algorithm arises as a natural generalisation of the one in [57] to the setting of powerset games (see Definition 6.1). The algorithm shares also some common ideas with the parity game solving techniques presented in past works [60,25] based on global and local, respectively, strategy improvement. All three solving procedures use the underlying discrete graph structure and nodes priorities to evaluate plays and strategies in a parity game, which is implicitly or explicitly reformulated (called a finite *cycle-domination game* in [60]) so that even infinite plays can be treated as finite objects. In particular, the algorithms in [60,25] use such valuations to construct and iteratively improve players' strategies, in the case of [25] limited to the locally explored part of the game graph. On the other hand, our algorithm does not really record nor explicitly state strategies for the players, although some relevant parts of them can be extracted from the current computed local information, and are in truth used to prove the correctness of the method. Recently, in [40], the use of local algorithms for the solution of symbolic general parity games has also been explored. The cited local algorithms and aforementioned tools already provide a solid, and possibly faster, apparatus for the solution of general parity games. However, our aim is not just to solve the parity game associated with a system of fixpoint equations, but also support the use of abstraction and up-to techniques (as explained later). To do this, the inherent properties of the powerset game and setting must also be exploited. Thus, we resorted to a new local algorithm with such capability. Nevertheless, it is worth mentioning that the algorithm presented below is still applicable to general parity games, as it will be clear from its description and the adopted notation.

We start by fixing some conventions and notation which will be useful for describing the algorithm.

Notation For the rest of the section, L denotes a complete lattice, with a basis B_L, and E is a system of m fixpoint equations over L of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$, with solution $\boldsymbol{s} \in L^m$.

A generic *player*, that can be either $\exists$ or $\forall$, is usually represented by the upper case letter P. The opponent of player P is denoted by $\overline{P}$, i.e., $\overline{\forall} = \exists$, $\overline{\exists} = \forall$. The set of all *positions* of the game is denoted by $Pos = Pos_\exists \cup Pos_\forall$, where $Pos_\exists = B_L \times \underline{m}$, ranged over by (b,i) is the set of positions controlled by $\exists$, and $Pos_\forall = (\mathbf{2}^{B_L})^m$, ranged over by $\boldsymbol{X}$ is the set of positions controlled by $\forall$. A generic position is usually denoted by the upper case letter C and we write $\mathsf{P}(C)$ for the player controlling the position C.

Given a position $C \in Pos$, the possible moves for player $\mathsf{P}(C)$ are indicated by $\mathsf{M}(C) \subseteq Pos$. In particular, if $C \in Pos_\exists$ then $\mathsf{M}(C) \subseteq Pos_\forall$, otherwise $\mathsf{M}(C) \subseteq Pos_\exists$. A function $\mathsf{i} : Pos \to (\underline{m} \cup \{0\})$ maps every position to a *priority*, which, for positions

[2] We recall that since there is only one equation, we omit indices and tuples. The explicit definition would have been $\sigma(n,1) = \{(\{n+1\})\}$, etc.

(b, i) of player $\exists$ is the index i (with $1 \le i \le m$), while it is 0 for positions of $\forall$. With this notation, the winning condition can be expressed as follows:

- A finite play is won by the player who moved last.
- An infinite play, seen as a sequence of positions $(C_1, C_2, \ldots)$, is won by player $\exists$ (resp. $\forall$) if there exists a priority $h \in \underline{m}$ such that $\eta_h = \nu$ (resp. μ), the set $\{j \mid \mathrm{i}(C_j) = h\}$ is infinite and the set $\{j \mid \mathrm{i}(C_j) > h\}$ is finite.

Note that the largest index which occurs infinitely often cannot be 0 since only positions of player $\forall$ have priority 0 and players alternate during the game. Hence the highest priority will always be a priority of player $\exists$. Hence this condition strongly resembles the parity condition [21] which requires that the highest priority that occurs infinitely often must be even. The difference can be explained by the fact that in our system of equations least and greatest fixpoints (μ, ν) need not alternate.

7.1. The algorithm

We propose an algorithm which given an element of the basis $b \in B_L$ and some index $i \in \underline{m}$, checks whether b is below the solution of the i-th fixpoint equation of the system, i.e., $b \sqsubseteq s_i$. According to Theorem 6.2, this corresponds to establish which of the players has a winning strategy in the powerset game starting from the position (b, i). The procedure roughly consists in a depth-first exploration of the tree of plays arising as unfolding of the game graph starting from the initial position (b, i). The algorithm optimises the search by making assumptions on particular subtrees, which are thus pruned. Assumptions can be later confirmed or invalidated, and thus withdrawn. The algorithm is split into three different functions (see Fig. 3).

- Function Explore explores the tree of plays of the game, trying different moves from each node in order to determine the player who has a winning strategy from such a node.
- Function Backtrack allows to backtrack from a node after the algorithm has established who was the winner from it, propagating the information backwards.
- Sometimes the algorithm makes erroneous assumptions when pruning the search in some position and this leads it to incorrectly designate a player as the winner from that position. However, the algorithm is able to detect this fact and correct its decisions. The correction is performed by the function Forget.

7.1.1. Data structures

The algorithm uses the following data structures:

- The *counter* $\boldsymbol{k}$, i.e., an m-tuple of natural numbers, which associates each non-zero priority with the number of times the priority has been encountered in the play since a higher priority was last encountered (the current position is not included). After any move, the counter is updated taking into account the priority of the current position. More precisely, the update of a counter $\boldsymbol{k}$ when moving from a position with priority i, denoted $next(\boldsymbol{k}, i)$, is defined as follows: $next(\boldsymbol{k}, i)_j = 0$ for all $j < i$, $next(\boldsymbol{k}, i)_i = k_i + 1$, and $next(\boldsymbol{k}, i)_j = k_j$ for all $j > i$. Note that, in particular, $next(\boldsymbol{k}, 0) = \boldsymbol{k}$, i.e., moves from a position with priority 0, which are the moves of $\forall$, do not change $\boldsymbol{k}$. We also define two total orders $<_\exists$ and $<_\forall$ on counters, that intuitively measure how good the current advancement of the game is for the two players. We let $\boldsymbol{k} <_\exists \boldsymbol{k}'$ when the largest i where $k_i \neq k'_i$ is the index of a greatest fixpoint equation and $k_i < k'_i$, or it is the index of a least fixpoint and $k_i > k'_i$. The other order $<_\forall$ is the inverse of $<_\exists$, that is $\boldsymbol{k} <_\forall \boldsymbol{k}'$ iff $\boldsymbol{k}' <_\exists \boldsymbol{k}$. For each player P, we write $\boldsymbol{k} \le_P \boldsymbol{k}'$ for $\boldsymbol{k} <_P \boldsymbol{k}'$ or $\boldsymbol{k} = \boldsymbol{k}'$. Notice that the update function *next* is monotone on the counter, that is, given a priority i, for every player P, if $\boldsymbol{k} \le_P \boldsymbol{k}'$, then $next(\boldsymbol{k}, i) \le_P next(\boldsymbol{k}', i)$.
- The *playlist* ρ, i.e., a list of the positions encountered from the initial position to the current node (empty if the current node is the initial position), each with the corresponding counter $\boldsymbol{k}$ and the indication of the alternative moves which have not been explored (exploration is performed depth-first). Thus, ρ is a list of triples $(C, \boldsymbol{k}, \pi)$, where C is a position, $\boldsymbol{k}$ is a counter and $\pi \subseteq Pos$ is the set of the unexplored moves from that position.
- The *assumptions* for players $\exists$ and $\forall$, i.e., a pair of finite sets $\Gamma = (\Gamma_\exists, \Gamma_\forall)$. A position C is assumed to be winning for some player when it is encountered for the second time in the current playlist ρ. This reveals the presence of a loop in the game graph which can be unfolded into an infinite play. Position C is assumed to be winning for the player who would win such an infinite play. In detail, if $\boldsymbol{k}$ is the current counter and $\boldsymbol{k}'$ is the counter of the previous occurrence of C, then the winner P is the player such that $\boldsymbol{k}' <_P \boldsymbol{k}$. In fact, this ensures that the highest priority in the loop is the index of a least fixpoint if $P = \forall$ and of a greatest fixpoint if $P = \exists$. The assumption is stored with the corresponding counter, i.e., Γ_P contains pairs of the kind $(C, \boldsymbol{k})$. Since other possible paths branching from the loop are possibly unexplored, assumptions can still be falsified afterwards.
- The *decisions* for player $\exists$ and $\forall$, i.e., a pair of finite sets $\Delta = (\Delta_\exists, \Delta_\forall)$. Intuitively, a decision for a player P is a position C of the game such that we established that P has a winning strategy from C. The decision is stored with the corresponding counter, i.e., Δ_P contains pairs of the kind $(C, \boldsymbol{k})$. When a new decision is added, we also record its *justification*, i.e., the assumptions and decisions we relied on for deriving the new decision, if any.

```
function EXPLORE(C, k, ρ, Γ, Δ)
    if M(C) = ∅ then
        Δ_{P(C)‾} := Δ_{P(C)‾} ∪ {(C, k)};
        BACKTRACK(P(C)‾, C, ρ, Γ, Δ);
    else if there is (C, k′) ∈ Δ_P such that k′ ≤_P k then
        BACKTRACK(P, C, ρ, Γ, Δ);
    else if there is (C, k′, π) ∈ ρ then
        let P such that k′ <_P k;
        Γ_P := Γ_P ∪ {(C, k′)};
        BACKTRACK(P, C, ρ, Γ, Δ);
    else
        pick C′ ∈ M(C);
        k′ := next(k, i(C));
        π := (M(C) ∖ {C′}) × {k′};
        EXPLORE(C′, k′, ((C, k, π) :: ρ), Γ, Δ);
    end if
end function
```

```
function BACKTRACK(P, C, ρ, Γ, Δ)
    if ρ = [] then
        P;
    else if ρ = ((C′, k′, π) :: t) then
        if P(C′) ≠ P and π ≠ ∅ then
            pick (C″, k″) ∈ π;
            π′ := π ∖ {(C″, k″)};
            EXPLORE(C″, k″, ((C′, k′, π′) :: t), Γ, Δ);
        else
            if P(C′) = P then
                Δ_P := Δ_P ∪ {(C′, k′)} justified by C;
            else
                Δ_P := Δ_P ∪ {(C′, k′)} justified by M(C′);
            end if
            Γ_P := Γ_P ∖ {(C′, k′)};
            if there is (C′, k′) ∈ Γ_{P‾} then
                Δ_{P‾} := FORGET(Δ_{P‾}, Γ_{P‾}, (C′, k′));
                Γ_{P‾} := Γ_{P‾} ∖ {(C′, k′)};
            end if
            BACKTRACK(P, C′, t, Γ, Δ);
        end if
    end if
end function
```

Fig. 3. The general local algorithm.

7.1.2. *The function* EXPLORE

For checking whether $b \sqsubseteq s_i$ for $b \in B_L$ and $i \in \underline{m}$, we call the function EXPLORE$((b,i)$, $\mathbf{0}$, [], $(\emptyset,\emptyset)$, $(\emptyset,\emptyset))$, where $\mathbf{0}$ is the everywhere-zero counter. This returns the (only) player P having a winning strategy from position (b,i), and, by Theorem 6.2, $P = \exists$ if and only if $b \sqsubseteq s_i$. (Here we use the fact that parity games are determined [21], i.e., either $\exists$ or $\forall$ has a winning strategy.)

Given the current position C, the corresponding counter $\boldsymbol{k}$, the playlist ρ describing the path that led to C, and the sets of assumptions Γ and decisions Δ, function EXPLORE$(C$, $\boldsymbol{k}$, ρ, Γ, $\Delta)$ checks if one of the following three conditions holds, each one corresponding to a different **if** branch.

- If $\mathsf{M}(C) = \emptyset$, then the controller $\mathsf{P}(C)$ of position C cannot move and its opponent $\overline{\mathsf{P}(C)}$ wins. Therefore, a new decision for the current position is added for the opponent, and we backtrack. A decision of this kind, with empty justification is called a *truth*.
- If there is already a decision for a player P for the current position C, that is, $(C, \boldsymbol{k}') \in \Delta_P$ and $\boldsymbol{k}' \leq_P \boldsymbol{k}$, then we can reuse that information to assert that P would win from the current position as well. The requirement $\boldsymbol{k}' \leq_P \boldsymbol{k}$ intuitively ensures that we arrived to the current position C with a play that is at least as good for P as the play which lead to the previous decision $(C, \boldsymbol{k}')$.

- If the current position C was already encountered in the play, i.e., $(C, \boldsymbol{k}', \pi) \in \rho$ for some $\boldsymbol{k}'$ and π, then C becomes an assumption for the player P for which the counter got strictly better, that is, $\boldsymbol{k}' <_P \boldsymbol{k}$. Then we backtrack.
- If none of the conditions above holds, the exploration continues from C. A move $C' \in \mathsf{M}(C)$ is chosen to be explored. The playlist is thus extended by adding $(C, \boldsymbol{k}, \pi)$ where π records the remaining moves to be explored. The counter $\boldsymbol{k}$ is updated according to the priority of the now past position C.

7.1.3. *The function* Backtrack

Function Backtrack(P, C, ρ, Γ, Δ) is used to backtrack from a position C, reached via the playlist ρ, after assuming or deciding that player P would win from such position.

- If $\rho = []$ we are back at the root, the position from where the computation started, and the exploration is concluded. The algorithm decides that player P is the winner from such a position.
- Otherwise, the head $(C', \boldsymbol{k}, \pi)$ of the playlist ρ is popped and the status of position C' is investigated.
  - If C' is controlled by the opponent of P ($\mathsf{P}(C') \neq P$) and there are still unexplored moves ($\pi \neq \emptyset$), we must explore such moves before deciding the winner from C'. Then, a new move is extracted from π and explored.
  - If instead the controller of C' is P ($\mathsf{P}(C') = P$) then P wins also from C'. Hence C' is inserted in Δ_P, justified by the move C from where we backtracked. Similarly, if the controller of C' is the opponent of P ($\mathsf{P}(C') \neq P$), we already explored all possible moves from C' ($\pi = \emptyset$) and all turn out to be winning for P, again we decide that P wins from C', which is inserted in Δ_P, justified by all possible moves from C'. Since we decided that P would win from C' we can now continue to backtrack. However, before backtracking we must discard all assumptions for the opponent of P in conflict with the newly taken decision, and this must be propagated to the decisions depending on such assumptions. This is done by the invocation Forget($\Delta_{\overline{P}}$, $\Gamma_{\overline{P}}$, $(C', \boldsymbol{k}')$).

In general the choice of moves to explore, performed by the action "pick" in the pseudocode, is random. However, by the results in § 6.1, we can restrict the moves of player $\exists$ to a selection. Furthermore, it is usually convenient to give priority to moves which are immediately reducible to valid decisions or assumptions for the player who is moving. A practical way to do this is to check if there is a decision for a position C', with a valid counter with respect to the current one, such that either the current position $C = (b, i)$, $C' = (b', i)$ and $b \sqsubseteq b'$, or $C = \boldsymbol{X}$, $C' = \boldsymbol{X}'$ and $\boldsymbol{X}' \subseteq \boldsymbol{X}$. Then, the move to pick is the one justifying such decision, which by those features is guaranteed to be a move also from the current position C.

7.1.4. *The function* Forget

The function Forget is not given explicitly. The precise definition of the property that function Forget must satisfy in order to ensure the correctness of the algorithm is quite technical (it can be found in the appendix provided as extra material). Intuitively, when an assumption in Γ_P fails and is withdrawn, then we must remove from Δ_P at least all the decisions depending on such an assumption. It is possible that decisions taken on the basis of the deleted assumption remain valid because they could be justified by other decisions or assumptions, possibly introduced later. Different sound realisations of Forget are then possible (see [57]) and, experimentally, it can be seen that those removing only the set of decisions depending on the falsified assumption can be practically inefficient. A simpler sound implementation, which, at least in the setting of the μ-calculus, as reported in [57] resulted to be the most efficient, is based on a temporal criterion: when an assumption fails, the function deletes all decisions which have been taken after the position corresponding to the failed assumption was initially encountered. It can be implemented by associating timestamps with decisions and assumptions, and avoiding the complex management of justifications. Note that the timestamp assigned to an assumption is not based on when the assumption is added, but rather on when the previous occurrence of the same position was initially explored. Then, Forget will remove all decisions with timestamps more recent than that of the failed assumption. This is sound because justification dependence clearly implies time dependence.

Example 7.1 *(model-checking μ-calculus).* Consider the transition system $T = (\mathbb{S}, \to)$ in Fig. 1a and the μ-calculus formula $\varphi = \mu x_2.((\nu x_1.(p \wedge \Box x_1)) \vee \Diamond x_2)$ discussed in Example 3.5. As already discussed, the formula φ interpreted over T leads to the system E in Fig. 1d over the lattice $\mathbf{2}^{\mathbb{S}}$.

Suppose that we want to verify whether the state $a \in \mathbb{S}$ satisfies the formula φ. This requires to determine the winner of the powerset game from position $(a, 2)$, which can be done by invoking Explore($(a, 2)$, $\mathbf{0}$, [], $(\emptyset, \emptyset)$, $(\emptyset, \emptyset)$). A computation performed by the algorithm is schematised in Fig. 4. Observe that we consider only moves from the least selection which exists by Proposition 6.10, since the lattice $\mathbf{2}^{\mathbb{S}}$ is finite. Since the choice of moves is non-deterministic, other search sequences are possible. In the diagram, positions of player $\exists$ are represented as diamonds, while those of $\forall$ are represented as boxes, the counters associated with the positions are on their righthand side.

