The Complexity of Independence-Friendly Fixpoint Logic

Julian Bradfield[1] and Stephan Kreutzer[2]

[1] Laboratory for Foundations of Computer Science, University of Edinburgh, jcb@inf.ed.ac.uk

[2] Institut für Informatik, Humboldt Universität, 10099 Berlin, Germany, kreutzer@informatik.hu-berlin.de

Abstract. We study the complexity of model-checking for the fixpoint extension of Hintikka and Sandu's independence-friendly logic. We show that this logic captures ExpTime; and by embedding PFP, we show that its combined complexity is ExpSpace-hard, and moreover the logic includes second order logic (on finite structures).

1 Introduction

In everyday life we often have to make choices in ignorance of the choices made by others that might have affected our choice. With the popularity of the agent paradigm, there is much theoretical and practical work on logics of knowledge and belief in which such factors can be explicitly expressed in designing multi-agent systems. However, ignorance is not the only reason for making independent choices: in mathematical writing, it is not uncommon to assert the existence of a value for some parameter uniformly in some earlier mentioned parameter.

Hintikka and Sandu [HiS96] introduced a logic, called Independence-friendly (IF) logic, in which such independent choices can be formalized by independent quantification. Some of the ideas go back some decades, for IF logic can also be viewed as an alternative account of branching quantifiers (Henkin quantifiers) in terms of games of imperfect information. Independent quantification is a subtle concept, with many pitfalls for the unwary. It is also quite powerful: it has long been known that it has existential second-order power. In previous work [BrF02], the first author and Fröschle applied the idea of independent quantification to modal logics, where it has natural links with the theory of true concurrency; this prompted some consideration of fixpoint versions of IF modal logics, since adding fixpoint operators is the easiest way to get a powerful temporal logic from a simple modal logic. This led the first author to an initial investigation [Bra03] of the fixpoint extension of first-order IF logic, which we call IF-LFP . It turned out that fixpoint IF logic is not trivial to define, and appears to be very expressive, with the interaction between fixpoints and independent quantification giving a dramatic increase in expressive power. In [Bra03], only some fairly simple complexity results were obtained; in this paper, we obtain much stronger results about the model-checking complexity of IF-LFP . For the data complexity, we show that not only is IF-LFP ExpTime-complete, but it captures ExpTime; and for the combined complexity, we obtain an ExpSpacehardness result. This latter result is obtained by an embedding of partial fixpoint logic into IF-LFP, which shows that on finite structures IF-LFP even includes second-order logic, a much stronger result than the first author previously conjectured.

2 Independence-Friendly Fixpoint Logic

2.1 Syntax

First of all, we state one important **notational convention**: to minimize the number of parentheses, we take the scope of all quantifiers and fixpoint operators to extend as far to the right as possible.

Now we define the syntax of first-order IF logic. Here we use the version of Hodges [Hod97], and we confine the 'independence-friendly' operators to the quantifiers; in the full logic, one can also specify conjunctions and disjunctions that are independent, but these are not necessary for our purposes – their addition changes none of our results.

Definition 2.1. As for FOL, IF-FOL has proposition (P, Q etc.), relation (R, S etc.), function (f, g etc.) and constant (a, b etc.) symbols, with given arities. It also has individual *variables* v, x etc. We write $\boldsymbol{x}, \boldsymbol{v}$ etc. for tuples of variables, and similarly for tuples of other objects; we use concatenation of symbols to denote concatenation of tuples with tuples or objects.

For formulae φ and terms t, the (meta-level) notations $\varphi[\boldsymbol{x}]$ and $t[\boldsymbol{x}]$ mean that the free variables of φ or t are included in the variables $\boldsymbol{x}$, without repetition.

The notions of 'term' and 'free variable' are as for FOL.

We assume equality $=$ is in the language, and atomic formulae are defined as usual by applying proposition or relation symbols to individual terms or tuples of terms. The free variables of the formula $R(\boldsymbol{t})$ are then those of $\boldsymbol{t}$.

The compound formulae are given as follows:

Conjunction and disjunction. If $\varphi[\boldsymbol{x}]$ and $\psi[\boldsymbol{y}]$ are formulae, then $(\varphi \vee \psi)[\boldsymbol{z}]$ and $(\varphi \wedge \psi)[\boldsymbol{z}]$ are formulae, where $\boldsymbol{z}$ is the union of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Quantifiers. If $\varphi[\boldsymbol{y}, x]$ is a formula, x a variable, and W a finite set of variables, then $(\forall x/W.\, \varphi)[\boldsymbol{y}]$ and $(\exists x/W.\, \varphi)[\boldsymbol{y}]$ are formulae. If W is empty, we write just $\forall x.\, \varphi$ and $\exists x.\, \varphi$.

Game negation. If $\varphi[\boldsymbol{x}]$ is a formula, so is $({\sim}\varphi)[\boldsymbol{x}]$.

Flattening. If $\varphi[\boldsymbol{x}]$ is a formula, so is $(\downarrow \varphi)[\boldsymbol{x}]$.

Negation. $\neg\varphi$ is an abbreviation for $\sim \downarrow \varphi$.

Definition 2.2. IF-FOL$^+$ is the logic in which $\sim$, $\downarrow$ and $\neg$ are applied only to atomic formulae.