Recall that the second equation is $x_2 =_\mu x_1 \cup \blacklozenge_T x_2$. Then, from the initial position $(a, 2)$, with counter $(0, 0)$, there are four available moves in the least selection, i.e., $(\{a\}, \emptyset)$, $(\emptyset, \{a\})$, $(\emptyset, \{b\})$ and $(\emptyset, \{c\})$, represented by the four outgoing edges from position $(a, 2)$ in the diagram, all four will have counter $(0, 1) = \mathit{next}((0, 0), 2)$. Indeed, it is easy to see that $a \in \{a\} \cup \blacklozenge_T \emptyset = \emptyset \cup \blacklozenge_T \{a\} = \emptyset \cup \blacklozenge_T \{b\} = \{a\} \subseteq \emptyset \cup \blacklozenge_T \{c\} = \{a, c\}$. Suppose that the algorithm chooses to explore the move $(\emptyset, \{b\})$, as highlighted by the bold arrow. Even though not shown in the diagram, the other moves are stored in the set of unexplored moves π associated with the position $(a, 2)$ in the playlist ρ. The search proceeds in this way along the moves

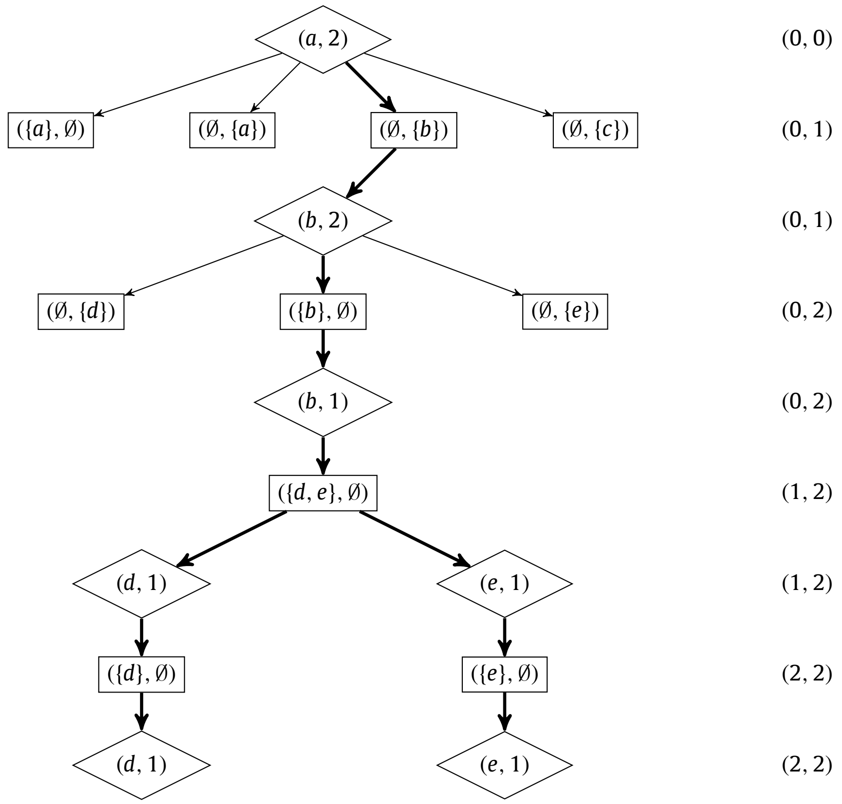


Fig. 4. An execution of the local algorithm.

$$(a, 2) \overset{\exists}{\rightsquigarrow} (\emptyset, \{b\}) \overset{\forall}{\rightsquigarrow} (b, 2) \overset{\exists}{\rightsquigarrow} (\{b\}, \emptyset) \overset{\forall}{\rightsquigarrow} (b, 1) \overset{\exists}{\rightsquigarrow} (\{d, e\}, \emptyset) \overset{\forall}{\rightsquigarrow} (d, 1) \overset{\exists}{\rightsquigarrow} (\{d\}, \emptyset) \overset{\forall}{\rightsquigarrow}$$

until position $(d, 1)$ occurs again, with counter $(2, 2)$. Since the counter associated with the first occurrence of $(d, 1)$ was $(1, 2)$ and $(1, 2) <_\exists (2, 2)$, then the pair position and counter $((d, 1), (1, 2))$ is added as an assumption for player $\exists$ and the algorithm starts backtracking. While backtracking it generates a decision for $\exists$, which is $((\{d\}, \emptyset), (2, 2))$ justified by the only possible move $(d, 1)$ of player $\forall$. When it comes back to the first occurrence of $(d, 1)$, since it is a position controlled by $\exists$, the procedure transforms the assumption $((d, 1), (1, 2))$ into a decision for $\exists$ justified by the move $(\{d\}, \emptyset)$. Then, it backtracks to position $(\{d, e\}, \emptyset)$, which is controlled by player $\forall$ and there is still an unexplored move $(e, 1)$. Therefore, the algorithm starts exploring again from $(e, 1)$, and does so similarly to the previous branch of $(d, 1)$. After making decisions for those positions as well, the algorithm resumes backtracking from $(\{d, e\}, \emptyset)$, since all possible moves have been explored, making decisions for player $\exists$ along the way back. This goes on up until the initial position is reached again. The last invocation $\textsc{Backtrack}(\exists, (a,2), [], \Gamma, \Delta)$ terminates since $\rho = []$, and returns player $\exists$. Indeed, $\exists$ wins starting from position $(a, 2)$ since the state a satisfies the formula φ.

7.1.5. Correctness

We show that, when the lattice is finite, the algorithm terminates. Moreover, when it terminates (which could happen also on infinite lattices), it provides a correct answer.

Termination on finite lattices can be proved by observing that the set of positions (which are either elements of the basis or tuples of sets of elements of the basis) is finite. The length of playlists is bounded by the number of positions, since, whenever a position repeats in a playlist, it necessarily becomes an assumption and backtracking starts. Finally, one can observe that it is not possible to cycle indefinitely between two positions and thus termination immediately follows.

Lemma 7.2 *(termination). Given a powerset game on a finite lattice, any call* $\textsc{Explore}(C_0, \mathbf{0}, [], (\emptyset, \emptyset), (\emptyset, \emptyset))$ *terminates with an invocation of* $\textsc{Backtrack}(P, C_0, [], (\emptyset, \emptyset), \Delta)$ *for some player P and a set Δ.*

The proof of correctness is long and technical. The underlying idea is to prove that, at any invocation of $\textsc{Explore}(\cdot, \cdot, \rho, \Gamma, \Delta)$ and $\textsc{Backtrack}(\cdot, \cdot, \rho, \Gamma, \Delta)$, the justifications for the decisions Δ_P, can be interpreted as a winning strategy for player P from the positions $C \in \Delta_P$, in a modified game where P immediately wins on the assumptions Γ_P. Since at termination, the set of assumptions is empty, the modified game coincides with the original one and thus we conclude.

Theorem 7.3 *(correctness). Given a powerset game, if a call* Explore$(C, \mathbf{0}, [], (\emptyset,\emptyset), (\emptyset,\emptyset))$ *returns a player P, then P wins the game from C.*

Notice that it is unnecessary to prove the converse implication, that is, if P wins the game from C, then the call Explore$(C, \mathbf{0}, [], (\emptyset,\emptyset), (\emptyset,\emptyset))$ returns P. Indeed, since the game can never result in a draw, this is equivalent to show that if the call Explore$(C, \mathbf{0}, [], (\emptyset,\emptyset), (\emptyset,\emptyset))$ returns $\overline{P}$, then $\overline{P}$ wins the game from C. And this already holds by Theorem 7.3.

7.2. Using up-to techniques in the algorithm

In the literature about bisimilarity checking, up-to techniques have been fruitfully integrated with local checking algorithms for speeding up the computation (see, e.g., [33]). Here we show that a similar idea can be exploited in our local algorithm for general systems of fixpoint equations.

By relying on Theorem 5.11 we can derive an algorithm that exploits the up-to function $\boldsymbol{u}$. It is obtained by instantiating the general algorithm discussed before to the system $d(E, \boldsymbol{u})$ (see § 5.2) and suitably restricting the moves considered in the exploration. Roughly, the idea is to allow the use of the up-to function only when it leads immediately to an assumption or a decision. This is in some sense similar to what is done for bisimilarity checking in [33], where the up-to function is used only to enlarge the set of states which are considered bisimilar. More precisely, when the exploration is in a position (b, i) corresponding to one of the added equations $y_i =_\mu u_i(y_i) \sqcup x_i$, according to the definition of the game, a possible move would be any $2m$-tuple of sets $(\boldsymbol{Y}, \boldsymbol{X})$ such that $b \sqsubseteq u_i(\bigsqcup Y_i) \sqcup \bigsqcup X_i$. First of all, since only the i-th and $(m+i)$-th components Y_i and X_i play a role and we can restrict to minimal moves (see § 6.2), we can assume $X_j = Y_j = \emptyset$ for $j \neq i$. Moreover, for X_i and Y_i, we only allow two types of moves:

1. $X_i = \{b\}$ and $Y_i = \emptyset$, which means that we keep the focus on element b and just jump to the "original" equation $x_i =_{\eta_i} f(y_i)$, or
2. $X_i = \emptyset$ and Y_i is a set of positions which would immediately become assumptions when explored or for which previous decisions can be used.

At the level of the pseudocode, this only means that the action "pick" needs to be refined. Instead of simply choosing randomly a move in $\mathsf{M}(C)$, in some cases it has to perform a constrained choice. This is made precise below.

Definition 7.4 *(up-to algorithm).* Let E be a system of m fixpoint equations over the complete lattice L and let $\boldsymbol{u}$ be a compatible tuple of up-to functions for E. The up-to algorithm for E based on $\boldsymbol{u}$ is just the algorithm in Fig. 3 applied to the system $d(E, \boldsymbol{u})$, where, in function Explore$(C, \boldsymbol{k}, \rho, \Gamma, \Delta)$, when $C = (b, i)$ with $i \in \underline{m}$, the action "pick" can select only moves $C' = (\boldsymbol{Y}, \boldsymbol{X})$ such that $Y_j = X_j = \emptyset$ for $j \neq i$ and complying with either of the following conditions

1. $Y_i = \emptyset$ and $X_i = \{b\}$ or
2. $X_i = \emptyset$ and for all $b' \in Y_i$ it holds
   (a) $((b', i), \boldsymbol{k}') \in \Delta_\exists$ with $\boldsymbol{k}' \leq_\exists \mathit{next}(\boldsymbol{k}, i)$ or
   (b) $((b', i), \boldsymbol{k}', \pi) \in \rho$ with $\boldsymbol{k}' <_\exists \mathit{next}(\boldsymbol{k}, i)$.

Condition (1) has been already clarified above. Condition (2) is a formal translation of the fact that Y_i can contain only positions for which there are decisions that can be used (case (2a)) or that will immediately become assumptions (Case 2b)).

Clearly the modification does not affect termination on finite lattices (in fact, we just restrict the possible moves of a procedure which is known to be terminating). We next show that the up-to algorithm is also correct.

Theorem 7.5 *(correctness with up-to). Let E be a system of m equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ over a complete lattice L. Let $\boldsymbol{u}$ be a compatible m-tuple of up-to functions for E. Then the up-to algorithm associated with the system $d(E, \boldsymbol{u})$ as given in Definition 7.4 is correct, i.e., if a call* Explore$(C, \mathbf{0}, [], (\emptyset,\emptyset), (\emptyset,\emptyset))$ *returns a player P, then P wins the game from C.*

The proof, reported in the appendix, is based on the observation that any winning strategy for player $\exists$ in the game associated with the original system E can be replicated in the game associated with the modified system $d(E, \boldsymbol{u})$, even when the moves are restricted as in Definition 7.4. This is done by choosing always moves corresponding to case (1) in Definition 7.4. Then strategies in the constrained game for $d(E, \boldsymbol{u})$ are also valid in the unconstrained game. We conclude since, by Theorem 5.11, we know that winning positions for player $\exists$ are the same in the game for E and in the game for $d(E, \boldsymbol{u})$.

Further optimizations of the up-to algorithm are possible. E.g., we can exploit the fact that a variable y_i has the same solution of the corresponding x_i in the system $d(E, \boldsymbol{u})$. Intuitively, decisions and assumptions for positions associated with a variable y_i could be used as decisions and assumptions for the corresponding positions of variable x_i, and the other way around.

Example 7.6 *(model-checking μ-calculus up-to bisimilarity).* In Example 7.1 we showed how the algorithm would solve a model-checking problem by exploring the corresponding fixpoint game. Suppose that here we want to use up-to bisimilarity as an up-to technique to answer the same question, that is, whether the state $a \in \mathbb{S}$ satisfies the formula $\varphi = \mu x_2.((\nu x_1.(p \wedge \Box x_1)) \vee \Diamond x_2)$. In Example 5.9 we presented the up-to function $u_\sim : \mathbf{2}^{\mathbb{S}} \to \mathbf{2}^{\mathbb{S}}$ corresponding to up-to bisimilarity defined as $u_\sim(X) = \{s \in \mathbb{S} \mid s \sim_T s' \ \wedge \ s' \in X\}$. In order to apply the procedure described above, first we need to build the system $d(E, (u_\sim, u_\sim))$, which is

$$\begin{aligned} y_1 &=_\mu u_\sim(y_1) \cup x_1 & \qquad x_1 &=_\nu \{b, d, e\} \cap \blacksquare_T y_1 \\ y_2 &=_\mu u_\sim(y_2) \cup x_2 & \qquad x_2 &=_\mu y_1 \cup \blacklozenge_T y_2 \end{aligned}$$

Then, to check whether the state a satisfies the formula φ we invoke the function Explore($(a, 4)$, $\mathbf{0}$, [], $(\emptyset, \emptyset)$, $(\emptyset, \emptyset)$), where the index 4 corresponds to variable x_2 in the system $d(E, (u_\sim, u_\sim))$. Then, the algorithm behaves in similar fashion to what was described in Example 7.1. However, this time the exploration of position $(d, 1)$ with counter $(0, 0, 1, 2)$ is pruned by using the up-to function. Recalling that position $(b, 1)$ occurred in the past, hence it is included in the playlist, with counter $(0, 0, 0, 2)$, we have that condition (2) of Definition 7.4 holds for the move $(\{b\}, \emptyset, \emptyset, \emptyset)$ since $d \sim b$, hence $d \in u_\sim(\{b\}) \cup \emptyset$, and $(0, 0, 0, 2) <_\exists next((0, 0, 1, 2), 1) = (1, 0, 1, 2)$. This leads to making an assumption for $(b, 1)$ and then backtracking. The same happens when exploring the other branch, that is position $(e, 1)$, since also $e \sim b$. Similarly to what happened in the previous example, the last invocation Backtrack($\exists$, $(a, 4)$, [], Γ, Δ) returns player $\exists$. Indeed, $\exists$ wins starting from position $(a, 4)$ since the state a satisfies the formula φ.

8. Conclusions

We proposed a general framework for dealing with abstraction in the solution of systems of fixpoint equations over complete lattices, mixing least and greatest fixpoints. We showed how up-to techniques, a classical tool used in coinductive settings, can be seen as a special form of abstraction and generalise to systems of fixpoint equations. Relying on the approximation theory, we provided a characterisation of the solution of systems of fixpoint equations over complete lattices in terms of a suitable parity game, generalising [3] that was restricted to continuous lattices. The game-theoretical characterisation of the solution of systems of fixpoint equations is exploited to derive a local algorithm for determining the winner of the game at a specific position, thus solving the corresponding verification problem. We also showed how to integrate up-to techniques in the algorithm.

We showed how the local algorithm for μ-calculus model checking in [57] can be adapted in our setting for solving arbitrary fixpoint equation systems. This works properly essentially for two reasons. Firstly, the fact that the set of moves is partially ordered allows one to skip the exploration of some positions of the game (as formalised by the notion of minimal selections) and this properly integrates with the local nature of the algorithm. In fact, we developed a prototype implementation of the algorithm [22] which relies on a symbolic representation of the set of moves to be explored, based on the so-called *symbolic moves* introduced in [3]. The information gained during the execution of the local algorithm is further utilised to progressively prune the symbolic moves, thus influencing how the remaining part of the game is explored. Secondly, the moves introduced in the game by the use of up-to techniques have a special status since they should be used only when they are conclusive, that is, they immediately determine a winner for the current position. The local algorithm in § 7.1 can be naturally adapted to embody this constraint.

As mentioned in the introduction, a number of other local approaches have been proposed for the solution of parity games and it would be interesting to investigate whether also these approaches can be adapted to our setting and determine those which are most effective. In particular, an in-depth comparison with the local algorithms for general parity games proposed in [25] and recently in [40] appears to be interesting. This is out of the scope of the current paper, but from an initial analysis it can be seen that those procedures and ours do similar things in different ways. More specifically, our method uses the currently obtained partial knowledge of the game graph to conjecture the winner in some of the explored positions. These conjectures can be later falsified by further exploring the game graph. Indeed, they are upheld only by the cyclic plays and the corresponding dominating priorities identified up to the current partial exploration of the game. Such information implicitly induces strategies for both players, which however are not guaranteed to still be winning for the same player and positions after other possible moves of the opponent are explored. The approach in [25] also exploits the graph structure and priorities to establish winning positions and strategies for both players in a subset of the game graph. The explored subset allows to determine positions which are winning with certainty only under certain conditions, i.e., when the opposing player has no choices other than those already inspected. The procedure then iteratively expands such knowledge by further exploring the game graph, up to the point where the initially requested position is assigned to the winning set of one of the players. In particular, in the symmetric version of the algorithm, each expansion phase starts from one of the unexplored nodes controlled by the player who, under the current strategies, would lose from the position initially requested, and stops when it loops back to previously encountered nodes. This matches the process of exploration in our algorithm, which, as mentioned above, could lead to falsify the current conjectures. However, in our case such conjectures are explicitly recorded and re-used, especially when integrating up-to techniques, while in [25] only the guaranteed winning positions seem to be recorded. Also the approach in [40] focuses on identifying smaller sets of positions where the winner can be established with certainty using the partial knowledge. While exploring the game graph it builds subsets, therein called *dominions*, of nodes where one player has a winning strategy that forces every play starting from any of those positions to remain inside that subset. These sets are further restricted to safe nodes whose winner can be definitively decided despite the partial knowledge. At the same time, dominions are also expanded using so-called *safe attractors*, i.e., sets of nodes from where a player can safely force plays to reach a specified desired position, for instance in one of that player's dominions, taking into account the possible current lack of knowledge. In a sense, the difference between our algorithm and those in [25,40] lies in the kind of approximation, over- or under-, of the winning positions that the algorithms perform based on their partial local knowledge of the game. It is worth noting that both mentioned local algorithms, applied to the setting of systems of fixpoint equations, appear to be compatible with the integration of the symbolic representation of moves and of up-to techniques, as described before. Even so, the effectiveness of the latter may vary w.r.t. our case, because of the differences explained above. Nevertheless, this requires a more thorough investigation, and so it remains open as a future research direction.

Inspired by [9,7,32,33], we also adapted the algorithm to allow the use of up-to techniques. Here, as in the mentioned works, up-to techniques are used to enhance the ability of proving properties, as opposed to disproving them. In our setting this essentially corresponds to an increment of the possible existential player moves, as described in § 7.2. On the other hand, it could be interesting to find ways to use the up-to functions also in the opposite manner, in our case as a tool for the universal player for generating assumptions and decisions. However, a straightforward adaptation of the device utilised for the existential player appears to be unfeasible.

The notion of progress measures that has been studied in [3] can be adapted to the game for arbitrary complete (rather than just continuous) lattices, introduced in this paper. A natural question is whether the local 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 [53,20,39] or to abstraction for the full μ-calculus as studied in [29] 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 [8].

Several authors have shown that systems of fixpoint equations can be suitably transformed while keeping the solution unchanged. This is discussed, e.g., for (parametrised) Boolean equation systems in [43,28]. The transformation can be used to simplify the system, possibly leading to a form in which the solution is immediate. This is surely an interesting direction to explore also for systems of equations over general lattices, as it could allow to bring the system to a shape in which the solution via the parity game becomes more efficient.

CRediT authorship contribution statement

Paolo Baldan: Writing – original draft. **Barbara König:** Writing – original draft. **Tommaso Padoan:** Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Proofs for Section 4 (Approximation for systems of fixpoint equations)

Lemma A.1 *(concretisation for single fixpoints). Let $\gamma : A \to C$ be a monotone function.*

1. *If*

$$f_C \circ \gamma \sqsubseteq \gamma \circ f_A \tag{A.1}$$

then $\mu f_C \sqsubseteq \gamma(\mu f_A)$; if, in addition, γ is co-continuous and co-strict $\nu f_C \sqsubseteq \gamma(\nu f_A)$.

2. *If*

$$\gamma \circ f_A \sqsubseteq f_C \circ \gamma \tag{A.2}$$

then $\gamma(\nu f_A) \sqsubseteq \nu f_C$; if, in addition, γ is continuous and strict then $\gamma(\mu f_A) \sqsubseteq \mu f_C$.

Proof. We focus on the soundness results since the completeness results follow by duality.

For least fixpoint, we prove that for all ordinals β we have $f_C^\beta(\bot_C) \leq \gamma(f_A^\beta(\bot_A))$, whence the thesis, since $\mu f_C = f_C^\beta(\bot_C)$ and $\mu f_A = f_A^\beta(\bot_A)$ for some ordinal β (just take the largest of the ordinals needed to reach the two fixpoints).

We proceed by transfinite induction:

- ($\beta = 0$) We have $f_C^0(\bot_C) = \bot_C \sqsubseteq \gamma(f_A^0(\bot_A))$, as desired.
- ($\beta \to \beta+1$) Observe that

$$\begin{aligned} f_C^{\beta+1}(\bot_C) &= f_C(f_C^{\beta}(\bot_C)) \\ &\sqsubseteq f_C(\gamma(f_C^{\beta}(\bot_C))) && \text{[by ind. hyp. and monotonicity of } f_C] \\ &\le \gamma(f_A(f_A^{\beta}(\bot_A))) && \text{[by (A.1)]} \\ &= \gamma(f_A^{\beta+1}(\bot_A)) \end{aligned}$$

- (β limit ordinal) In this case

$$\begin{aligned} f_C^{\beta}(\bot_C) &= \bigsqcup_{\beta'<\beta} f_C^{\beta'}(\bot_C) \\ &\sqsubseteq \bigsqcup_{\beta'<\beta} \gamma(f_A^{\beta'}(\bot_A)) && \text{[by ind. hyp.]} \\ &\sqsubseteq \gamma(\bigsqcup_{\beta'<\beta} f_A^{\beta'}(\bot_A)) && \text{[by properties of joins]} \\ &= \gamma(f_A^{\beta}(\bot_A)) \end{aligned}$$

For greatest fixpoints, we prove that for all ordinals β we have $f_C^{\beta}(\top_C) \le \gamma(f_A^{\beta}(\top_A))$, again by transfinite induction.

- ($\beta = 0$) We have $f_C^0(\top_C) = \top_C = \gamma(\top_A) = \gamma(f_A^0(\top_A))$, since γ is assumed to be co-strict, hence we have the desired inequality.
- ($\beta \to \beta+1$) Observe that

$$\begin{aligned} f_C^{\beta+1}(\top_C) &= f_C(f_C^{\beta}(\top_C)) \\ &\sqsubseteq f_C(\gamma(f_A^{\beta}(\top_A))) && \text{[by ind. hyp. and monotonicity of } f_C] \\ &\sqsubseteq \gamma(f_A(f_A^{\beta}(\top_A))) && \text{[by (A.1)]} \\ &= \gamma(f_A^{\beta+1}(\top_A)) \end{aligned}$$

- (β limit ordinal) In this case

$$\begin{aligned} f_C^{\beta}(\top_C) &= \bigsqcap_{\beta'<\beta} f_C^{\beta'}(\top_C)) \\ &\sqsubseteq \bigsqcap_{\beta'<\beta} \gamma(f_A^{\beta'}(\top_A)) && \text{[by ind. hyp.]} \\ &= \gamma(\bigsqcap_{\beta'<\beta} f_A^{\beta'}(\top_A)) && \text{[since } \gamma \text{ is co-continuous]} \\ &= \gamma(f_A^{\beta}(\top_A)) \quad \square \end{aligned}$$

We can get analogous results for abstractions, by duality.

Lemma A.2 *(abstraction for single fixpoints). Let $\alpha : C \to A$ be an abstraction function.*

1. *If*

$$\alpha \circ f_C \le f_A \circ \alpha \tag{A.3}$$

then $\alpha(\nu f_C) \le \nu f_A$; if, in addition, α is continuous and strict $\alpha(\mu f_C) \le \mu f_A$.

2. *If*

$$f_A \circ \alpha \le \alpha \circ f_C \tag{A.4}$$

then $\mu f_A \le \alpha(\mu f_C)$; if, in addition, α is co-continuous and co-strict then $\nu f_A \le \alpha(\nu f_C)$.

Lemma A.3 *(Galois insertions). Let $f_C : C \to C$ and $f_A : A \to A$ be monotone functions and let $\langle \alpha, \gamma \rangle : C \to A$ be a Galois insertion.*

1. *Assume soundness for α i.e., (A.3) (equivalent to soundness for γ, i.e., (A.1)), and completeness for both α and β, i.e., (A.4), (A.2). Then*

$$\alpha(\eta f_C) = \eta f_A \text{ for } \eta \in \{\mu, \nu\} \qquad \nu f_C = \gamma(\nu f_A) \qquad \mu f_C \sqsubseteq \gamma(\mu f_A)$$

2. *Assume*

$$f_C = \gamma \circ f_A \circ \alpha \tag{A.5}$$

then $\alpha(\eta f_C) = \eta f_A$ and $\eta f_C = \gamma(\eta f_A)$ for $\eta \in \{\mu, \nu\}$.

Proof. 1. Just using Lemma A.1 and Lemma A.2, we obtain

$$\text{(a) } \alpha(\mu f_C) = \mu f_A \quad \text{(b) } \nu f_C = \gamma(\nu f_A) \quad \text{(c) } \alpha(\mu f_C) \le \mu f_A \quad \text{(d) } \mu f_C \sqsubseteq \gamma(\mu f_A)$$

From (b), applying α, we obtain $\alpha(\nu f_C) = \alpha(\gamma(\nu f_A) = \nu f_A$, and we are done.

2. In this case, from the assumption $f_C = \gamma \circ f_A \circ \alpha$ one can easily deduce the soundness and completeness conditions for α and γ, i.e., (A.3), (A.4), (A.1), (A.2). Therefore, by the previous point we get all desired inequalities but $\gamma(\mu f_A) \sqsubseteq \mu f_C$. For this observe that

$$\begin{aligned}
\gamma(\mu f_A) &= \gamma(\alpha(\mu f_C)) && [\text{since } \mu f_A = \alpha(\mu f_C)]\\
&= \gamma(\alpha(f_C(\mu f_C))) && [\text{since } \mu f_C \text{ is a fixpoint of } f_C]\\
&= \gamma(\alpha(\gamma(f_A(\alpha(\mu f_C))))) && [\text{since } f_C = \gamma \circ f_A \circ \alpha]\\
&= \gamma(f_A(\alpha(\mu f_C))) && [\text{since } \alpha \circ \gamma = id_A]\\
&= f_C(\mu f_C) && [\text{since } f_C = \gamma \circ f_A \circ \alpha]\\
&= \mu f_C && [\text{since } \mu f_C \text{ is a fixpoint of } f_C] \quad \square
\end{aligned}$$

Theorem 4.1 *(sound concretisation for systems). Let $(C, \sqsubseteq)$ and $(A, \le)$ be complete lattices, let E_C of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^C(\boldsymbol{x})$ and E_A of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^A(\boldsymbol{x})$ 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 $\boldsymbol{s}^C \sqsubseteq \boldsymbol{\gamma}^\times(\boldsymbol{s}^A)$.*

Proof. We proceed by induction on m. The case $m = 0$ is trivial.

For the inductive case, consider systems with $m + 1$ equations. Recall that, in order to solve the system, the last variable x_{m+1} is considered as a fixed parameter x and the system of m equations that arises from dropping the last equation is recursively solved. This produces an m-tuple $\boldsymbol{t}^{\boldsymbol{z}}_{1,m}(x) = sol(E_z[x_{m+1} := x])$ parametric on x, for $z \in \{A, C\}$. For all $a \in A$, by inductive hypothesis applied to the systems $E_A[x_{m+1} := a]$ and $E_C[x_{m+1} := \gamma_{m+1}(a)]$ we obtain

$$\boldsymbol{t}^{\boldsymbol{C}}_{1,m}(\gamma_{m+1}(a)) \sqsubseteq \boldsymbol{\gamma}_{1,m}{}^\times(\boldsymbol{t}^{\boldsymbol{A}}_{1,m}(a)) \tag{A.6}$$

Inserting the parametric solution into the last equation, we get an equation in a single variable

$$a =_{\eta_m} f^A_{m+1}(\boldsymbol{t}^{\boldsymbol{A}}_{1,m}(a), a).$$

This equation can be solved by taking the corresponding fixpoint, i.e., if we define $f_A(a) = f^A_{m+1}(\boldsymbol{t}^{\boldsymbol{A}}_{1,m}(a), a)$, then $s^A_{m+1} = \eta_{m+1} f_A$. In the same way, $s^C_{m+1} = \eta_{m+1} f_C$ where $f_C(c) = f^C_{m+1}(\boldsymbol{t}^{\boldsymbol{C}}_{1,m}(c), c)$.

Observe that $f_C \circ \gamma_{m+1} \sqsubseteq \gamma_{m+1} \circ f_A$. In fact

$$\begin{aligned}
&f_C(\gamma_{m+1}(a)) = \\
&= f^C_{m+1}(\boldsymbol{t}^{\boldsymbol{C}}_{1,m}(\gamma_{m+1}(a)), \gamma_{m+1}(a))) && [\text{definition of } f_C]\\
&\sqsubseteq f^C_{m+1}(\boldsymbol{\gamma}_{1,m}{}^\times(\boldsymbol{t}^{\boldsymbol{A}}_{1,m}(a)), \gamma_{m+1}(a))) && [\text{by (A.6)}]\\
&\sqsubseteq f^C_{m+1}(\boldsymbol{\gamma}^\times(\boldsymbol{t}^{\boldsymbol{A}}_{1,m}(a), a)) && [\text{application of } \boldsymbol{\gamma}]\\
&\sqsubseteq \gamma_{m+1}(f^A_{m+1}(\boldsymbol{t}^{\boldsymbol{A}}_{1,m}(a), a)) && [\text{hypothesis } \boldsymbol{f}^C \circ \boldsymbol{\gamma}^\times \sqsubseteq \boldsymbol{\gamma}^\times \circ \boldsymbol{f}^A]\\
&= \gamma_{m+1}(f_A(a)) && [\text{definition of } f_A]
\end{aligned}$$

Therefore, recalling that when $\eta_{m+1} = \mu$ we are assuming co-continuity and co-strictness for γ_{m+1}, we can apply Lemma A.1(1) and deduce that

$$s^C_{m+1} = \eta_{m+1} f_C \sqsubseteq \gamma_{m+1}(\eta_{m+1} f_A) = \gamma_{m+1}(s^A_{m+1}) \tag{A.7}$$

Finally, recall that the first m components of the solutions are $\boldsymbol{s}^{\boldsymbol{z}}_{1,m} = \boldsymbol{t}^{\boldsymbol{z}}_{1,m}(s^z_{m+1})$ for $z \in \{C, A\}$. Therefore, exploiting (A.6), we have

$$\begin{aligned}
&\boldsymbol{s}^{\boldsymbol{C}}_{1,m} = \\
&= \boldsymbol{t}^{\boldsymbol{C}}_{1,m}(s^C_{m+1}) \\
&\sqsubseteq \boldsymbol{t}^{\boldsymbol{C}}_{1,m}(\gamma_{m+1}(s^A_{m+1})) && \text{[by (A.7)]} \\
&\sqsubseteq \boldsymbol{\gamma}_{1,m}{}^{\times}(\boldsymbol{t}^{\boldsymbol{A}}_{1,m}(s^A_{m+1})) && \text{[by (A.6)]} \\
&= \boldsymbol{\gamma}_{1,m}{}^{\times}(\boldsymbol{s}^{\boldsymbol{A}}_{1,m})
\end{aligned}$$

This concludes the inductive step. □

Everything can be dually formulated in terms of abstraction functions.

Theorem A.4 *(sound abstraction for systems). Let $(C, \sqsubseteq)$ and $(A, \leq)$ be complete lattices and let E_C of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^C(\boldsymbol{x})$ and E_A of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^A(\boldsymbol{x})$ 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 $\boldsymbol{\alpha}$ be an m-tuple of monotone functions, with $\alpha_i : C \to A$ for $i \in \underline{m}$. If $\boldsymbol{\alpha}$ satisfies*

$$\boldsymbol{\alpha}^{\times} \circ \boldsymbol{f}^C \leq \boldsymbol{f}^A \circ \boldsymbol{\alpha}^{\times}$$

with α_i continuous and strict for each $i \in \underline{m}$ such that $\eta_i = \mu$, then $\boldsymbol{\alpha}^{\times}(\boldsymbol{s}^C) \leq \boldsymbol{s}^A$.

Proof. This follows from Lemma 4.1 by duality. □

Theorem 4.2 *(abstraction via Galois connections). Let $(C, \sqsubseteq)$ and $(A, \leq)$ be complete lattices, let E_C of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^C(\boldsymbol{x})$ and E_A of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^A(\boldsymbol{x})$ 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 $\boldsymbol{\gamma}^{\times}(\boldsymbol{s}^A) \sqsubseteq \boldsymbol{s}^C$.*

Proof. Due to Theorems 4.1 and A.4 (and the fact that we can apply the theorems to lattices with reversed order), the only thing to prove is that the conditions $\boldsymbol{\alpha}^{\times} \circ \boldsymbol{f}^C \leq \boldsymbol{f}^A \circ \boldsymbol{\alpha}^{\times}$ and $\boldsymbol{f}^C \circ \boldsymbol{\gamma}^{\times} \sqsubseteq \boldsymbol{\gamma}^{\times} \circ \boldsymbol{f}^A$ are equivalent. If we assume $\boldsymbol{\alpha}^{\times} \circ \boldsymbol{f}^C \leq \boldsymbol{f}^A \circ \boldsymbol{\alpha}^{\times}$, by definition of Galois connection, we get $\boldsymbol{f}^C \sqsubseteq \boldsymbol{\gamma}^{\times} \circ \boldsymbol{f}^A \circ \boldsymbol{\alpha}^{\times}$. Now, post-composing with $\boldsymbol{\gamma}^{\times}$ and exploiting the fact that $\boldsymbol{\alpha}^{\times} \circ \boldsymbol{\gamma}^{\times} \sqsubseteq \boldsymbol{id}^{\times}$ we obtain

$$\boldsymbol{f}^C \circ \boldsymbol{\gamma}^{\times} \sqsubseteq \boldsymbol{\gamma}^{\times} \circ \boldsymbol{f}^A \circ \boldsymbol{\alpha}^{\times} \circ \boldsymbol{\gamma}^{\times} \sqsubseteq \boldsymbol{\gamma} \circ \boldsymbol{f}^A$$

as desired.