In the rest of this paper, we shall be working with IF-FOL$^+$, in which $\sim$ and $\neg$ merge, and $\downarrow$ has no effect. We shall therefore omit $\sim$ and $\downarrow$ from future definitions, and take $\neg$ as primitive, which also allows a simpler semantics than that for the full IF-FOL. Since we are proving lower bounds, the results apply also to the logic with negations.

2.2 Traditional Semantics

In the independent quantifiers the intention is that W is the set of independent variables, whose values the player is not allowed to know at this choice point: thus the classical Henkin quantifier $\begin{smallmatrix}\forall x & \exists y\\ \forall u & \exists v\end{smallmatrix}$, where x and y are independent of u and v, can be written as $\forall x/\varnothing.\, \exists y/\varnothing.\, \forall u/\{x,y\}.\, \exists v/\{x,y\}$. This notion of independence is the reason for saying that IF logic is natural in mathematical English: statements such as "For every x, and for all $\epsilon > 0$, there exists δ, depending only on ϵ ..." can be transparently written as $\forall x, \epsilon > 0.\, \exists \delta/x.\, \ldots$ in IF logic.

If one then plays the Hintikka evaluation game (otherwise known as the model-checking game) with this additional condition, which can be formalized by requiring strategies to be uniform in the 'unknown' variables, one gets a game semantics of imperfect information, and defines a formula to be true iff Eloise has a winning strategy.

These games are not determined, so it is *not* the case that Abelard has a winning strategy iff the formula is not true. For example, $\begin{smallmatrix}\forall x\\ \exists y\end{smallmatrix}.x = y$ (or $\forall x.\, \exists y/\{x\}.\, x = y$) is untrue in any structure with more than one element, but Abelard has no winning strategy.

An alternative interpretation of the logic, dating from the early work on branching quantifiers, and one that is easier to handle mathematically in straightforward cases, is via Skolem functions with limited arguments. In FOL, the first order sentence $\forall x.\, \exists y.\, x = y$, over some universe A, is converted via Skolemization to the existential second-order sentence $\exists f : A \to A.\, \forall x.\, x = f(x)$. In this procedure, the Skolem function always takes as arguments all the universal variables currently in scope. By allowing Skolem functions to take only some of the arguments, we get a similar translation of IF-FOL$^+$: for example, $\forall x.\, \exists y/\{x\}.\, x = y$ becomes $\exists f : 1 \to A.\, \forall x.\, x = f()$. It can be shown that these two semantics are equivalent, in that an IF-FOL$^+$ sentence is true in the game semantics iff its Skolemization is true.

It is also well known that IF-FOL$^+$ is equivalent to existential second-order logic (in the cases where this matters, 'second-order' here means function quantification rather than set quantification). This is because the Skolemization process can be inverted: given a Σ^1_1 sentence, it can be turned into an IF-FOL$^+$ sentence (or equivalently, a sentence with Henkin quantifiers). We shall make use of this procedure in later results. Details can be found in [Wal70,End70] or in the full version of the paper, but here let us illustrate it by a standard example that demonstrates the power of IF logic. Consider the sentence 'there is an injective endofunction that is not surjective'. This is true only in infinite domains, and therefore not first-order expressible. It can be expressed directly in Σ^1_1 as

$$\exists f.\, (\forall x_1, x_2.\, f(x_1) = f(x_2) \Rightarrow x_1 = x_2) \wedge (\exists c.\, \forall x.\, f(x) \neq c)$$

which for the sake of reducing complexity below we will simplify to

$$\exists f.\, \exists c.\, \forall x_1, x_2.\, (f(x_1) = f(x_2) \Rightarrow x_1 = x_2) \wedge f(x_1) \neq c.$$

The basic trick for talking about functions in IF-FOL is to replace $\exists f.\, \forall x$ by $\forall x.\, \exists y$, so that y plays the role of $f(x)$. In FOL, this works only if there is just one application of f; but in IF-FOL, we can do it for two (or more) applications of f: we write $\forall x_1.\, \exists y_1$, and then we write an independent $\forall x_2/\{x_1, y_1\}.\, \exists y_2/\{x_1, y_1\}$. Now in order to make sure that these two (x_i, y_i) pairings represent the same f, the body of the translated formula is given a clause $(x_1 = x_2) \Rightarrow (y_1 = y_2)$. Applying this procedure to the Σ^1_1 sentence above and optimizing a bit, we get

$$\forall x_1, x_2.\, \exists y_1/x_2.\, \exists y_2/x_1.\, \exists c/\{x_1, x_2\}.\, (y_1 = y_2 \Leftrightarrow x_1 = x_2) \wedge y_1 \neq c.$$

2.3 Trump Semantics

The game semantics is how Hintikka and Sandu originally interpreted IF logic. Later on, the trump semantics of Hodges [Hod97], with variants by others, gave a Tarski-style semantics, equivalent to the original. This semantics is as follows:

Definition 2.3. Let a structure A be given, with constants, propositions and relations interpreted in the usual way. A *deal* $\boldsymbol{a}$ for $\varphi[\boldsymbol{x}]$ or $\boldsymbol{t}[\boldsymbol{x}]$ is an assignment of an element of A to each variable in $\boldsymbol{x}$. Given a deal $\boldsymbol{a}$ for a tuple of terms $\boldsymbol{t}[\boldsymbol{x}]$, let $\boldsymbol{t}(\boldsymbol{a})$ denote the tuple of elements obtained by evaluating the terms under the deal $\boldsymbol{a}$.