The converse implication is analogous. □

For Galois insertions, we make explicit a very special case where we get rid of all the (co-)continuity and (co-)strictness requirements, and get soundness and completeness both for the abstraction and the concretisation.

Lemma A.5 *(Galois insertions for systems). Let $(C, \sqsubseteq)$ and $(A, \leq)$ be complete lattices, let E_C of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^C(\boldsymbol{x})$ and E_A of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}^A(\boldsymbol{x})$ 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 abstraction and concretisation functions, with $\langle \alpha_i, \gamma_i \rangle : C \to A$ a Galois insertion for each $i \in \underline{m}$. If*

$$\boldsymbol{f}_C = \boldsymbol{\gamma}^{\times} \circ \boldsymbol{f}^A \circ \boldsymbol{\alpha} \tag{A.8}$$

then $\boldsymbol{\alpha}^{\times}(\boldsymbol{s}^C) = \boldsymbol{s}^A$ and $\boldsymbol{s}^C = \boldsymbol{\gamma}^{\times}(\boldsymbol{s}^A)$.

Appendix B. Proofs for Section 5 (Up-to techniques)

Lemma 5.3 *(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 $\mu f = i(\mu f_{|u(L)}) = \mu(f \circ u)$.*

$$\begin{tikzcd} L \arrow[loop above, "f"] \arrow[loop left, "f\circ u"] \arrow[r, shift right, "u"'] & u(L) \arrow[l, shift right, "i"'] \arrow[loop right, "f_{|u(L)}"] \end{tikzcd}$$

Proof. 1. We have that for all $l \in u(L)$, the f-image $f(l) \in u(L)$. Let $l \in u(L)$, i.e., $l = u(l')$ for some $l' \in L$. Observe that

$$\begin{aligned} f(l) &\sqsubseteq u(f(l)) && \text{[by extensiveness]} \\ &\sqsubseteq f(u(l)) && \text{[by compatibility]} \\ &= f(u(u(l'))) && \\ &= f(u(l')) && \text{[by idempotency]} \\ &= f(l) && \end{aligned}$$

Hence $f(l) = u(f(l))$, which means that $f(l) \in u(L)$.

2. We first prove that $\nu f = \nu f_{|u(L)}$. Consider

$$\begin{tikzcd} L \arrow[loop below, "f"] \arrow[r, shift left, "\alpha=u"] & u(L) \arrow[l, shift left, "\gamma=i"] \arrow[loop below, "f_{|u(L)}"] \end{tikzcd}$$

Note that for all $l \in u(L)$, we have $f(\gamma(l)) = f(l) = \gamma(f_{|u(L)}(l))$, i.e., γ satisfies soundness (A.1) and completeness (A.2) in Lemma A.1. Therefore, $\nu f = \gamma(\nu f_{|u(L)}) = \eta f_{|u(L)}$, as desired.

Next we prove that $\nu(f \circ u) = \nu f_{|u(L)}$ Consider

$$\begin{tikzcd} L \arrow[loop below, "f\circ u"] \arrow[r, shift left, "\alpha=u"] & u(L) \arrow[l, shift left, "\gamma=i"] \arrow[loop below, "f_{|u(L)}"] \end{tikzcd}$$

Again, for all $l \in u(L)$, we have $f \circ u(\gamma(l)) = f(u(l)) = f(l) = \gamma(f_{|u(L)}(l))$, i.e., γ satisfies soundness (A.1) and completeness (A.2) in Lemma A.1. Therefore, $\nu(f \circ u) = \gamma(\nu f_{|u(L)}) = \nu f_{|u(L)}$, as desired.

Finally, if u is continuous and strict then also $\gamma = i$ is so: First, since $\bot = u(\bot) \in u(L)$ and hence the inclusion i maps $\bot$ to $\bot$. Second, since u is continuous, directed suprema in both lattices coincide: let $D \subseteq u(L)$, then $\bigsqcup D = \bigsqcup\{u(d) \mid d \in D\} = u(\bigsqcup D) \in u(L)$. Hence i preserves directed suprema.
Hence we get the previous results also for least fixpoints. $\square$

Lemma 5.5 *(properties of $\bar{u}$). Let $u : L \to L$ be a monotone function. Then*

1. *$\bar{u}$ is the least closure larger than u;*
2. *if u is f-compatible then $\bar{u}$ is;*
3. *if u is continuous and strict then $\bar{u}$ is.*

Proof. 1. We first observe that $\bar{u}$ is a closure. For extensiveness, just observe that $\hat{u}_x(y) = u(y) \sqcup x \sqsupseteq x$ for all $y \in L$ and thus obviously $\bar{u}(x) = \mu(\hat{u}_x) \sqsupseteq x$.
In order to show that $\bar{u}$ is idempotent, note that, by extensiveness, $\bar{u} \sqsubseteq \bar{u} \circ \bar{u}$. Hence to conclude, we just need to prove the converse inequality $\bar{u} \circ \bar{u} \sqsubseteq \bar{u}$. For all $x \in L$, we have $\bar{u}(\bar{u}(x)) = \mu(\hat{u}_{\bar{u}(x)}) = \hat{u}^{\gamma}_{\bar{u}(x)}$ for some ordinal γ. We prove, by transfinite induction that for all α, that $\hat{u}^{\alpha}_{\bar{u}(x)} \sqsubseteq \bar{u}(x)$.

($\alpha = 0$) We have that $\hat{u}^{0}_{\bar{u}(x)} = \bot \sqsubseteq \bar{u}(x)$.

($\alpha \to \alpha+1$) We have that

$$\begin{aligned}
\hat{u}^{\alpha+1}_{\bar{u}(x)} &= \hat{u}_{\bar{u}(x)}(\hat{u}^{\alpha}_{\bar{u}(x)}) \\
&= u(\hat{u}^{\alpha}_{\bar{u}(x)}) \sqcup \bar{u}(x) && [\text{by def. } \hat{u}_{\bar{u}(x)}] \\
&\sqsubseteq u(\bar{u}(x)) \sqcup \bar{u}(x) && [\text{by ind. hyp.}] \\
&\sqsubseteq \hat{u}_x(\bar{u}(x)) \sqcup \bar{u}(x) && [\text{since } u \sqsubseteq \hat{u}_x] \\
&= \bar{u}(x) \sqcup \bar{u}(x) && [\text{since } \hat{u}_x(\bar{u}(x)) = \bar{u}(x)] \\
&= \bar{u}(x)
\end{aligned}$$

(α limit) We have that

$$\begin{aligned}
\hat{u}^{\alpha}_{\bar{u}(x)} &= \bigsqcup_{\beta<\alpha} \hat{u}^{\beta}_{\bar{u}(x)} \\
&\sqsubseteq \bigsqcup_{\beta<\alpha} \bar{u}(x) && [\text{by ind. hyp.}] \\
&= \bar{u}(x)
\end{aligned}$$

Moreover, $\bar{u}$ is larger than u, i.e., $u \sqsubseteq \bar{u}$. In fact,

$$\begin{aligned}
\bar{u}(x) &= \hat{u}_x(\bar{u}(x)) && [\text{since } \bar{u}(x) \text{ is a fixpoint of } \hat{u}_x] \\
&= u(\bar{u}(x)) \sqcup x && [\text{by def. of } \hat{u}_x] \\
&\sqsupseteq u(x) \sqcup x && [\text{since } \bar{u} \text{ is extensive}] \\
&\sqsupseteq u(x)
\end{aligned}$$

Finally, let v any closure such that $u \sqsubseteq v$. We show that for all $x \in L$, $\hat{u}^{\alpha}_x \sqsubseteq v(x)$, whence $\bar{u}(x) \sqsubseteq v(x)$, as desired.
($\alpha = 0$) We have that $\hat{u}^0_{\bar{u}(x)} = \bot \sqsubseteq v(x)$.

($\alpha \to \alpha+1$) We have that

$$\begin{aligned}
\hat{u}^{\alpha+1}_{\bar{u}(x)} &= \hat{u}_x(\hat{u}^{\alpha}_x) \\
&= u(\hat{u}^{\alpha}_x) \sqcup x && [\text{by def. } \hat{u}_x] \\
&\sqsubseteq u(v(x)) \sqcup x && [\text{by ind. hyp.}] \\
&\sqsubseteq v(v(x)) \sqcup x && [\text{since } u \sqsubseteq v] \\
&= v(x) \sqcup x && [\text{by idempotency of } v] \\
&= v(x) && [\text{by extensiveness of } v]
\end{aligned}$$

(α limit) We have that

$$\begin{aligned}
\hat{u}^{\alpha}_x &= \bigsqcup_{\beta<\alpha} \hat{u}^{\beta}_x \\
&\sqsubseteq \bigsqcup_{\beta<\alpha} v(x) && [\text{by ind. hyp.}] \\
&= v(x)
\end{aligned}$$

2. Observe that for all $x \in L$, we have $\bar{u}(f(x)) = \hat{u}^{\gamma}_{f(x)}$ for some ordinal γ. Hence also here we proceed by transfinite induction, showing that for all α

$$\hat{u}^{\alpha}_{f(x)} \sqsubseteq f(\bar{u}(x))$$

($\alpha = 0$) We have that $\hat{u}^0_{f(x)} = \bot \sqsubseteq f(\bar{u}(x))$.

($\alpha \to \alpha+1$) We have that

$$\begin{aligned}
\hat{u}^{\alpha+1}_{f(x)} &= \hat{u}_{f(x)}(\hat{u}^{\alpha}_{f(x)}) \\
&\sqsubseteq \hat{u}_{f(x)}(f(\bar{u}(x))) && \text{[by ind. hyp.]} \\
&= u(f(\bar{u}(x))) \sqcup f(x) && \text{[by def. of } \hat{u}_{f(x)}] \\
&\sqsubseteq f(u(\bar{u}(x))) \sqcup f(x) && \text{[by compatibility of } f] \\
&\sqsubseteq f(u(\bar{u}(x)) \sqcup x) && \text{[by general properties of } \sqcup] \\
&= f(\hat{u}_x(\bar{u}(x)))) && \text{[by def. of } \hat{u}_x] \\
&= f(\bar{u}(x)) && \text{[since } \hat{u}(x) \text{ is a fixpoint]}
\end{aligned}$$

(α limit) We have that

$$\begin{aligned}
\hat{u}^{\alpha}_{f(x)} &= \bigsqcup_{\beta<\alpha} \hat{u}^{\beta}_{f(x)} \\
&\sqsubseteq \bigsqcup_{\beta<\alpha} f(\bar{u}(x)) && \text{[by ind. hyp.]} \\
&= f(\bar{u}(x))
\end{aligned}$$

3. Assume that u is continuous and strict. Then $\hat{u}_x$ is continuous for all $x \in L$. In fact, for each directed set $D \subseteq L$ we have

$$\begin{aligned}
\hat{u}_x(\bigsqcup D) &= u(\bigsqcup D) \sqcup x \\
&= \bigsqcup\{u(d) \mid d \in D\}) \sqcup x \\
&= \bigsqcup\{u(d) \sqcup x \mid d \in D\}) \\
&= \bigsqcup\{\hat{u}_x(d) \mid d \in D\})
\end{aligned}$$

Now, we can show that $\bar{u}$ is continuous. Let $D \subseteq L$ be a directed set. We have to prove that $\bar{u}(\bigsqcup D) = \bigsqcup_{d\in D} \bar{u}(d)$. It is sufficient to prove that $\bar{u}(\bigsqcup D) \sqsubseteq \bigsqcup_{d\in D} \bar{u}(d)$, as the other inequality follows by monotonicity and general properties of $\bigsqcup$. As usual, we recall that $\bar{u}(\bigsqcup D) = \hat{u}^{\gamma}_{\bigsqcup D}$ for some γ and thus show, by transfinite induction on α that

$$\hat{u}^{\alpha}_{\bigsqcup D} \sqsubseteq \bigsqcup_{d\in D} \bar{u}(d).$$

($\alpha = 0$) We have that $\hat{u}^{0}_{\bigsqcup D} = \bot \sqsubseteq \bigsqcup_{d\in D} \bar{u}(d)$.

($\alpha \to \alpha+1$) We have that

$$\begin{aligned}
\hat{u}^{\alpha+1}_{\bigsqcup D} &= \hat{u}_{\bigsqcup D}(\hat{u}^{\alpha}_{\bigsqcup D}) \\
&\sqsubseteq \hat{u}_{\bigsqcup D}(\bigsqcup_{d\in D} \bar{u}(d)) && \text{[by ind. hyp.]} \\
&= \bigsqcup_{d\in D} \hat{u}_{\bigsqcup D}(\bar{u}(d)) && \text{[by continuity of } \hat{u}_{\bigsqcup D}] \\
&= \bigsqcup_{d\in D} (u(\bar{u}(d)) \sqcup \bigsqcup D) && \text{[by def. of } \hat{u}_{\bigsqcup D}] \\
&\sqsubseteq \bigsqcup_{d\in D} (\hat{u}_d(\bar{u}(d)) \sqcup \bigsqcup D) && \text{[since } u \sqsubseteq \hat{u}_d] \\
&= \bigsqcup_{d\in D} (\bar{u}(d) \sqcup \bigsqcup D) && \text{[since } \hat{u}(d) \text{ is a fixpoint]} \\
&= \bigsqcup_{d\in D} (\bar{u}(d) \sqcup d) \\
&= \bigsqcup_{d\in D} \bar{u}(d) && \text{[by extensiveness of } \bar{u}]
\end{aligned}$$

(α limit) We have that

$$\begin{aligned}
\hat{u}^{\alpha}_{\sqcup D} &= \bigsqcup_{\beta<\alpha} \hat{u}^{\beta}_{\sqcup D} \\
&\sqsubseteq \bigsqcup_{\beta<\alpha} \bigsqcup_{d\in D} \bar{u}(d) && \text{[by ind. hyp.]} \\
&= \bigsqcup_{d\in D} \bar{u}(d)
\end{aligned}$$

Furthermore, $\bar{u}$ is strict since $\hat{u}_{\perp}(\perp) = u(\perp) \sqcup \perp = \perp \sqcup \perp = \perp$, and thus $\bar{u}(\perp) = \mu(\hat{u}_{\perp}) = \perp$. □

Theorem 5.8 *(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 $\bar{\boldsymbol{u}} = (\bar{u}_1, \dots, \bar{u}_m)$ 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 $\boldsymbol{s}' = \boldsymbol{s}$.*

Proof. Immediate extension to systems of the proofs of the Lemma 5.3 and Corollary 5.6, exploiting Theorem 4.1. □

Theorem 5.11 *(preserving solutions with up-to). Let E be a system of m equations of the kind $\boldsymbol{x} =_{\boldsymbol{\eta}} \boldsymbol{f}(\boldsymbol{x})$ over a complete lattice L. Let $\boldsymbol{u}$ be an m-tuple of up-to functions compatible for E. The solution of the system $d(E, \boldsymbol{u})$ is $sol(d(E, \boldsymbol{u})) = (sol(E), sol(E))$.*

Proof. We proceed by induction on the length m of the original system. The base case is vacuously true since, for $m = 0$, both systems have empty solution. Then, for $m > 0$, assume that the property holds for systems of size $m - 1$. By definition of solution we have that the solution of x_m is

$$sol_{2m}(d(E, \boldsymbol{u})) = \eta_m(\lambda x.\ f_m(sol_{1,m}(d(E, \boldsymbol{u})[x_m := x])))$$

and the parametric solution of y_m is the function $s' : L^m \to L$

$$s'(\boldsymbol{x}') = sol_m(d(E, \boldsymbol{u})[\boldsymbol{x} := \boldsymbol{x}']) = \mu(\lambda y.\, u_m(y) \sqcup x'_m).$$

Observe that since $s'(\boldsymbol{x}')$ depends only on x'_m, we can define the parametric solution of y_m using just a function $s : L \to L$ instead of s'

$$s(x) = \mu(\lambda y.\, u_m(y) \sqcup x).$$

Substituting the parametric solution of y_m in the solution of x_m we obtain

$$sol_{2m}(d(E, \boldsymbol{u})) = \eta_m(\lambda x.\ f_m(sol_{1,m-1}(d(E, \boldsymbol{u})[x_m := x][y_m := s(x)]), s(x))).$$

Let $h(x) = f_m(sol_{1,m-1}(d(E, \boldsymbol{u})[x_m := x][y_m := s(x)]), s(x))$ and $g_x(y) = u_m(y) \sqcup x$, so that $sol_{2m}(d(E, \boldsymbol{u})) = \eta_m(h)$ and $s(x) = \mu(g_x)$. Clearly h and g are both monotone (hence s as well). The former because the solutions of a system (see [3]) and $\boldsymbol{f}$ are monotone, the latter because both u_m and the supremum are. Also notice that s is an extensive function, i.e., $x \sqsubseteq s(x)$ for all x. In fact, since s computes a (least) fixpoint we have that $s(x) = u_m(s(x)) \sqcup x$, and clearly $x \sqsubseteq u_m(s(x)) \sqcup x$ by definition of supremum. Furthermore, we can prove that s is compatible (with respect to h, i.e., $s(h(x)) \sqsubseteq h(s(x))$ for all x), continuous, and strict, whenever u_m satisfies those conditions, respectively. First, if u_m is continuous, then so is g in both variables, since $\sqcup$ is continuous. Then, since $s(x)$ is the least fixpoint of g_x, it is immediate that s is continuous as well. Recalling that $s(x) = g_x^{\alpha}(\perp)$ for some ordinal α, both remaining properties can be proved by transfinite induction on $g_x^{\alpha}(\perp)$ for every α. First we show that for all x, $g_{h(x)}^{\alpha}(\perp) \sqsubseteq h(s(x))$ for every ordinal α (hence $s(h(x)) \sqsubseteq h(s(x))$). For $\alpha = 0$, we have $g_{h(x)}^{0}(\perp) = \perp \sqsubseteq h(s(x))$. For a successor ordinal $\alpha = \beta + 1$, we have $g_{h(x)}^{\beta+1}(\perp) = g_{h(x)}(g_{h(x)}^{\beta}(\perp))$, and by inductive hypothesis we know that $g_{h(x)}^{\beta}(\perp) \sqsubseteq h(s(x))$. Then

$$\begin{aligned}
& g_{h(x)}(g_{h(x)}^{\beta}(\perp)) \\
\sqsubseteq\quad & \text{[since } g \text{ is monotone]} \\
& g_{h(x)}(h(s(x))) \\
=\quad & \text{[by definition of } g\text{]} \\
& u_m(h(s(x))) \sqcup h(x) \\
=\quad & \text{[by definition of } h\text{]}
\end{aligned}$$

$$
\begin{aligned}
& u_m(f_m(sol_{1,m-1}(d(E,\boldsymbol{u})[x_m := s(x)][y_m := s^2(x)]), s^2(x))) \sqcup h(x) \\
\sqsubseteq \quad & [\text{by compatibility of } \boldsymbol{u}] \\
& f_m(\boldsymbol{u}^\times(sol_{1,m-1}(d(E,\boldsymbol{u})[x_m := s(x)][y_m := s^2(x)]), s^2(x))) \sqcup h(x)
\end{aligned}
$$

Observe that $u_m(s(z)) \sqsubseteq s(z) = g_z(s(z)) = u_m(s(z)) \sqcup z$ for all z. A similar reasoning applies to the other solutions as well, obtaining that

$$u_i(sol_i(d(E,\boldsymbol{u})[x_m := s(x)][y_m := s^2(x)])) \sqsubseteq sol_i(d(E,\boldsymbol{u})[x_m := s(x)][y_m := s^2(x)])$$

for all $i \in \underline{m-1}$. Therefore we have

$$
\begin{aligned}
& f_m(prd\boldsymbol{u}(sol_{1,m-1}(d(E,\boldsymbol{u})[x_m := s(x)][y_m := s^2(x)]), s^2(x))) \sqcup h(x) \\
\sqsubseteq \quad & [\text{since } f_m \text{ is monotone}] \\
& f_m(sol_{1,m-1}(d(E,\boldsymbol{u})[x_m := s(x)][y_m := s^2(x)]), s^2(x)) \sqcup h(x) \\
= \quad & [\text{by definition of } h] \\
& h(s(x)) \sqcup h(x) \\
\sqsubseteq \quad & [\text{since } h \text{ is monotone}] \\
& h(s(x) \sqcup x) \\
= \quad & [\text{since } s \text{ is extensive}] \\
& h(s(x))
\end{aligned}
$$

And so we established that $g^{\beta+1}_{h(x)}(\bot) \sqsubseteq h(s(x))$. For α limit ordinal, by inductive hypothesis we immediately have that $g^{\alpha}_{h(x)}(\bot) = \bigsqcup_{\beta<\alpha} g^{\beta}_{h(x)}(\bot) \sqsubseteq \bigsqcup h(s(x)) = h(s(x))$. Now we show that $g^{\alpha}_{\bot}(\bot) = \bot$ for every ordinal α. For $\alpha = 0$, we have $g^{0}_{\bot}(\bot) = \bot$. For $\alpha = \beta + 1$, by inductive hypothesis we have that $g^{\beta+1}_{\bot}(\bot) = g_\bot(g^{\beta}_{\bot}(\bot)) = g_\bot(\bot)$. And in turn, $g_\bot(\bot) = u_m(\bot) \sqcup \bot = \bot$, since u_m is strict. For α limit ordinal, by inductive hypothesis we obtain that $g^{\alpha}_{\bot}(\bot) = \bigsqcup_{\beta<\alpha} g^{\beta}_{\bot}(\bot) = \bigsqcup \bot = \bot$. Now we have two different cases depending on η_m.

- $\eta_m = \nu$
  In this case $sol_{2m}(d(E,\boldsymbol{u})) = h^\alpha(\top)$ for some ordinal α. Here we show that actually $s(h^\alpha(\top)) = h^\alpha(\top)$ for every ordinal α. Since as we mentioned above s is extensive, we just need to prove that $s(h^\alpha(\top)) \sqsubseteq h^\alpha(\top)$ for every ordinal α. We proceed by transfinite induction on α. For $\alpha = 0$, we have $s(h^0(\top)) \sqsubseteq \top = h^0(\top)$. If α is a successor ordinal $\beta + 1$, assuming the property holds for β, we show that $s(h^{\beta+1}(\top)) \sqsubseteq h^{\beta+1}(\top)$. Since h is monotone, by inductive hypothesis we have that $h(s(h^\beta(\top))) \sqsubseteq h(h^\beta(\top)) = h^{\beta+1}(\top)$. Recalling that $s(h(x)) \sqsubseteq h(s(x))$ for all x, we also have that $s(h^{\beta+1}(\top)) = s(h(h^\beta(\top))) \sqsubseteq h(s(h^\beta(\top)))$. When α is a limit ordinal we have that $h^\alpha(\top) = \bigsqcap_{\beta<\alpha} h^\beta(\top)$. Since s is monotone, we have that $s(h^\alpha(\top)) = s(\bigsqcap_{\beta<\alpha} h^\beta(\top)) \sqsubseteq \bigsqcap_{\beta<\alpha} s(h^\beta(\top))$. And since by inductive hypothesis $s(h^\beta(\top)) \sqsubseteq h^\beta(\top)$ for all $\beta < \alpha$, we conclude also that $\bigsqcap_{\beta<\alpha} s(h^\beta(\top)) \sqsubseteq \bigsqcap_{\beta<\alpha} h^\beta(\top)$.
- $\eta_m = \mu$
  In this case $sol_{2m}(d(E,\boldsymbol{u})) = h^\alpha(\bot)$ for some ordinal α. Recall also that since $\eta_m = \mu$, by hypothesis we know that u_m is continuous and strict. In such case, as shown above, s is continuous and strict as well. Again, we already know that s is extensive, so we just prove by transfinite induction that $s(h^\alpha(\bot)) \sqsubseteq h^\alpha(\bot)$ for every ordinal α. For $\alpha = 0$, we have $s(h^0(\bot)) = s(\bot) = \bot$, since s is strict. If α is a successor ordinal $\beta + 1$, assuming the property holds for β, we show that $s(h^{\beta+1}(\bot)) \sqsubseteq h^{\beta+1}(\bot)$. Since h is monotone, by inductive hypothesis we have that $h(s(h^\beta(\bot))) \sqsubseteq h(h^\beta(\bot)) = h^{\beta+1}(\bot)$. Recalling that $s(h(x)) \sqsubseteq h(s(x))$ for all x, we also have that $s(h^{\beta+1}(\bot)) = s(h(h^\beta(\bot))) \sqsubseteq h(s(h^\beta(\bot)))$. When α is a limit ordinal we have that $h^\alpha(\bot) = \bigsqcup_{\beta<\alpha} h^\beta(\bot)$. Since s is continuous, we have that $s(h^\alpha(\bot)) = s(\bigsqcup_{\beta<\alpha} h^\beta(\bot)) = \bigsqcup_{\beta<\alpha} s(h^\beta(\bot))$. And since by inductive hypothesis $s(h^\beta(\bot)) \sqsubseteq h^\beta(\bot)$ for all $\beta < \alpha$, we conclude also that $\bigsqcup_{\beta<\alpha} s(h^\beta(\bot)) \sqsubseteq \bigsqcup_{\beta<\alpha} h^\beta(\bot)$.

So in both cases we have $s(h^\alpha(\top)) = h^\alpha(\top)$ or $s(h^\alpha(\bot)) = h^\alpha(\bot))$, respectively, for every ordinal α. Consider the function $h'(x) = f_m(sol_{1,m-1}(d(E,\boldsymbol{u})[x_m := x][y_m := x]), x)$. The previous fact implies that actually $\eta_m(h') = \eta_m(h) = sol_{2m}(d(E,\boldsymbol{u}))$. Furthermore, for the same reason we have that $s(sol_{2m}(d(E,\boldsymbol{u}))) = sol_{2m}(d(E,\boldsymbol{u}))$. Since $sol_{2m}(d(E,\boldsymbol{u}))$ is the solution of x_m and by definition of solution $s(sol_{2m}(d(E,\boldsymbol{u}))) = sol_m(d(E,\boldsymbol{u}))$ is that of y_m, this means that x_m and y_m have the same solution in $d(E,\boldsymbol{u})$. So we can rewrite the solutions of x_m and y_m as $\eta_m(h')$, that is

$$sol_{2m}(d(E,\boldsymbol{u})) = sol_m(d(E,\boldsymbol{u})) = \eta_m(\lambda x.\, f_m(sol_{1,m-1}(d(E,\boldsymbol{u})[x_m := x][y_m := x]), x)).$$

Now, observe that the system $d(E,\boldsymbol{u})[x_m := x][y_m := x]$ is actually $d(E[x_m := x], \boldsymbol{u}_{1,m-1})$. Therefore, since $E[x_m := x]$ has size $m-1$, by inductive hypothesis we know that

$$\begin{aligned} sol_{1,m-1}(d(E,\boldsymbol{u})[x_m := x][y_m := x]) &= sol_{m,2m-2}(d(E,\boldsymbol{u})[x_m := x][y_m := x]) \\ &= sol(E[x_m := x]). \end{aligned}$$

Thus, substituting these solutions in those of x_m and y_m above, we obtain

$$sol_{2m}(d(E,\boldsymbol{u})) = sol_m(d(E,\boldsymbol{u})) = \eta_m(\lambda x.\, f_m(sol(E[x_m := x]), x))$$

which is also the definition of the solution of x_m in E. Which means that $sol_{2m}(d(E,\boldsymbol{u})) = sol_m(d(E,\boldsymbol{u})) = sol_m(E)$. Then, the remaining solutions are

$$\begin{aligned} &(sol_{1,m-1}(d(E,\boldsymbol{u})), sol_{m+1,2m-1}(d(E,\boldsymbol{u}))) \\ &= sol(d(E,\boldsymbol{u})[x_m := sol_{2m}(d(E,\boldsymbol{u}))][y_m := sol_m(d(E,\boldsymbol{u}))]) && \text{[by definition of solution]} \\ &= sol(d(E,\boldsymbol{u})[x_m := sol_m(E)][y_m := sol_m(E)]) \\ &= (sol(E[x_m := sol_m(E)]), sol(E[x_m := sol_m(E)])) && \text{[by inductive hypothesis]} \\ &= (sol_{1,m-1}(E), sol_{1,m-1}(E)) && \text{[by definition of solution]} \end{aligned}$$

This and the previous fact allow us to conclude that

$$sol(d(E,\boldsymbol{u})) = (sol_{1,m-1}(E), sol_m(E), sol_{1,m-1}(E), sol_m(E))$$

that is indeed $sol(d(E,\boldsymbol{u})) = (sol(E), sol(E))$. □

Appendix C. Proofs for Section 6 (Solving systems of equations via games)

Theorem 6.2 *(correctness and completeness). Let E be a system of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}(\boldsymbol{x})$ 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).*

Proof. Define $\langle \alpha, \gamma \rangle : \mathbf{2}^{B_L} \to L$, by letting $\alpha(X) = \bigsqcup X$ for $X \in \mathbf{2}^{B_L}$ and $\gamma(l) = \downarrow l \cap B_L$ for $l \in L$. It is immediate to see that this is a Galois insertion: for all $X \in \mathbf{2}^{B_L}$ we have $X \subseteq \gamma(\alpha(X)) = (\downarrow \bigsqcup X) \cap B_L$ and, for $l \in L$ we have $l = \alpha(\gamma(l)) = \bigsqcup(\downarrow l \cap B_L)$.

Below we abuse the notation and write $\downarrow$ and $\bigsqcup$ for the m-tuples where each function is $\downarrow$ and $\bigsqcup$ applied componentwise, respectively.

$$\begin{tikzcd} ((\mathbf{2}^{B_L})^m, \subseteq) \arrow[loop below, "\boldsymbol{f}^C = \downarrow \boldsymbol{f} \sqcup"] \arrow[r, shift right, "\boldsymbol{\alpha} = \sqcup\_"'] & L^m \arrow[l, shift right, "\boldsymbol{\gamma} = \downarrow\_ \cap B_L"'] \arrow[loop below, "\boldsymbol{f}"] \end{tikzcd}$$

Define a "concrete" system $\boldsymbol{x} =_{\eta} \boldsymbol{f}^C(\boldsymbol{x})$ where $\boldsymbol{f}^C = \boldsymbol{\gamma}^{\times} \circ \boldsymbol{f} \circ \boldsymbol{\alpha}^{\times} : (\mathbf{2}^{B_L})^m \to (\mathbf{2}^{B_L})^m$. Then we can use Lemma A.5 to deduce that, if we denote by $\boldsymbol{S}^C$ the solution of the concrete system and by $\boldsymbol{s}$ the solution of the original system, we have $\boldsymbol{S}^C = \downarrow \boldsymbol{s} \cap B_L^m$.

Now, $(\mathbf{2}^{B_L}, \subseteq)$ is an algebraic, hence continuous lattice. Therefore, by [3, Theorem 4.8], the lattice game for the "concrete" system on $(\mathbf{2}^{B_L})^m$ is sound and complete.

It is immediate to realise that, if we fix as basis for $\mathbf{2}^{B_L}$ the set of singletons, this corresponds exactly to what we called here the powerset game. In fact, the game aims to show that $\{b\} \subseteq S_i^C = \downarrow s_i$, for some $b \in B_L$ and $i \in \underline{m}$, and this amounts to $b \sqsubseteq s_i$. Positions of $\exists$ are pairs $(\{b\}, i)$ where $b \in B_L$ and $i \in \underline{m}$, and she has to play some tuples $\boldsymbol{X} \in (\mathbf{2}^{B_L})^m$ such that $\{b\} \subseteq f_i^C(\boldsymbol{X}) = \downarrow f_C(\bigsqcup \boldsymbol{X})$ which amounts to $b \sqsubseteq f_C(\bigsqcup \boldsymbol{X})$. Positions of $\forall$ are tuples $\boldsymbol{X} \in (\mathbf{2}^{B_L})^m$ and he chooses some $j \in \underline{m}$ and $b' \in X_j$. This is exactly the powerset game, hence we conclude. □

Theorem 6.5 *(game with selections). Let E be a system of m equations over a complete lattice L of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}(\boldsymbol{x})$ with solution $\boldsymbol{s}$, and let σ be a selection for E. 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) in the game restricted to the selection σ.*

Proof. First, observe that if $b \not\sqsubseteq s_i$ then by Theorem 6.2 we know that $\exists$ has no winning strategy from position (b, i) in the original, unrestricted, game, while $\forall$ would have one. Thus, since the restriction does not apply to the moves of $\forall$, such a winning strategy for $\forall$ is still a valid winning strategy also in the game restricted to the selection, and so, $\exists$ cannot have one.

Now, let $b \sqsubseteq s_i$, hence by Theorem 6.2 $\exists$ has a winning strategy t from position (b, i) in the original game. We show that $\exists$ has a winning strategy in the game restricted to the selection σ, starting from every position (b', i) such that $b' \sqsubseteq b$ (hence also from b itself) independently from the moves of $\forall$. In the original unrestricted game, $\exists$ from position (b, i) would play $\boldsymbol{X} = t(b, i)$ according to the winning strategy t. Since $b' \sqsubseteq b$, we know that $\boldsymbol{X} \in \mathbf{E}(b, i) \subseteq \mathbf{E}(b', i)$, and thus by definition of selection there must exist $\boldsymbol{Y} \in \sigma(b', i)$ such that $\boldsymbol{Y} \sqsubseteq_H^\wedge \boldsymbol{X}$. Now, we have two cases depending on $\boldsymbol{X}$. If all components of $\boldsymbol{X}$ are empty, then also $\boldsymbol{Y}$ must be so, which means that player $\forall$ has no possible move and $\exists$ wins. Otherwise, for every move (b'', j) of $\forall$, such that $b'' \in Y_j \sqsubseteq_H X_j$, there must exist $b''' \in X_j$ such that $b'' \sqsubseteq b'''$. In the original game, this means that $(b''', j) \in \mathbf{A}(t(b, i))$, and since $\exists$ is supposed to win playing according to t, then t must be a winning strategy also from position (b''', j). Having reached a situation analogous to the initial one, it is easy to see that the same reasoning can be repeated indefinitely, until $\exists$ wins or we obtain a pair of infinite plays, one in the original game and the other in the restricted game, such as the following

$$(b_0, i_0) \to \boldsymbol{X}^0 \to (b_1, i_1) \to \boldsymbol{X}^1 \to (b_2, i_2) \to \boldsymbol{X}^2 \to \dots$$
$$(b'_0, i_0) \to \boldsymbol{Y}^0 \to (b'_1, i_1) \to \boldsymbol{Y}^1 \to (b'_2, i_2) \to \boldsymbol{Y}^2 \to \dots$$

where $(b_0, i_0) = (b, i)$, $b'_0 = b'$, and for all k we have $b'_k \sqsubseteq b_k$ and $\boldsymbol{X}^k = t(b_k, i_k) \sqsupseteq_H^\wedge \boldsymbol{Y}^k \in \sigma(b'_k, i_k)$. Clearly the first play above is a play in the original unrestricted game where $\exists$ plays according to the strategy t, hence it must be won by $\exists$. The second play, instead, is clearly a play in the game restricted to the selection σ. Moreover, all the indices i_k appearing along the two plays are the same, and so we can conclude that also the second play is won by $\exists$. $\square$

Lemma 6.9 *(finite moves). Let E be a system of m equations over a complete lattice L without infinite ascending chains. For every position $(b, i) \in B_L \times \underline{m}$ and move $\boldsymbol{X} \in \mathbf{E}(b, i)$, there exists a finite move $\boldsymbol{Y} \in \mathbf{E}(b, i)$ such that $\boldsymbol{Y} \subseteq^\wedge \boldsymbol{X}$.*

Proof. We prove a slightly stronger property, that is, for every set $X \subseteq L$ there exists a finite subset $Y \subseteq X$ such that $\bigsqcup Y = \bigsqcup X$. This immediately implies that for every position $(b, i) \in B_L \times \underline{m}$ and move $\boldsymbol{X} \in \mathbf{E}(b, i)$, there must be a tuple $(Y_1, \dots, Y_m)$ such that each Y_j is a finite subset of X_j and $\bigsqcup Y_j = \bigsqcup X_j$, hence $\boldsymbol{Y} \in \mathbf{E}(b, i)$ and $\boldsymbol{Y} \subseteq^\wedge \boldsymbol{X}$.