If $\varphi[\boldsymbol{x}]$ is a formula and W is a subset of the variables in $\boldsymbol{x}$, two deals $\boldsymbol{a}$ and $\boldsymbol{b}$ for φ are $\simeq_W$*-equivalent* ($\boldsymbol{a} \simeq_W \boldsymbol{b}$) iff they agree on the variables not in W. A $\simeq_W$*-set* is a non-empty set of pairwise $\simeq_W$-equivalent deals.

The denotation $[\![\varphi]\!]$ of a formula is a set T of *trumps*. If φ has n free variables, then $T \in \wp(\wp(A^n))$ – that is, a trump is a set of deals.

- If $(R(\boldsymbol{t}))[\boldsymbol{x}]$ is atomic, then a non-empty set D of deals is a trump iff $\boldsymbol{t}(\boldsymbol{a}) \in R$ for every $\boldsymbol{a} \in D$.
- D is a trump for $(\varphi \wedge \psi)[\boldsymbol{x}]$ iff D is a trump for $\varphi[\boldsymbol{x}]$ and D is a trump for $\psi[\boldsymbol{x}]$.
- D is a trump for $(\varphi \vee \psi)[\boldsymbol{x}]$ iff it is non-empty and there are trumps E of φ and F of ψ such that every deal in D belongs either to E or F.
- D is a trump for $(\forall y/W.\, \psi)[\boldsymbol{x}]$ iff the set $\{\, \boldsymbol{a}b \mid \boldsymbol{a} \in D, b \in A \,\}$ is a trump for $\psi[\boldsymbol{x}, y]$.
- D is a trump for $(\exists y/W.\, \psi)[\boldsymbol{x}]$ iff there is a trump E for $\psi[\boldsymbol{x}, y]$ such that for every $\simeq_W$-set $F \subseteq D$ there is a b such that $\{\, \boldsymbol{a}b \mid \boldsymbol{a} \in F \,\} \subseteq E$.
- D is a trump for $(\neg R(\boldsymbol{t}))[\boldsymbol{x}]$ iff $\boldsymbol{t}(\boldsymbol{a}) \notin R$ for every $\boldsymbol{a} \in D$.

A trump for φ is essentially a set of winning positions for the model-checking game for φ, for a given *uniform* strategy, that is, a strategy where choices are uniform in the 'hidden' variables. The most intricate part of the above definition is the clause for $\exists y/W.\, \psi$: it says that a trump for $\exists y/W.\, \psi$ is got by adding a witness for y, uniform in the W-variables, to trumps for ψ.

It is easy to see that any subset of a trump is a trump. In the case of an ordinary first-order $\varphi(\boldsymbol{x})$, the set of trumps of φ is just the power set of the set of tuples satisfying φ. To see how a more complex set of trumps emerges, consider the following formula, which has x free: $\exists y/\{x\}.\, x = y$. Any singleton set of deals is a trump, but no other set of deals is a trump. Thus we obtain that $\forall x.\, \exists y/\{x\}.\, x = y$ has no trumps (unless the domain has only one element).

The strangeness of the trump definitions is partly to do with some more subtle features of IF logics, that we do not here have space to discuss, but which are considered in detail in Ahti-Veikko Pietarinen's thesis [Pie00]. However, to take one good example, raised by a referee, consider $\varphi = \exists x.\, \exists y/\{x\}.\, x = y$. What are its trumps? As above, the trumps of $\exists y/\{x\}.\, x = y$ are singleton sets of deals. The only potential trump for φ is the set containing the empty deal $D = \{\langle\rangle\}$. Applying the definition, D is a trump for φ iff there is a singleton deal set $\{a\}$ for x such that there is a b such that $\{b\} \subseteq \{a\}$. The right hand side is true – take $b = a$ – so D is a trump. How come, if there is more than one element in A? Surely we must choose y independently of x, and therefore φ can't be true? Not so: because the choices are both made by the same player (Eloise), she can, as it were, make a uniform choice of y that, by 'good luck' agrees with her previous choice of x. Since she is not in the business of making herself lose, she will always do so. In game-theoretic terms, this is the difference between requiring a strategy to make uniform moves, and requiring a player to choose a strategy uniformly. In fact Hintikka and Sandu avoided this problem by only allowing the syntax to express quantifications independent in the other player's variables, which is in practice all one wishes to use in any case. Hodges removed this restriction to make his semantics cleaner, exposing the curiosity we have just described.

A sentence is said to be true if $\{\langle\rangle\} \in T$ (the empty deal is a trump set), and false if $\{\langle\rangle\} \in C$; this corresponds to Eloise or Abelard having a uniform winning strategy. Otherwise, it is undetermined. Note that 'false' is reserved for a strong sense of falsehood – undetermined sentences are also not true, and in the simple cases where negation and flattening are not employed, an undetermined sentence is as good as false.

2.4 IF-LFP

We now describe the addition of fixpoint operators to IF-FOL. This is slightly intricate, although the normal intuitions for understanding fixpoint logics still apply.