First, given $X \subseteq L$, observe that the set $D = \{\bigsqcup Y \mid Y \subseteq X \wedge Y \text{ finite}\}$ of the suprema of the finite subsets of X is directed. Moreover, $\bigsqcup X = \bigsqcup D$. In fact, $X \subseteq D$ since for each $x \in X$, $x = \bigsqcup\{x\}$. Hence $\bigsqcup X \sqsubseteq \bigsqcup D$. Conversely, $\bigsqcup D \sqsubseteq \bigsqcup X$ since for all $d \in D$, $d = \bigsqcup Y$ for some $Y \subseteq X$ and thus $d = \bigsqcup Y \sqsubseteq \bigsqcup X$.

Now, take any $d \in D$. Then, either $d = \bigsqcup D$ or $d \sqsubset \bigsqcup D$. In the latter case there must be $d' \in D$ such that $d' \not\sqsubseteq d$, and since D is directed, there must also be $d'' \in D$ such that $d \sqsubset d \sqcup d' \sqsubseteq d''$. Since this argument can be repeated until a $\hat{d} = \bigsqcup D$ is found, we can conclude that such a $\hat{d}$ must be eventually found, otherwise we would obtain an infinite ascending chain $d_1 \sqsubset d_2 \sqsubset \dots$, contradicting the fact that L does not include any such chain. Finally, since every element of D is the supremum of some finite subset of X, there must be a finite $Y \subseteq X$ whose supremum is $\bigsqcup Y = \hat{d} = \bigsqcup D = \bigsqcup X$. $\square$

Proposition 6.10 *(least selection). Let E be a system of m equations over a complete lattice L with finite height. Then, there exists a unique selection σ such that $\sigma \subseteq_H \sigma'$ for all selections σ'.*

Proof. Consider the function σ defined as follows. For every position $(b, i) \in B_L \times \underline{m}$

$$\sigma(b, i) = \{max\, \boldsymbol{X} \mid \text{finite } \boldsymbol{X} \in \mathbf{E}(b, i) \wedge \forall \boldsymbol{Y} \in \mathbf{E}(b, i).\, \downarrow\boldsymbol{Y} \subseteq^\wedge \downarrow\boldsymbol{X} \Rightarrow \downarrow\boldsymbol{Y} = \downarrow\boldsymbol{X}\}$$

i.e., the tuples of maximals w.r.t. $\sqsubseteq$ of finite moves, whose downward-closure w.r.t. $\sqsubseteq$ is minimal w.r.t. pointwise subset inclusion. Observe that we write $max\, \boldsymbol{X}$ to denote the tuple $(M_1, \dots, M_m)$ where each $M_j = \{x \in X_j \mid \forall x' \in X_j. x \not\sqsubset x'\}$. This is well-defined whenever, as required above, $\boldsymbol{X}$ is finite. We show that (i) σ is a selection, i.e., $\uparrow_H \sigma(b, i) = \mathbf{E}(b, i)$, and (ii) for every selection σ' it holds $\sigma \subseteq_H \sigma'$ and $\sigma' \subseteq_H \sigma \Rightarrow \sigma' = \sigma$.

(i). We have to show that $\uparrow_H \sigma(b, i) = \mathbf{E}(b, i)$. First, it is easy to see that $\sigma(b, i) \subseteq \mathbf{E}(b, i)$, since clearly $\sigma(b, i) \subseteq \{max\, \boldsymbol{X} \mid \text{finite } \boldsymbol{X} \in \mathbf{E}(b, i)\}$ and for every finite move $\boldsymbol{X} \in \mathbf{E}(b, i)$ we have that $\bigsqcup max\, \boldsymbol{X} = \bigsqcup \boldsymbol{X}$, hence $max\, \boldsymbol{X} \in \mathbf{E}(b, i)$ as well. Thus $\uparrow_H \sigma(b, i) \subseteq \uparrow_H \mathbf{E}(b, i) = \mathbf{E}(b, i)$ because the set of possible moves is upward-closed. In order to prove the converse inclusion, for every $\boldsymbol{X} \in \mathbf{E}(b, i)$ we show that there exists $\boldsymbol{Y} \in \sigma(b, i)$ such that $\boldsymbol{Y} \sqsubseteq_H^\wedge \boldsymbol{X}$, thus $\boldsymbol{X} \in \uparrow_H \sigma(b, i)$. By Lemma 6.9 we know that there exists a finite move $\boldsymbol{X}' \in \mathbf{E}(b, i)$ such that $\boldsymbol{X}' \subseteq^\wedge \boldsymbol{X}$, hence $\boldsymbol{X}' \sqsubseteq_H^\wedge \boldsymbol{X}$. Moreover, since $\boldsymbol{X}'$ is finite and L has no infinite descending chain (hence $\downarrow\boldsymbol{X}'$ is finite), there must exist a finite move $\boldsymbol{X}'' \in \mathbf{E}(b, i)$, possibly $\boldsymbol{X}'$ itself, whose downward-closure $\downarrow\boldsymbol{X}''$ is included in $\downarrow\boldsymbol{X}'$ and it is minimal w.r.t. $\subseteq^\wedge$. Let $\boldsymbol{Y} = max\, \boldsymbol{X}''$. Then, we have that $\boldsymbol{Y} \in \sigma(b, i)$ and $\boldsymbol{Y} \sqsubseteq_H^\wedge \boldsymbol{X}'$, since $\boldsymbol{Y} \subseteq^\wedge \downarrow\boldsymbol{X}'' \subseteq^\wedge \downarrow\boldsymbol{X}' \equiv_H \boldsymbol{X}'$. Now we can immediately conclude by transitivity that $\boldsymbol{Y} \sqsubseteq_H^\wedge \boldsymbol{X}$, since we already know that $\boldsymbol{X}' \sqsubseteq_H^\wedge \boldsymbol{X}$.

(ii). We first show that $\sigma \subseteq_H \sigma'$, i.e., for every selection σ' and every move $\boldsymbol{X} \in \sigma(b,i)$, there exists $\boldsymbol{Y} \in \sigma'(b,i)$ such that $\boldsymbol{X} \subseteq^\wedge \boldsymbol{Y}$. Note that since σ' is a selection, given $\boldsymbol{X} \in \sigma(b,i) \subseteq \mathbf{E}(b,i)$ there must exist $\boldsymbol{Y} \in \sigma'(b,i)$ such that $\boldsymbol{Y} \sqsubseteq_H^\wedge \boldsymbol{X}$. This implies that $\downarrow\boldsymbol{Y} \subseteq^\wedge \downarrow\boldsymbol{X}$, since $\downarrow\boldsymbol{Y} \equiv_H \boldsymbol{Y} \sqsubseteq_H^\wedge \boldsymbol{X} \equiv_H \downarrow\boldsymbol{X}$ and, by definition of $\sqsubseteq_H^\wedge$, $\downarrow\boldsymbol{Y} \sqsubseteq_H^\wedge \downarrow\boldsymbol{X}$ implies $\downarrow\boldsymbol{Y} \subseteq^\wedge \downarrow\boldsymbol{X}$. But then, by definition of σ, we must have $\downarrow\boldsymbol{Y} = \downarrow\boldsymbol{X}$, and so we can conclude that $\boldsymbol{X} = \mathit{max}\,\boldsymbol{X} = \mathit{max}\,\boldsymbol{Y} \subseteq^\wedge \boldsymbol{Y}$.

Now, assuming that $\sigma' \subseteq_H \sigma$, we prove that $\sigma' = \sigma$. First, observe that combining this assumption with what we just proved above, we have that $\sigma \subseteq_H \sigma' \subseteq_H \sigma$. Which means that for every $\boldsymbol{X} \in \sigma(b,i)$ there exist $\boldsymbol{Y} \in \sigma'(b,i)$ and $\boldsymbol{X'} \in \sigma(b,i)$ such that $\boldsymbol{X} \subseteq^\wedge \boldsymbol{Y} \subseteq^\wedge \boldsymbol{X}'$. This immediately implies that $\downarrow\boldsymbol{X} \subseteq^\wedge \downarrow\boldsymbol{Y} \subseteq^\wedge \downarrow\boldsymbol{X}'$, which, in turn, by definition of σ, means that $\downarrow\boldsymbol{X} = \downarrow\boldsymbol{X}'$ and thus $\boldsymbol{X} = \mathit{max}\,\boldsymbol{X} = \mathit{max}\,\boldsymbol{X}' = \boldsymbol{X}'$. Recalling that $\boldsymbol{X} \subseteq^\wedge \boldsymbol{Y} \subseteq^\wedge \boldsymbol{X}'$, we deduce $\boldsymbol{X} = \boldsymbol{Y}$. And so we have that $\sigma(b,i) \subseteq \sigma'(b,i)$. We next prove the other inclusion $\sigma'(b,i) \subseteq \sigma(b,i)$. For every $\boldsymbol{Y} \in \sigma'(b,i)$, since $\boldsymbol{Y} \in \mathbf{E}(b,i) = \uparrow_H \sigma(b,i)$, there must exist $\boldsymbol{X} \in \sigma(b,i)$ such that $\boldsymbol{X} \sqsubseteq_H^\wedge \boldsymbol{Y}$. Moreover, since $\sigma' \subseteq_H \sigma$, there must also be $\boldsymbol{X}' \in \sigma(b,i)$ such that $\boldsymbol{X} \sqsubseteq_H^\wedge \boldsymbol{Y} \subseteq^\wedge \boldsymbol{X}'$. Then, by definition of $\sqsubseteq_H$, we have that $\downarrow\boldsymbol{X} \subseteq^\wedge \downarrow\boldsymbol{Y} \subseteq^\wedge \downarrow\boldsymbol{X}'$. From this, reasoning as before, we obtain that $\boldsymbol{X} = \boldsymbol{X}' = \mathit{max}\,\boldsymbol{X}$. Therefore, using the previous facts, we deduce that $\mathit{max}\,\boldsymbol{X} \sqsubseteq_H^\wedge \boldsymbol{Y} \subseteq^\wedge \mathit{max}\,\boldsymbol{X}$. Being all elements in each $(\mathit{max}\,\boldsymbol{X})_j$ incomparable, by definition of $\sqsubseteq_H^\wedge$ we must have $\boldsymbol{Y} = \mathit{max}\,\boldsymbol{X} = \boldsymbol{X}$. And so we conclude that $\sigma'(b,i) \subseteq \sigma(b,i)$. □

Appendix D. Proof for Section 7 (Local algorithm for solving the game)

Definition D.1 *(sound forget).* Whenever function FORGET(Δ_P, Γ_P, $(C, \boldsymbol{k})$) is invoked, it returns $\Delta'_P \subseteq \Delta_P$ such that for every decision $(C', \boldsymbol{k}') \in \Delta'_P$, for every position C'' justifying that decision, there exists $(C'', \boldsymbol{k}'') \in \Delta'_P$ such that $\boldsymbol{k}'' \leq_P \mathit{next}(\boldsymbol{k}', \mathrm{i}(C'))$ or there exists $(C'', \boldsymbol{k}'') \in \Gamma_P \smallsetminus \{(C, \boldsymbol{k})\}$ such that $\boldsymbol{k}'' <_P \mathit{next}(\boldsymbol{k}', \mathrm{i}(C'))$.

Lemma D.2 *(assumptions and plays). Given a powerset game, whenever functions* EXPLORE($\cdot$, $\cdot$, ρ, Γ, Δ) *and* BACKTRACK($\cdot$, $\cdot$, ρ, Γ, Δ) *are invoked, for every player P, for all $(C, \boldsymbol{k}) \in \Gamma_P$ it holds $(C, \boldsymbol{k}, \pi) \in \rho$ for some π.*

Proof. Easily proved by an inspection of the code. Initially, on the call EXPLORE(C_0, $\mathbf{0}$, [], $(\emptyset,\emptyset)$, $(\emptyset,\emptyset)$), the property vacuously holds since both $\Gamma_\exists$ and $\Gamma_\forall$ are empty. Now, the only way that could make the property fail is by adding new assumptions or backtracking, hence shortening the playlist ρ. The only position in the code where new assumptions are added is in the function EXPLORE. A new assumption $(C, \boldsymbol{k}')$ is added only if $(C, \boldsymbol{k}', \pi) \in \rho$, for some π, thus the property still holds. On the other hand, the only place where the backtracking really happens, that is, ρ is effectively shorten, is at the end of the backtracking function, when BACKTRACK(P, C', t, Γ, Δ) is invoked. More precisely, the head $(C', \boldsymbol{k}', \pi)$ is removed from the playlist ρ. However, before the aforementioned invocation, $(C', \boldsymbol{k}')$ was already removed from Γ_P and from $\Gamma_{\overline{P}}$, if it were in $\Gamma_{\overline{P}}$. And so again the property still holds. □

Lemma 7.2 *(termination). Given a powerset game on a finite lattice, any call* EXPLORE(C_0, $\mathbf{0}$, [], $(\emptyset,\emptyset)$, $(\emptyset,\emptyset)$) *terminates with an invocation of* BACKTRACK(P, C_0, [], $(\emptyset,\emptyset)$, Δ) *for some player P and a set Δ.*

Proof. Consider the sequence σ of invocations to functions EXPLORE and BACKTRACK in the order they happen, originating from a call EXPLORE(C_0, $\mathbf{0}$, [], $(\emptyset,\emptyset)$, $(\emptyset,\emptyset)$). Let τ be the subsequence of σ obtained removing all calls to BACKTRACK. We show that such sequence is finite. First, since the lattice is finite, hence *Pos* is finite, the set of playlists ρ in the invocations in τ is also finite. Actually, this is not true in general for any set of playlists, but it holds for the set of lists we obtain during any computation. Indeed, this can be seen inductively, showing that every playlist ρ has length bounded by $|\mathit{Pos}|$. At the beginning we have the empty list [] which is clearly bounded by $|\mathit{Pos}|$. Then, by inspecting the code it can be seen that the only function which increases the size of ρ is EXPLORE, and it happens only if the current position C, with counter $\boldsymbol{k}$, is not already contained in ρ with a counter $\boldsymbol{k}'$ such that $\boldsymbol{k}' <_P \boldsymbol{k}$ for some player P. But whenever a position C already in ρ is encountered again it must be with a counter strictly larger for one of the players. The only case where this could possibly fail is when the subsequence of ρ between the two occurrences of C contains only positions with priority 0. But, as already mentioned, this cannot happen because players alternate during the game and only $\forall$ has positions with priority 0. Thus, every time a position recurs, the playlist is not extended any more. So, the size of the playlist is necessarily bounded by the size of *Pos*. Furthermore, the set of playlists of length bounded by $|\mathit{Pos}|$ is finite because every π in them is bounded as well, since $\pi \subseteq \mathit{Pos}$, and the same happens for the counters $\boldsymbol{k}$ since they are computed starting from $\mathbf{0}$ and increased at most by 1 in some component only when the list is extended. Therefore, τ must contain only a finite number of different playlists ρ, possibly with repetitions. Now, in order to show that τ is finite, we define a partial order $\leq$ over the playlists in τ as follows, $\forall \rho, \rho', \rho'', C, \boldsymbol{k}, \pi, \pi'$:

- $\rho'\rho \leq \rho$
- if $\pi \subsetneq \pi'$, then $\rho''((C, \boldsymbol{k}, \pi) :: \rho) \leq \rho'((C, \boldsymbol{k}, \pi') :: \rho)$.

It is easy to see that such order is reflexive, antisymmetric, and transitive. Since the set of playlists in τ is finite, so is the corresponding poset with the given partial order. By an inspection of the code it can be seen that for every two playlists ρ, ρ' in consecutive invocations of EXPLORE in τ, we have that $\rho' < \rho$, since:

- function Explore extends the playlist ρ until function Backtrack is invoked
- function Backtrack shortens the playlist ρ until it is empty or function Explore is invoked, after shortening the set of unexplored moves π in ρ.