Definition 2.4. IF-LFP extends the syntax of IF-FOL as follows:

- There is a set $\mathrm{Var} = \{X, Y, \ldots\}$ of fixpoint variables. Each variable X has an arity $\mathrm{ar}(X)$.
- If X is a fixpoint variable, and $\boldsymbol{t}$ an $\mathrm{ar}(X)$-vector of terms then $X(\boldsymbol{t})$ is a formula.
- Let φ be a formula with free fixpoint variable X. φ has free individual variables $\boldsymbol{x} = \langle x_1, \ldots, x_{\mathrm{ar}(X)}\rangle$ for the elements of X, together with other free individual variables $\boldsymbol{z}$; let $\mathrm{fv}_\varphi(X)$ be the length of $\boldsymbol{z}$. Now if $\boldsymbol{t}$ is a sequence of $\mathrm{ar}(X)$ terms with free variables $\boldsymbol{y}$, then $(\mu X(\boldsymbol{x}).\varphi)(\boldsymbol{t})[\boldsymbol{z}, \boldsymbol{y}]$ is a

formula; **provided that** φ is IF-FOL$^+$. In this context, we write just $\mathrm{fv}(X)$ for $\mathrm{fv}_\varphi(X)$.
– similarly for $\nu X(\boldsymbol{x}).\varphi$.

To give the semantics of IF-LFP, we first define valuations for free fixpoint variables, in the context of some IF-LFP formula.

Definition 2.5. A fixpoint valuation $\mathscr{V}$ maps each fixpoint variable X to a set $\mathscr{V}(X) \in \wp(\wp(A^{\mathrm{ar}(X)+\mathrm{fv}(X)}))$.

Let D be a non-empty set of deals for $X(\boldsymbol{t})[\boldsymbol{x},\boldsymbol{z},\boldsymbol{y}]$, where $\boldsymbol{y}$ are the free variables of $\boldsymbol{t}$ not already among $\boldsymbol{x},\boldsymbol{z}$. A deal $d = \boldsymbol{acb} \in D$, where $\boldsymbol{a},\boldsymbol{c},\boldsymbol{b}$ are the deals for $\boldsymbol{x},\boldsymbol{z},\boldsymbol{y}$ respectively, determines a deal $d' = \boldsymbol{t}(d)\boldsymbol{c}$ for $X[\boldsymbol{x},\boldsymbol{z}]$. Let $D' = \{\, d' \mid d \in D \,\}$. D is a trump for $X(\boldsymbol{t})$ iff $D' \in \mathscr{V}(X)$.

The intuition here is that a fixpoint variable needs to carry the trumps both for the elements of the fixpoint and for any free variables, as we shall see below. Then we define a suitable complete partial order on the range of valuations, which will also be the range of denotations for formulae; it is simply the inclusion order on trump sets.

Definition 2.6. If T_1 and T_2 are elements of $\wp(\wp(A^n))$, define $T_1 \preceq T_2$ iff $T_1 \subseteq T_2$.

This order gives the standard basic lemma for fixpoint logics:

Lemma 2.7. *If $\varphi(X)[\boldsymbol{x},\boldsymbol{z}]$ is an IF-FOL$^+$ formula and $\mathscr{V}$ is a fixpoint valuation, the map on $\wp(\wp(A^{\mathrm{ar}(X)+\mathrm{fv}(X)})$ given by*

$$T \mapsto [\![\varphi]\!]_{\mathscr{V}[X:=T]}$$

is monotone with respect to $\preceq$; hence it has least and greatest fixpoints, constructible by iteration from the bottom and top elements of the set of denotations.

Thus we have the familiar definition of the μ operator:

Definition 2.8. $[\![\mu X(x).\varphi(X)[\boldsymbol{x},\boldsymbol{z}]]\!]$ is the least fixpoint of the map on $\wp(\wp(A^{\mathrm{ar}(X)+\mathrm{fv}(X)})$ given by

$$T \mapsto [\![\varphi]\!]_{\mathscr{V}[X:=T]};$$

and $[\![\nu X(x).\varphi(x)[\boldsymbol{x},\boldsymbol{z}]]\!]$ is the greatest fixpoint. $\mu^\zeta X(x).\varphi$ means the ζth approximant of $\mu X(x).\varphi$, defined recursively by $\mu^\zeta X(x).\varphi = \varphi(\bigcup_{\xi<\zeta} \mu^\xi X(x).\varphi)$.

A distinctive feature of the definition, compared to the normal LFP definition, is the way that free variables are explicitly mentioned. Normally, one can fix values for the free variables, and then compute the fixpoint, but because of independent quantification this is not possible in the IF setting. For example, consider the formula fragment

$$\forall z. \ldots \mu X(x). \ldots \vee \exists y/\{z\}. X(y)$$

The independent choice of y means that the trumps for the fixpoint depend on the possible deals for z, not just a single deal.

2.5 Examples of IF-LFP

In order to give some human-readable examples of IF-LFP, we here reproduce a section from [Bra03].

For convenience, we introduce the abbreviation $\varphi \Rightarrow \psi$ for $\psi \vee \neg\varphi$ provided that φ is atomic.

Let $G = (V, E)$ be a directed graph. The usual LFP formula $R(y, z) \stackrel{\text{def}}{=} (\mu X(x).z = x \vee \exists w.\, E(x, w) \wedge X(w))(y)$ asserts that the vertex z is reachable from y. Hence the formula $\forall y.\, \forall z.\, R(y, z)$ asserts that G is strongly connected. Now consider the IF-LFP formula

$$\forall y.\, \forall z.\, (\mu X(x).z = x \vee \exists w/\{y, z\}.\, E(x, w) \wedge X(w))(y).$$

At first sight, one might think this asserts not only that every z is reachable from every y, but that the path taken is independent of the choice of y and z. This is true exactly if G has a directed Hamiltonian cycle, a much harder property than being strongly connected.

Of course, the formula does not mean this, because the variable w is fresh each time the fixpoint is unfolded. In the trump semantics, the denotation of the fixpoint will include all the possible choice functions at each step, and hence all possible combinations of choice functions. Thus the formula reduces to strong connectivity.

It may be useful to look at the approximants of this formula in a little more detail, to get some intuitions about the trump semantics. Considering just

$$H \stackrel{\text{def}}{=} (\mu X(x).z = x \vee \exists w/\{y, z\}.\, E(x, w) \wedge X(w))[x, y, z],$$

we see that in computing each approximant, the calculation of $[\![\exists w/\{y, z\}.\, \ldots]\!]$ involves generating a trump for every possible value of a choice function $f\colon x \mapsto w$. This is a feature of the original trump semantics, and can be understood by viewing it as a second-order semantics: just as the compositional Tarskian semantics of $\exists x.\, \varphi(x)$ involves computing all the witnesses for $\varphi(x)$, so computing the trumps of $\exists x/\{y\}.\, \varphi$ involves computing all the Skolem functions; and unlike the first-order case, it is necessary to work with functions (as IF can express existential second-order logic). Consequently, the nth approximant includes all states such that $x \to f_1(x) \to f_2 f_1(x) \to \ldots \to f_n \ldots f_1(x) = z$ for any sequence of successor-choosing functions f_i. Thus we see that the cumulative effect is the same as for a normal $\exists w$, and the independent choice has indeed not bought us anything.

It is, however, possibleto produce a slightly more involved formula expressing the Hamiltonian cycle property in this inductively defined way, by using the standard trick for expressing functions in Henkin quantifier logics. We replace the formula H by

$$\begin{aligned}&\forall s.\, \exists t/\{y, z\}.\, E(s, t) \wedge (\mu X(x).x = z \vee\\ &\forall u.\, \exists v/\{x, y, z, s, t\}.\, (s = u \Rightarrow t = v) \wedge (x = u \Rightarrow X(v)))(y).\end{aligned}$$

This works because the actual function f selecting a successor for every node is made outside the fixpoint by $\forall s.\, \exists t/\{y, z\}.\, E(s,t) \wedge \ldots$; then inside the fixpoint, a new choice function g is made so that $X(g(x))$, and g is constrained to be the same as f by the clause $(s = u \Rightarrow t = v)$. (The reader who is not familiar with the IF/Henkin to existential second-order translation might wish to ponder why $\forall s.\, \exists t/\{y, z\}.\, E(s,t) \wedge \mu X(x).x = z \vee (x = s \Rightarrow X(t))$ does not work.)

3 Second-Order Inductions and Independence-Friendly Logics

It has been known from the early studies of Henkin quantifiers [Wal70,End70] that existential second-order sentences can be transformed into sentences with the Henkin quantifier, and thus into IF-FOL. A technique frequently used in our results is the translation of existential second order inductions into IF-LFP. For this we show that the translation of existential second-order logic into independence-friendly logic can be extended to a translation of positive existential second-order inductions into independence-friendly fixpoint logic. Throughout this paper we only consider finite structures. Therefore we only give the translation for finite structures here.

We first give a formal definition of positive Σ^1_1-inductive formulae.

Definition 3.1. An (n,k)-ary third-order variable $\mathscr{R}$ is a variable interpreted by a set whose members are n-tuples of k-ary functions. Let, for some $k, n < \omega$, $\mathscr{R}$ be a (n,k)-ary third-order variable. A formula $\varphi(\mathscr{R}, f_1, \ldots, f_n)$ is Σ^1_1-*inductive* if it is built up by the usual formula building rules for Σ^1_1 augmented by a rule that allows the use of atoms $\mathscr{R} f_1 \ldots f_n$, where the f_i are k-ary function symbols, provided that the variable $\mathscr{R}$ is only used positively in φ.

Σ^1_1-inductive formulae φ can be used to define least fixpoint inductions in the same way as first-order formulae with a free relation variable in which they are positive are used to define fixpoint inductions. So we can define the stages $\mathscr{R}^\alpha$, $\alpha < \omega$, of the fixpoint induction in φ which ultimately lead to the least fixpoint of the operator defined by the formula φ. We call a relation that is obtained as the least fixpoint of a Σ^1_1-inductive formula Σ^1_1-*inductive*. Note, that the Σ^1_1-inductive relations are third-order objects, i.e. sets of functions.

We show next that any Σ^1_1-inductive third-order relation $\mathscr{R}$ can be defined by an IF-LFP-formula in the sense that there is a formula $\varphi(R, \boldsymbol{x}, y)$, positive in the second-order variable R, such that the *maximal* trumps in the least fixpoint of the operator defined by φ are precisely the graphs of the functions in $\mathscr{R}$. For the sake of simplicity, we only consider the case of $(1,k)$-ary inductions, i.e. where the fixpoint is a set of functions.

An important concept used in the following proofs is the notion of *functional trumps*; and a technically useful concept is that of *maximal trumps*.

Definition 3.2. Let $\varphi(\boldsymbol{x}, y)$ be a formula. A trump T for φ is *functional in* $\boldsymbol{x}$ *and* y, if for all pairs $(\boldsymbol{a}, b), (\boldsymbol{a}', b')$ of deals in T we have $b = b'$ whenever $\boldsymbol{a} = \boldsymbol{a}'$. T is *maximal* if there is no $T' \supsetneq T$ that is a trump for φ.

Note that because any subset of a trump is a trump, the trumps of a formula are determined by its maximal trumps. Of course, any subset of a functional trump is functional.