So the playlists in τ form a strictly descending chain in a finite poset, thus τ must be finite. And this immediately proves that σ is finite as well, because otherwise from a certain point on we would have infinitely many calls to Backtrack only, which would shorten the playlist infinitely many times. And so we can conclude that any computation originating from a call Explore$(C_0, \mathbf{0}, [], (\emptyset,\emptyset), (\emptyset,\emptyset))$ must terminate. Finally, since the only instruction returning a value (hence terminating the execution) is in the function Backtrack and it is reached only when $\rho = []$, then Backtrack$(P, C, [], \Gamma, \Delta)$ must have been invoked on some P, C, Γ, Δ. Furthermore, $C = C_0$ because $\rho = []$ is the list of positions from the initial position C_0 to the current node C.

We immediately conclude that $\Gamma = (\emptyset,\emptyset)$ by exploiting Lemma D.2. □

Lemma D.3 *(backtracking position). Given a powerset game, whenever function* Backtrack$(P, C, \rho, \Gamma, \Delta)$ *is invoked, it holds* $(C, \boldsymbol{k}) \in \Delta_P \cup \Gamma_P$ *for some* $\boldsymbol{k}$.

Proof. Immediate by inspecting the invocations of Backtrack in the code. □

Lemma D.4 *(uncontrolled decisions). Given a powerset game, whenever functions* Explore$(\cdot,\cdot,\cdot,\Gamma,\Delta)$ *and* Backtrack$(\cdot,\cdot,\cdot,\Gamma,\Delta)$ *are invoked, for every player P, for all $(C, \boldsymbol{k}) \in \Delta_P$, if $\mathsf{P}(C) \neq P$, then for all $C' \in \mathsf{M}(C)$ it holds $(C', \boldsymbol{k}') \in \Delta_P \cup \Gamma_P$ for some $\boldsymbol{k}'$.*

Proof. By inspecting the code it is easy to see that every time we add a new decision $(C, \boldsymbol{k})$ for a player P that is not the owner of C, either:

- $\mathsf{M}(C) = \emptyset$, thus the property vacuously holds, or
- the procedure already explored all possible moves $\mathsf{M}(C)$ and they all became decisions or assumptions for P, since we are in the case where $\mathsf{P}(C) \neq P$ and $\pi = \emptyset$.

Furthermore, such a decision $(C, \boldsymbol{k})$ is justified by $\mathsf{M}(C)$. Therefore, if one of those moves were to be deleted from the assumptions or decisions of P at some point, the function Forget would delete $(C, \boldsymbol{k})$ as well. □

For the next results we make use of powerset games suitably modified for a set of assumptions for a player. For a set S of decisions or assumptions we denote by $\mathsf{C}(S)$ its first projection, that is, the set of positions appearing as first component in the elements of S.

Definition D.5 *(game with assumptions).* Given a powerset game G and a player P, the corresponding *game with assumptions* Γ_P is a parity game $G(\Gamma_P)$ obtained from G where for all $C \in Pos$, if $C \in \mathsf{C}(\Gamma_P)$, then $\mathsf{P}(C) = \overline{P}$ and $\mathsf{M}(C) = \emptyset$, otherwise they are the same as in G.

Notice that when the set of assumptions is empty $\Gamma_P = \emptyset$, the modified game is the same of the original one.

Then, we define a kind of strategies based on decisions and assumptions for a player, which fit the modified games above. Such strategies are history-free partial strategies. Indeed they only prescribe moves from decisions.

Definition D.6 *(strategy with assumptions).* Let G be a powerset game. Given a player P, a *strategy with assumptions* Γ_P *from decisions* Δ_P for P is a function $s_P : \mathsf{C}(\Delta_P \cup \Gamma_P) \to \mathbf{2}^{\mathsf{C}(\Delta_P \cup \Gamma_P)}$ where for all $C \in \mathsf{C}(\Gamma_P)$, $s_P(C) = \emptyset$, and for all $C \in \mathsf{C}(\Delta_P) \smallsetminus \mathsf{C}(\Gamma_P)$, $s_P(C)$ is the set of positions, possibly empty, justifying the decision $(C, \min_{\leq_P}\{\boldsymbol{k} \mid (C,\boldsymbol{k}) \in \Delta_P\})$. Given a position $C \in \mathsf{C}(\Delta_P)$, we denote by $d_P(C) = \min_{\leq_P}\{\boldsymbol{k} \mid (C,\boldsymbol{k}) \in \Delta_P\}$ the counter that was associated with C.

We say that the strategy s_P is *winning* when it is winning in the modified game $G(\Gamma_P)$, that is, every play in $G(\Gamma_P)$ following s_P starting from a position in $\mathsf{C}(\Delta_P)$ is won by player P.

The definition above is well given since by Lemmas D.3 and D.4 we know that when we add a new decision justified by some other, those are already included in the decisions or assumptions for the same player. Moreover, notice that the minimum of $\{\boldsymbol{k} \mid (C,\boldsymbol{k}) \in \Delta_P\}$ is guaranteed to be in the set itself because $\leq_P$ is a total order and the set is always finite and never empty since $C \in \mathsf{C}(\Delta_P)$.

In the modified game $G(\Gamma_P)$, given the strategy s_P with assumptions Γ_P from decisions Δ_P, for each position $C \in \mathsf{C}(\Delta_P)$ we can build a tree including all the plays starting from C where player P follows the strategy s_P.

Definition D.7 *(tree of plays).* Let G be a powerset game. Given a player P and the strategy s_P with assumptions Γ_P from decisions Δ_P, for each position $C \in \mathsf{C}(\Delta_P)$, the *tree of the plays following s_P starting from C* is the tree $\tau^C_{s_P}$ rooted in C, where every node C' in it has successors $s_P(C')$.

Such trees can contain both finite and infinite paths. Finite complete paths terminate in assumptions or truths, infinite ones contain only decisions. By construction and definition of strategy with assumptions every node is either a decision or an assumption for P. More precisely, every inner node is a position in $\mathsf{C}(\Delta_P)$, and every leaf corresponds to either a truth in Δ_P or an assumption in Γ_P. It is easy to see that a tree $\tau_{s_P}^C$ includes all the possible plays from C following s_P since the successors of inner nodes owned by the opponent are all the possible moves from those positions (decisions controlled by the opponent are justified by all the possible opponent's moves, Lemma D.4).

The trees defined above are all we need to show that a strategy with assumptions is winning. Indeed, it is enough to show that every complete path in each of those trees corresponds to a play won by the player. To this end, first we observe some key properties of the paths in the trees.

Lemma D.8 *(priorities in strategy paths). Given a powerset game, whenever functions* $\textsc{Explore}(\cdot,\cdot,\cdot,\Gamma,\Delta)$ *and* $\textsc{Backtrack}(\cdot,\cdot,\cdot,\Gamma,\Delta)$ *are invoked, for every player P, given the strategy s_P with assumptions Γ_P from decisions Δ_P, for all $\hat{C} \in \mathsf{C}(\Delta_P)$, the tree of plays $\tau_{s_P}^{\hat{C}}$ satisfies the following properties*

1. *for every pair of inner nodes C, C' in $\tau_{s_P}^{\hat{C}}$ such that C' is a successor of C, it holds $d_P(C') \leq_P \mathit{next}(d_P(C), \mathsf{i}(C))$*
2. *for every non-empty inner path $C_1, \ldots, C_n$ in $\tau_{s_P}^{\hat{C}}$, if $d_P(C_1) <_P \mathit{next}(d_P(C_n), \mathsf{i}(C_n))$, then $P = \exists$ iff $\eta_h = \nu$, where h is the highest priority occurring along the path.*

Proof. We prove the two properties separately.

1. Observe that we must have $C' \in s_P(C)$ by definition of $\tau_{s_P}^{\hat{C}}$. This means that there exists a decision $(C, d_P(C)) \in \Delta_P$ justified by the position C'. Then $(C, d_P(C))$ must have been added by a call to $\textsc{Backtrack}$. By inspecting the code it is easy to see that we were backtracking either after adding a new decision $(C', \mathit{next}(d_P(C), \mathsf{i}(C)))$ or because there was already a decision $(C', \boldsymbol{k}')$ such that $\boldsymbol{k}' \leq_P \mathit{next}(d_P(C), \mathsf{i}(C))$. Since $d_P(C') = \min_{\leq_P}\{\boldsymbol{k} \mid (C', \boldsymbol{k}) \in \Delta_P\}$, in both cases we can immediately conclude that $d_P(C') \leq_P \mathit{next}(d_P(C), \mathsf{i}(C))$.
2. We assume that $d_P(C_1) <_P \mathit{next}(d_P(C_n), \mathsf{i}(C_n))$ and $P = \exists$, and we prove that $\eta_h = \nu$, where h is the highest priority occurring along the path. A dual reasoning holds for $P = \forall$. Let next^j be a function that computes the counter after a subsequence of positions $C_1, \ldots, C_j$ in the path $C_1, \ldots, C_n$, for $j \in \underline{n}$. The function is inductively defined by $\mathit{next}^j(\boldsymbol{k}) = \mathit{next}(\mathit{next}^{j-1}(\boldsymbol{k}), \mathsf{i}(C_j))$ for all $j \in \underline{n}$, and $\mathit{next}^0(\boldsymbol{k}) = \boldsymbol{k}$. The inductive computation just repeatedly applies the function next for each position encountered along the sequence starting from a given counter $\boldsymbol{k}$. We observe that the function satisfies the property $d_\exists(C_j) \leq_\exists \mathit{next}^{j-1}(d_\exists(C_1))$ for all $j \in \underline{n}$. We show this by induction on j. Clearly it holds for $j = 1$, since by definition $\mathit{next}^0(d_\exists(C_1)) = d_\exists(C_1)$. Then, assuming it holds for j, we prove it for $j+1$. Since we know that next is monotone with respect to the input counter, by inductive hypothesis we obtain that $\mathit{next}(d_\exists(C_j), \mathsf{i}(C_j)) \leq_P \mathit{next}(\mathit{next}^{j-1}(d_\exists(C_1)), \mathsf{i}(C_j)) = \mathit{next}^j(d_\exists(C_1))$, where the last equality holds by definition of next^j. Furthermore, we know that $d_\exists(C_{j+1}) \leq_\exists \mathit{next}(d_\exists(C_j), \mathsf{i}(C_j))$ by (a) above, since C_{j+1} is a successor of C_j. And so we can immediately deduce that indeed $d_\exists(C_{j+1}) \leq_\exists \mathit{next}^j(d_\exists(C_1))$. From this and the initial assumptions we have that $d_\exists(C_1) <_\exists \mathit{next}(d_\exists(C_n), \mathsf{i}(C_n)) \leq_\exists \mathit{next}^n(d_\exists(C_1))$, where the last inequality holds by definition of next^n and monotonicity of next. Observe that since next^n just recursively applies the function next on the positions $C_1, \ldots, C_n$, the final result and the initial counter $d_\exists(C_1)$ can only differ on priorities among those of the positions $C_1, \ldots, C_n$ and lower ones (which could have been zeroed). Therefore, the highest priority on which $d_\exists(C_1)$ and $\mathit{next}^n(d_\exists(C_1))$ do not coincide must be the highest priority h appearing along the path. Furthermore, we must have $d_\exists(C_1)_h < \mathit{next}^n(d_\exists(C_1))_h$, because values can only increase or become zero, when a higher priority is encountered (and its value increased), but this would contradict the fact that h is the highest. Now we can easily conclude since by hypothesis $d_\exists(C_1) <_\exists \mathit{next}^n(d_\exists(C_1))$, and so by definition of the order $<_\exists$ we must have that $\eta_h = \nu$. □

We observe that winning strategies with assumptions are preserved by a sound function $\textsc{Forget}$ after removing an assumption and the related decisions.

Lemma D.9 *(strategies and forget). Given a powerset game, whenever* $\textsc{Forget}(\Delta_P, \Gamma_P, (C, \boldsymbol{k}))$ *is invoked, returning Δ'_P, if the strategy with assumptions Γ_P from decisions Δ_P is winning in the modified game with assumptions Γ_P, then the strategy with assumptions $\Gamma_P \smallsetminus \{(C, \boldsymbol{k})\}$ from decisions Δ'_P is winning in the modified game with assumptions $\Gamma_P \smallsetminus \{(C, \boldsymbol{k})\}$.*

Proof. It follows immediately from Definitions D.1 and D.6. □

Lemma D.10 *(winning strategy from decisions). Given a powerset game, whenever functions* $\textsc{Explore}(\cdot,\cdot,\cdot,\Gamma,\Delta)$ *and* $\textsc{Backtrack}(\cdot,\cdot,\cdot,\Gamma,\Delta)$ *are invoked, for every player P, the strategy with assumptions Γ_P from decisions Δ_P is winning in the modified game with assumptions Γ_P.*

Proof. We prove this by induction on the sequence of functions calls. Initially, on the first call $\textsc{Explore}(C,\ \mathbf{0},\ [],\ (\emptyset,\emptyset),\ (\emptyset,\emptyset))$, the property vacuously holds since $\Delta_\exists = \Delta_\forall = \emptyset$. Now, assuming that the property holds when a function is called, we show that it holds also on every invocation performed by such function.

Assume that the property holds when $\textsc{Explore}(C,\ \boldsymbol{k},\ \rho,\ \Gamma,\ \Delta)$ is called. The only invocation where the property could possibly fail is $\textsc{Backtrack}(\overline{\mathsf{P}(C)},\ C,\ \rho,\ \Gamma,\ \Delta)$ after $(C, \boldsymbol{k})$ has been added to the decisions for $\overline{\mathsf{P}(C)}$, when $\mathsf{M}(C) = \emptyset$. However we can immediately see that $\overline{\mathsf{P}(C)}$ wins from C since the opponent $\mathsf{P}(C)$ cannot move (the strategy is always winning from C). On all the other calls the property is preserved since all decisions are unchanged and no assumption has been removed.

Assume that the property holds when $\textsc{Backtrack}(P,\ C,\ \rho,\ \Gamma,\ \Delta)$ is called. There are only two invocations to check. Clearly the property is preserved on the first one, i.e., $\textsc{Explore}(C'',\ \boldsymbol{k}'',\ \rho,\ \Gamma,\ \Delta)$, since all decisions and assumptions are unchanged. The second case is instead more complex. This is when the function $\textsc{Backtrack}(P,\ C',\ t,\ \Gamma,\ \Delta)$ is invoked. Let us analyse the strategy for one player at a time. First, consider the opponent $\overline{P}$. Even though the assumption $(C', \boldsymbol{k}')$ might have been removed from $\Gamma_{\overline{P}}$, all decisions in $\Delta_{\overline{P}}$ depending on such assumption have been removed as well via the function $\textsc{Forget}(\Delta_{\overline{P}},\ \Gamma_{\overline{P}},\ (C', \boldsymbol{k}'))$. Let $\Delta'_{\overline{P}}$ be the remaining decisions. By Lemma D.9 we know that the strategy with assumptions $\Gamma_{\overline{P}} \smallsetminus \{(C', \boldsymbol{k}')\}$ from decisions $\Delta'_{\overline{P}}$ is winning as long as the strategy with assumptions $\Gamma_{\overline{P}}$ from decisions $\Delta_{\overline{P}}$ was winning. Then by inductive hypothesis the property still holds for $\overline{P}$. Now we need to prove the property for player P as well. That is, the strategy s_P with assumptions $\Gamma_P \smallsetminus \{(C', \boldsymbol{k}')\}$ from decisions $\Delta_P \cup \{(C', \boldsymbol{k}')\}$ is winning in the modified game with assumptions $\Gamma_P \smallsetminus \{(C', \boldsymbol{k}')\}$. To do this we just need to show that for every position $\hat{C} \in \mathsf{C}(\Delta_P \cup \{(C', \boldsymbol{k}')\})$, every complete path in the tree of plays $\tau_{s_P}^{\hat{C}}$ is a play won by P. First, recall that every finite complete path in $\tau_{s_P}^{\hat{C}}$ terminates in a position of an assumption or a truth. In both cases such a finite play is always won by P since in the modified game assumptions and truths correspond to positions owned by the opponent with no available moves. By inductive hypothesis we know that the strategy s'_P with assumptions Γ_P from decisions Δ_P was winning in the modified game with assumptions Γ_P. Notice that the two strategies can only differ on the position C' of the new decision $(C', \boldsymbol{k}')$. It may be that s'_P was not defined on C', if there was no decision or assumption for such position before now. Anyway, this means that if C' never occurs along the path, then the play must be won by P since s_P and s'_P coincide on all the positions in the path and s'_P was winning by inductive hypothesis. Therefore we just need to check those paths containing C'. If C' appears just finitely many times along the path, consider the subpath starting from the successor C'' of the last occurrence of C'. Such subpath does not contain C' and it is still infinite. Recalling that all positions in infinite paths must come from decisions and $C'' \neq C'$, then the subpath must be one of the complete paths in the tree of plays $\tau_{s'_P}^{C''}$. Thus, by inductive hypothesis the subpath, as well as the initial one, must be a play won by P. Otherwise, C' appears infinitely many times along the path. Consider every subpath between two consecutive occurrences of C', including only the first one. In such subpath let $C'' \neq C'$ be the last position, which is the predecessor of the second occurrence of C'. Observe that no decision $(C', \boldsymbol{k})$ could have been added after exploring $(C', \boldsymbol{k}')$ and before now, because we would necessarily have either $\boldsymbol{k} <_P \boldsymbol{k}'$ or $\boldsymbol{k} <_{\overline{P}} \boldsymbol{k}'$, thus satisfying the condition of the third **if** branch of function $\textsc{Explore}$, in which case the exploration would have stopped and $(C', \boldsymbol{k})$ would have never been added as a decision. Furthermore, any decision $(C', \boldsymbol{k})$ added before exploring $(C', \boldsymbol{k}')$ must be such that $\boldsymbol{k}' < \boldsymbol{k}$, because otherwise the exploration would have stopped satisfying the second **if** branch of function $\textsc{Explore}$ and $(C', \boldsymbol{k}')$ would have never been added as a decision. Therefore we must have $d_P(C') = \boldsymbol{k}'$ and, if $C' \in \mathsf{C}(\Delta_P) \smallsetminus \mathsf{C}(\Gamma_P)$ hence s'_P is defined on C', $d_P(C') <_P d'_P(C')$ since $d'_P(C')$ is the minimum $\boldsymbol{k}$ among the decisions for C' added before $(C', \boldsymbol{k}')$. Moreover, in the latter case, by Lemma D.8(a) we obtain that $d_P(C') <_P d'_P(C') \leq_P \mathit{next}(d_P(C''), \mathsf{i}(C''))$ since C' succeeds C''. If instead $C' \notin \mathsf{C}(\Delta_P) \smallsetminus \mathsf{C}(\Gamma_P)$, then we must have that $(C', \boldsymbol{k}') \in \Gamma_P$, since $C' \in s_P(C'') = s'_P(C'') \subseteq \mathsf{C}(\Delta_P \cup \Gamma_P)$ and $C' \in \mathsf{C}(\Delta_P \cup \{(C', \boldsymbol{k}')\}) \smallsetminus \mathsf{C}(\Gamma_P \smallsetminus \{(C', \boldsymbol{k}')\})$ because $s_P(C') \neq \emptyset$. In fact, by inspecting the code it can be seen that C' must have been added as an assumption after exploring C'', which then became a decision $(C'', d_P(C''))$, and it must have held $\boldsymbol{k}' <_P \mathit{next}(d_P(C''), \mathsf{i}(C''))$ as required by the third **if** branch in the function $\textsc{Explore}$. Thus, in both cases we have $\boldsymbol{k}' = d_P(C') <_P \mathit{next}(d_P(C''), \mathsf{i}(C''))$. And so by Lemma D.8(b) we know that $P = \exists$ iff $\eta_h = \nu$, where h is the highest priority appearing along the subpath. For now assume $P = \exists$. Since this holds for all subpaths between two consecutive occurrences of C', and there are infinitely many of them, which sequenced form the initial infinite path, then there must exist a priority h such that $\eta_h = \nu$ and it is the highest priority appearing infinitely many times along the complete path. A dual reasoning holds for $P = \forall$. Recalling that an infinite play is won by player $\exists$ (resp. $\forall$) if the highest priority $h \in \underline{m}$ appearing infinitely often is such that $\eta_h = \nu$ (resp. μ), we deduce that the path is won by P, whoever P is. And so we conclude that s_P is indeed winning in the modified game with assumptions $\Gamma_P \smallsetminus \{(C', \boldsymbol{k}')\}$. $\square$