Notation. In the following proofs we will frequently use a construction like

$$\forall \boldsymbol{x}/\{\boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_n, y_n\}\exists y/\{\boldsymbol{x}_1, y_1, \ldots, \boldsymbol{x}_n, y_n\}\big((\boldsymbol{x} = \boldsymbol{x}_1 \to y = y_1) \wedge \varphi\big)$$

for some formula φ. We will abbreviate this by

$$\forall \boldsymbol{x}\exists y \, \mathrm{clone}(\boldsymbol{x}_1, y_1; \boldsymbol{x}_2, y_2 \ldots, \boldsymbol{x}_n, y_n)\varphi.$$

and we will usually omit the list $(\boldsymbol{x}_2, y_2 \ldots, \boldsymbol{x}_n, y_n)$ of other variables which appear in the independence sets of the quantifiers, assuming that all other variables than the clones and originals are in that list. Essentially, this formula says that the Skolem functions f_y and f_{y_1} chosen for y and y_1, respectively, are the same. The next lemma makes this precise and establishes some useful properties of the clone construction.

Lemma 3.3. *Let $\mathfrak{A}$ be a structure and let $\boldsymbol{x}$ be a k-tuple of variables.*

(i) Let ψ be a formula defined as $\psi(\boldsymbol{x}, y) := \forall \boldsymbol{x}'\exists y'\mathrm{clone}(\boldsymbol{x}, y)\psi'$. Then the trumps for ψ are precisely the sets of deals functional in $\boldsymbol{x}$ and y with some Skolem function f, such that the deals $(\ldots, \boldsymbol{x}, f(\boldsymbol{x}), \boldsymbol{x}', f(\boldsymbol{x}'))$ form a trump for ψ'. In particular, if ψ' is **true**, *then the trumps of ψ are just the deals functional in $\boldsymbol{x}$ and y.*

(ii) Let $\varphi(\boldsymbol{x}', y')$ be a formula with only functional trumps and let ψ be defined as $\psi(\boldsymbol{x}, y) := \forall \boldsymbol{x}'\exists y'\mathrm{clone}(\boldsymbol{x}, y)\ \varphi$. Then the trumps for ψ and the trumps for φ are the same, in the sense that for every trump $T' \subseteq A^{k+1}$ of φ there is a trump $T \subseteq A^{k+1}$ of ψ such that an assignment of elements $\boldsymbol{a}$ to the variables $\boldsymbol{x}'$ and b to y' is a deal in T' if, and only if, the corresponding assignment of $\boldsymbol{a}$ to $\boldsymbol{x}$ and b to y is a deal in T and, conversely, for every trump T of ψ there is a corresponding trump T' for φ.

Proof. We first prove Part (i) of the lemma. Following our notation, the formula ψ is an abbreviation for

$$\forall \boldsymbol{x}'/\{\boldsymbol{x}, y\}\exists y'/\{\boldsymbol{x}, y\}(\boldsymbol{x} = \boldsymbol{x}' \to y = y') \wedge \psi'.$$

Towards a contradiction, suppose there was a non-functional trump T for ψ, i.e. T contains deals $(\boldsymbol{a}, b)$ and $(\boldsymbol{a}, b')$ for some $\boldsymbol{a}$ and $b \neq b'$. By the semantics of universal quantifiers, this implies that there must be a trump for $\exists y_1/\{\boldsymbol{x}, y\}(\boldsymbol{x} = \boldsymbol{x}' \to y = y') \wedge \psi'$ containing $(\boldsymbol{a}, b, \boldsymbol{a})$ and $(\boldsymbol{a}, b', \boldsymbol{a})$. But then, the set $\{(\boldsymbol{a}, b, \boldsymbol{a}), (\boldsymbol{a}, b', \boldsymbol{a})\}$ is a $\{\boldsymbol{x}, y\}$-set (recall Definition 2.3). Hence, there must be trump T' for $(\boldsymbol{x} = \boldsymbol{x}' \to y = y') \wedge \psi'$ and an element c so that T' contains the deals $(\boldsymbol{a}, b, \boldsymbol{a}, c)$ and $(\boldsymbol{a}, b', \boldsymbol{a}, c)$. But this is impossible as not both $b = c$ and $b' = c$ can be true but obviously every deal $(\boldsymbol{d}, e, \boldsymbol{d}', e')$ in a trump for $(\boldsymbol{x} = \boldsymbol{x}' \to y = y')$ satisfies the condition that if $\boldsymbol{d} = \boldsymbol{d}'$ then also $e = e'$. Finally, if T is a functional trump, then the corresponding T' must be a trump for ψ', and so the deals $(\boldsymbol{a}, b, \boldsymbol{a}, b)$ must be a trump for ψ'.

Part (ii) of the lemma follows analogously. $\square$

The next lemma shows that every formula in Σ^1_1 is equivalent to a formula in IF-LFP. The proof of the lemma follows easily from the work on Henkin-quantifiers. However, some care has to be taken with free occurrences of function variables.