Now we can finally present the correctness result.

Theorem 7.3 *(correctness). Given a powerset game, if a call* $\textsc{Explore}(C,\ \mathbf{0},\ [],\ (\emptyset,\emptyset),\ (\emptyset,\emptyset))$ *returns a player* P*, then* P *wins the game from* C*.*

Proof. Assume that the call $\textsc{Explore}(C,\ \mathbf{0},\ [],\ (\emptyset,\emptyset),\ (\emptyset,\emptyset))$ returns some player P. Since the only instruction returning a value is in the function $\textsc{Backtrack}$ and it is reached only when $\rho = []$, then $\textsc{Backtrack}(P,\ C',\ [],\ \Gamma,\ \Delta)$ must have been invoked for some Γ and Δ. Furthermore, $C' = C$ because $\rho = []$ is the list of positions from the initial one C to the current node C'. Also, by Lemma D.2 we have that $\Gamma_P = \emptyset$. Thus, by Lemma D.3 we have that $(C, \boldsymbol{k}) \in \Delta_P$ for some counter $\boldsymbol{k}$.

And so by Lemma D.10 we can immediately conclude that P wins the game from C, since the modified game with no assumptions coincides with the original one. □

Theorem 7.5 *(correctness with up-to). Let E be a system of m equations of the kind $\boldsymbol{x} =_{\eta} \boldsymbol{f}(\boldsymbol{x})$ over a complete lattice L. Let $\boldsymbol{u}$ be a compatible m-tuple of up-to functions for E. Then the up-to algorithm associated with the system $d(E, \boldsymbol{u})$ as given in Definition 7.4 is correct, i.e., if a call* Explore$(C, \mathbf{0}, [], (\emptyset, \emptyset), (\emptyset, \emptyset))$ *returns a player P, then P wins the game from C.*

Proof. Let G be the powerset game associated with the initial system E, G_u be the one associated with the modified system $d(E, \boldsymbol{u})$, and G'_u be the game obtained from G_u by restricting the moves of player $\exists$ from positions associated with variables y_i to only those satisfying either condition (1) or (2). Observe that the moves from every position controlled by player $\exists$ of G are included in the moves from the corresponding position in G'_u since they satisfy condition (1), since in E there are no up-to functions. Therefore, every winning strategy for $\exists$ in G can be easily converted into a winning strategy for the same player in G'_u. So the winning positions of player $\exists$ in G are necessarily included in those of G'_u. Furthermore, the same clearly happens between G'_u and G_u since the moves of $\exists$ in G'_u are defined as a restriction of those in G_u. Then, calling $W_\exists(G)$ the set of winning positions of player $\exists$ in the corresponding G, we have that $W_\exists(G) \subseteq W_\exists(G'_u) \subseteq W_\exists(G_u) = W_\exists(G)$, where the last equality holds by Theorem 5.11. Since in our case every position not winning for $\exists$ is necessarily winning for $\forall$, this means that even if we restrict certain moves of player $\exists$, thus playing in the game G'_u, we still have the same exact winning positions for both players. □

Data availability

No data was used for the research described in the article.

References

[1] Samson Abramsky, Achim Jung, Domain theory, in: Samson Abramsky, Dov Gabbay, Thomas Stephen Edward Maibaum (Eds.), Handbook of Logic in Computer Science, Oxford University Press, 1994, pp. 1–168.
[2] André Arnold, Damian Niwinski, Pawel Parys, A quasi-polynomial black-box algorithm for fixed point evaluation, in: Proc. of CSL'21, in: LIPIcs, vol. 183, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 9:1–9:23.
[3] Paolo Baldan, Barbara König, Christina Mika-Michalski, Tommaso Padoan, Fixpoint games on continuous lattices, in: Stephanie Weirich (Ed.), Proc. of POPL'19, vol. 3, ACM, 2019, pp. 26:1–26:29.
[4] Paolo Baldan, Barbara König, Tommaso Padoan, Abstraction, up-to techniques and games for systems of fixpoint equations, in: Proc. of CONCUR'20, in: LIPIcs, vol. 171, Schloss Dagstuhl – Leibniz Center for Informatics, 2020, pp. 25:1–25:20.
[5] Gourinath Banda, John P. Gallagher, Constraint-based abstract semantics for temporal logic: a direct approach to design and implementation, in: Edmund M. Clarke, Andrei Voronkov (Eds.), LPAR 2010, in: Lecture Notes in Computer Science, vol. 6355, Springer, 2010, pp. 27–45.
[6] Andrea Bianco, Luca de Alfaro, Model checking of probabalistic and nondeterministic systems, in: Proc. of FSTTCS'95, in: Lecture Notes in Computer Science, vol. 1026, Springer, 1995, pp. 499–513.
[7] Filippo Bonchi, Pierre Ganty, Roberto Giacobazzi, Dusko Pavlovic, Sound up-to techniques and complete abstract domains, in: Proc. of LICS'18, ACM, 2018, pp. 175–184.
[8] Filippo Bonchi, Barbara König, Daniela Petrişan, Up-to techniques for behavioural metrics via fibrations, in: Proc. of CONCUR'18, in: LIPIcs, vol. 118, Schloss Dagstuhl – Leibniz Center for Informatics, 2018, pp. 17:1–17:17.
[9] Filippo Bonchi, Damien Pous, Checking NFA equivalence with bisimulations up to congruence, in: Proc. of POPL'13, ACM, 2013, pp. 457–468.
[10] Julian Bradfield, Igor Walukiewicz, The mu-calculus and model checking, in: Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, Roderick Bloem (Eds.), Handbook of Model Checking, Springer, 2018, pp. 871–919.
[11] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, Frank Stephan, Deciding parity games in quasipolynomial time, in: Proc. of STOC'17, ACM, 2017, pp. 252–263.
[12] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron A. Peled, Helmut Veith, Model Checking, 2nd edition, MIT Press, 2018.
[13] Rance Cleaveland, Marion Klein, Bernhard Steffen, Faster model checking for the modal mu-calculus, in: Proc. of CAV 1992, in: Lecture Notes in Computer Science, vol. 663, Springer, 1992, pp. 410–422.
[14] Patrick Cousot, Partial completeness of abstract fixpoint checking, in: Berthe Y. Choueiry, Toby Walsh (Eds.), Proceedings of the Fourth International Symposium on Abstraction, Reformulations and Approximation, SARA'2000, in: Lecture Notes in Computer Science, vol. 1864, Springer, 26–29 July 2000, pp. 1–25.
[15] Patrick Cousot, Radhia Cousot, Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, in: Proc. of POPL'77, ACM, 1977, pp. 238–252.
[16] Patrick Cousot, Radhia Cousot, Systematic design of program analysis frameworks, in: Proc. of POPL'79, ACM, 1979, pp. 269–282.
[17] Patrick Cousot, Radhia Cousot, Temporal abstract interpretation, in: Mark N. Wegman, Thomas W. Reps (Eds.), Proc. of POPL'00, ACM, 2000, pp. 12–25.
[18] Radhia Cousot, Patrick Cousot, A constructive characterization of the lattices of all retractions, preclosure, quasi-closure and closure operators on a complete lattice, Port. Math. 38 (1–2) (1979) 185–198.
[19] Radhia Cousot, Patrick Cousot, Constructive versions of Tarski's fixed point theorems, Pac. J. Math. 82 (1) (1979) 43–57.
[20] Dennis Dams, Rob Gerth, Orna Grumberg, Abstract interpretation of reactive systems, ACM Trans. Program. Lang. Syst. 19 (2) (1997) 253–291.
[21] E. Allen Emerson, Charanjit S. Jutla, Tree automata, mu-calculus and determinacy, in: Proc. of SFCS'91, IEEE, 1991, pp. 368–377.
[22] Alessandro Flori, A local algorithm for systems of fixpoint equations: Study and implementation, Master's thesis, University of Padua, 2023.
[23] Oliver Friedmann, Martin Lange, The pgsolver collection of parity game solvers, Technical report, University of Munich, 2009.
[24] Oliver Friedmann, Martin Lange, Solving parity games in practice, in: Zhiming Liu, Anders P. Ravn (Eds.), Automated Technology for Verification and Analysis, Springer Berlin Heidelberg, 2009, pp. 182–196.
[25] Oliver Friedmann, Martin Lange, Two local strategy iteration schemes for parity game solving, Int. J. Found. Comput. Sci. 23 (03) (2012) 669–685.
[26] Roberto Giacobazzi, Francesco Ranzato, Francesca Scozzari, Making abstract interpretations complete, J. ACM 47 (2) (2000) 361–416.
[27] Patrice Godefroid, Michael Huth, Radha Jagadeesan, Abstraction-based model checking using modal transition systems, in: Proc. of CONCUR'01, in: Lecture Notes in Computer Science, vol. 2154, Springer, 2001, pp. 426–440.

[28] Jan Friso Groote, Tim A.C. Willemse, Parameterised Boolean equation systems, Theor. Comput. Sci. 343 (2005) 332–369.
[29] Orna Grumberg, Martin Lange, Martin Leucker, Sharon Shoham, When not losing is better than winning: abstraction and refinement for the full mu-calculus, Inf. Comput. 205 (8) (2007) 1130–1148.
[30] Ichiro Hasuo, Shunsuke Shimizu, Corina Cîrstea, Lattice-theoretic progress measures and coalgebraic model checking, in: Proc. of POPL'16, ACM, 2016, pp. 718–732.
[31] D. Hausmann, L. Schröder, Quasipolynomial computation of nested fixpoints, in: Proc. of TACAS'21, in: Lecture Notes in Computer Science, vol. 12651, Springer, 2021, pp. 38–56.
[32] Daniel Hirschkoff, Automatically proving up to bisimulation, in: Proc. of MFCS'98 Workshop on Concurrency, in: Electronic Notes in Theoretical Computer Science, vol. 18, Elsevier, 1998, pp. 75–89.
[33] Daniel Hirschkoff, Mise en oeuvre de preuves de bisimulation, PhD thesis, Ecole Nationale des Ponts et Chaussées, 1999.
[34] Chung-Kil Hur, Georg Neis, Derek Dreyer, Viktor Vafeiadis, The power of parameterization in coinductive proof, in: Proc. of POPL'13, ACM, 2013, pp. 193–206.
[35] Michael Huth, Marta Kwiatkowska, Quantitative analysis and model checking, in: Proc. of LICS'97, IEEE, 1997, pp. 111–122.
[36] Marcin Jurdziński, Small progress measures for solving parity games, in: Proc. of STACS'00, in: Lecture Notes in Computer Science, vol. 1770, Springer, 2000, pp. 290–301.
[37] Marcin Jurdzinski, Ranko Lazic, Succinct progress measures for solving parity games, in: Proc. of LICS'17, ACM/IEEE, 2017, pp. 1–9.
[38] Dexter Kozen, Results on the propositional μ-calculus, Theor. Comput. Sci. 27 (3) (1983) 333–354.
[39] Kim Guldstrand Larsen, Bent Thomsen, A modal process logic, in: Proc. of LICS'88, IEEE, 1988, pp. 203–210.
[40] Maurice Laveaux, Wieger Wesselink, Tim A.C. Willemse, On-the-fly solving for symbolic parity games, in: Dana Fisman, Grigore Rosu (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2022, pp. 137–155.
[41] Karoliina Lehtinen, A modal μ perspective on solving parity games in quasi-polynomial time, in: Proc. of LICS'18, ACM/IEEE, 2018, pp. 639–648.
[42] Claire Loiseaux, Susanne Graf, Joseph Sifakis, Ahmed Bouajjani, Saddek Bensalem, Property preserving abstractions for the verification of concurrent systems, Form. Methods Syst. Des. 6 (1995) 1–35.
[43] Angelika Mader, Verification of Modal Properties Using Boolean Equation Systems, PhD thesis, TU München, 1997.
[44] Annabelle McIver, Carroll Morgan, Results on the quantitative μ-calculus qmμ, ACM Trans. Comput. Log. 1 (2007) 3.
[45] Robin Milner, Communication and Concurrency, Prentice Hall, 1989.
[46] Antoine Miné, Tutorial on static inference of numeric invariants by abstract interpretation, Found. Trends Program. Lang. 4 (3–4) (2017) 120–372.
[47] Matteo Mio, Alex Simpson, Łukasiewicz μ-calculus, Fundam. Inform. 150 (3–4) (2017) 317–346.
[48] Damien Pous, Complete lattices and up-to techniques, in: Proc. of APLAS'07, in: LNCS, vol. 4807, Springer, 2007, pp. 351–366.
[49] Damien Pous, Coinduction all the way up, in: Proc. of LICS'16, ACM, 2016, pp. 307–316.
[50] Damien Pous, Davide Sangiorgi, Enhancements of the bisimulation proof method, in: Davide Sangiorgi, Jan Rutten (Eds.), Advanced Topics in Bisimulation and Coinduction, Cambridge University Press, 2011.
[51] Davide Sangiorgi, Introduction to Bisimulation and Coinduction, Cambridge University Press, 2011.
[52] Davide Sangiorgi, Robin Milner, The problem of "weak bisimulation up to", in: W.R. Cleaveland (Ed.), Proc. of CONCUR'92, Springer, 1992, pp. 32–46.
[53] David A. Schmidt, Binary relations for abstraction and refinement, in: Workshop on Refinement and Abstraction, Elsevier Electronic, 2000.
[54] Dana Scott, Continuous lattices, in: F.W. Lawvere (Ed.), Toposes, Algebraic Geometry and Logic, in: Lecture Notes in Mathematics, Springer, 1972, pp. 97–136.
[55] Helmut Seidl, Fast and simple nested fixpoints, Inf. Process. Lett. 59 (6) (1996) 303–308.
[56] David Sprunger, Lawrence S. Moss, Precongruences and parametrized coinduction for logics for behavioral equivalence, in: Proc. of CALCO'17, in: LIPIcs, vol. 72, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017, pp. 23:1–23:15.
[57] Perdita Stevens, Colin Stirling, Practical model-checking using games, in: Proc. of TACAS'98, in: Lecture Notes in Computer Science, vol. 1384, Springer, 1998, pp. 85–101.
[58] Colin Stirling, Local model checking games, in: Proc. of CONCUR'95, in: Lecture Notes in Computer Science, vol. 962, Springer, 1995, pp. 1–11.
[59] Alfred Tarski, A lattice-theoretical fixpoint theorem and its applications, Pac. J. Math. 5 (1955) 285–309.
[60] Jens Vöge, Marcin Jurdziński, A discrete strategy improvement algorithm for solving parity games, in: E. Allen Emerson, Aravinda Prasad Sistla (Eds.), Computer Aided Verification, Springer Berlin Heidelberg, 2000, pp. 202–215.
[61] Wieslaw Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Theor. Comput. Sci. 200 (1–2) (1998) 135–183.