Lemma 3.4. *Let $\varphi(f_1, \ldots, f_n)$ be a Σ^1_1-formula with free function variables $f_1, \ldots, f_k$. Then there is a formula $\hat{\varphi}(\boldsymbol{x}_{f_1}, y_{f_1}, \ldots, \boldsymbol{x}_{f_k}, y_{f_k}) \in$ IF-FOL such that for every structure $\mathfrak{A}$ a set T is a maximal trump for $\hat{\varphi}$ if, and only if, there are functions $F_1, \ldots, F_k$ such that $\mathfrak{A} \models \varphi(F_1, \ldots, F_k)$ and*

$$T = \{(\boldsymbol{a}_1, b_1, \ldots, \boldsymbol{a}_k, b_k) : F_i(\boldsymbol{a}_i) = b_i \textit{ for all } 1 \leq i \leq k\}.$$

We are now ready to prove the main theorem of this section.

Theorem 3.5. *Let $\mathscr{R}$ be a $(1, k)$-ary third-order variable and let $\varphi(\mathscr{R}, f)$ be a Σ^1_1-inductive formula where f is a k-ary function symbol. Then there is a formula $\hat{\varphi}(R, \boldsymbol{x}, y) \in$ IF-LFP, where R is a $k+1$-ary second-order variable that only occurs positively in φ and $\boldsymbol{x}$ is a k-tuple of variables, such that the least fixpoint R^∞ of φ satisfies the following properties.*

1. *Every trump T in R^∞ is functional.*
2. *Every maximal trump encodes the graph of a function in $\mathscr{R}^\infty$ and, conversely,*
3. *for every function $f \in \mathscr{R}^\infty$ there is a trump T in R^∞ encoding the graph of f.*

Proof. Let $\varphi(\mathscr{R}, f)$ be as in the statement of the theorem. W.l.o.g. we assume that φ has the form $\varphi(\mathscr{R}, f_0) := \varphi_0(f_0) \vee \exists f_1 \ldots \exists f_n \big((\bigwedge_{i=1}^n \mathscr{R} f_i) \wedge \varphi_1\big)$ so that $\mathscr{R}$ does not occur in φ_0 or φ_1. (See [EF99] for a proof of this normal form for existential first-order inductions. The proof for this case is analogous.) The formula φ is translated into a formula $\hat{\varphi}(R, \boldsymbol{x}, y) \in$ IF-LFP defined as follows:

$$\hat{\varphi}(R, \boldsymbol{x}, y) := \forall \boldsymbol{x}_1 . \exists y_1 . \mathrm{clone}(\boldsymbol{x}, y)\big(\psi_0(\boldsymbol{x}, y) \ \vee \ \psi_1(R, \boldsymbol{x}_1, y_1)\big)$$

where

$$\psi_0(\boldsymbol{x}, y) := \forall \boldsymbol{x}_{f_0} \exists y_{f_0} \, \mathrm{clone}(\boldsymbol{x}, y) \ \hat{\varphi}_0(\boldsymbol{x}_{f_0}, y_{f_0})$$

and

$$\psi_1(R, \boldsymbol{x}_1, y_1) := \forall \boldsymbol{x}_{f_0} \exists y_{f_0} \, \mathrm{clone}(\boldsymbol{x}_1, y_1) \ \psi'_1(\boldsymbol{x}_{f_0}, y_{f_0})$$

and

$$\psi'_1(R, \boldsymbol{x}_{f_0}, y_{f_0}) := \forall \boldsymbol{x}_{f_1} \exists y_{f_1} \ldots \forall \boldsymbol{x}_{f_n} \exists y_{f_n} \bigwedge_{i=1}^n (\forall \boldsymbol{x}' \exists y' \mathrm{clone}(\boldsymbol{x}_{f_i}, y_{f_i}) \ R\boldsymbol{x}'y') \wedge \hat{\varphi}_1(\boldsymbol{x}_{f_0}, y_{f_0}, \boldsymbol{x}_{f_1}, y_{f_1}, \ldots, \boldsymbol{x}_{f_n}, y_{f_n}).$$

Here $\hat{\varphi}_0$ and $\hat{\varphi}_1$ are the formulae obtained from φ_0 and φ_1 by applying Lemma 3.4. We claim that the formula $\hat{\varphi}$ satisfies the properties stated in the theorem. Let $\mathfrak{A}$ be a structure with universe A. By Lemma 3.3(i), the trumps T for are functional in $\boldsymbol{x}$ and y, with Skolem function g such that g satisfies ψ_0 or ψ_1.

The theorem now follows by showing via an induction on the ordinals α that every maximal trump in R^α is the graph of a function in $\mathscr{R}^\alpha$ and, conversely, the graph of every function in $\mathscr{R}^\alpha$ is a trump in R^α. $\square$

4 Independence-Friendly vs. Partial Fixpoint Logic

By definition, independence-friendly fixpoint logic is a least fixpoint logic. However, contrary to the fixpoint logics usually considered in finite model theory, here the fixpoints are not sets of elements but sets of trumps and therefore essentially third-order objects. In particular, it is no longer guaranteed that any fixpoint induction closes in polynomially many steps in the size of the structure – to the contrary, it may take an exponential number of steps to close. We will see below, that this greatly increases the expressive power of IF-LFP compared to normal least fixpoint logics.

As a first step in this direction we relate independence-friendly fixpoint logic to partial fixpoint logic. Partial fixpoint logic is an important logic in finite model theory. Syntactically, PFP is defined as the extension of first-order logic by formulae $\psi := [\mathbf{pfp}_{R,\boldsymbol{x}}\, \varphi](\boldsymbol{t})$, where R is a second-order variable of arity k, $\boldsymbol{x}$ a k-tuple of variables, $\boldsymbol{t}$ a k-tuple of terms, and φ itself an arbitrary PFP-formula. In particular, R may occur positive and negative in φ. On any finite structure $\mathfrak{A}$ with universe A the formula φ defines a sequence R^α, $\alpha < \omega$, of sets defined as $R^0 := \varnothing$ and $R^{\alpha+1} := \{\boldsymbol{a} : (\mathfrak{A}, R^\alpha) \models \varphi(\boldsymbol{a})\}$. As there are no restrictions on φ, this sequence need not reach a fixpoint. In this case, ψ is equivalent on $\mathfrak{A}$ to false. Otherwise, if the sequence becomes stationary and reaches a fixpoint R^∞, then for any tuple $\boldsymbol{a} \in A^k$, $\mathfrak{A} \models [\mathbf{pfp}_{R,\boldsymbol{x}}\, \varphi](\boldsymbol{a})$ if, and only if, $\boldsymbol{a} \in R^\infty$.

Among the various fixpoint logics commonly considered in finite model theory, PFP is the most expressive subsuming logics such as LFP and IFP and, on ordered structures, even second-order logic SO.

A central issue in finite model theory is to relate the expressive power of logics to the computational complexity of classes of structures definable in the logic. Of particular interest are so-called *capturing results*: A logic $\mathscr{L}$ *captures* a complexity class $\mathfrak{C}$ if every class of finite structures definable in $\mathscr{L}$ can be decided in $\mathfrak{C}$ and conversely, for every class $\mathscr{C}$ of finite structures which can be decided in $\mathfrak{C}$ there is a sentence $\varphi \in \mathscr{L}$ such that for all structures $\mathfrak{A}$, $\mathfrak{A} \models \varphi$ if, and only if, $\mathfrak{A} \in \mathscr{C}$.

Capturing results are important as they provide logical characterisations of complexity classes, i.e. characterisations independent of machine models and time or space bounds. In particular, non-expressibility results on the logic transfer directly into non-definability results on the complexity class. As such results are notoriously hard to come by, capturing results provide an interesting alternative for proving non-definability of problems in a complexity class.

Much effort has been spent on capturing results and for all major complexity classes such results have been found (see [EF99] for a summary). However, in many cases it could only be shown that a logic captures a complexity class on the class of ordered structures. As for PFP, it has been shown by Abiteboul and Vianu [AV89], that PFP captures PSPACE on the class of finite ordered structures.

As every class of structures definable in second-order logic is decidable in the polynomial time hierarchy, it follows immediately that PFP contains SO on ordered structures. One feature that makes PFP so expressive is its ability to define fixpoint inductions of exponential length in the size of the structure. We show next that every formula of PFP is equivalent to one in IF-LFP. For this we show that every partial fixed-point induction can be translated into a Σ^1_1-inductive definition which, by Theorem 3.5, is equivalent to a formula in IF-LFP. Due to space restrictions we refrain from giving the full proof here and refer to the full version of the paper.

Theorem 4.1. *For every formula $\varphi \in$ PFP there is an equivalent formula $\psi \in$ IF-LFP.*

We have already mentioned that pure independence-friendly logic is equivalent to Σ^1_1 where we can use an existential second-order quantifier to state the existence of a linear order on the universe of a structure even on classes of otherwise unordered structures. Thus the theorem above implies that IF-LFP contains SO on all rather than just ordered structures.

Corollary 4.2. *On finite structures, every formula of* SO *is equivalent to a formula in* IF-LFP*.*

In the next section we will derive some further corollaries of this theorem concerning the model-checking complexity of IF-LFP.

5 Complexity of Independence-Friendly Fixpoint Logic

In this section we analyse the complexity of IF-LFP on finite structures, both with respect to data and model-checking complexity. By data-complexity we understand the complexity of deciding for a fixed formula $\varphi \in$ IF-LFP and a given structure $\mathfrak{A}$ whether $\mathfrak{A} \models \varphi$. In particular, the input only consists of the structure $\mathfrak{A}$. By model-checking we mean the problem of deciding for a given finite structure $\mathfrak{A}$ and formula $\varphi \in$ IF-LFP whether $\mathfrak{A} \models \varphi$. Here, both φ and $\mathfrak{A}$ are part of input.

We begin our analysis with data-complexity. In [Bra03], the first author already noticed that any given formula of IF-LFP can be evaluated in time exponential in the size of the structure. For, every fixpoint $\mu R(\boldsymbol{x}).\varphi$ can be evaluated in time linear in the number of trumps for φ and therefore exponential in the size of the structure.

Proposition 5.1. IF-LFP *has exponential time data-complexity.*

We aim at a much stronger result. Not only will we show that IF-LFP is ExpTime-complete with respect to data-complexity but we will prove that it actually captures ExpTime, i.e. every class of structures decidable by an exponential time Turing-machine can be defined in IF-LFP and vice versa every class of structures definable in IF-LFP can be decided in deterministic exponential time. Again we refrain from giving the full proof here and refer to the full paper.

Theorem 5.2. IF-LFP *captures* ExpTime*.*

Clearly, if a logic $\mathscr{L}$ captures a complexity class $\mathfrak{C}$, then the evaluation problem of $\mathscr{L}$ must be $\mathfrak{C}$-complete with respect to data complexity. Thus we get the following simple corollary.

Corollary 5.3. *There exist formulae in* IF-LFP *with* ExpTime*-complete data-complexity.*

We continue our complexity analysis of IF-LFP with the study of its model-checking complexity. For an upper bound, it is easily seen that for any given structure $\mathfrak{A}$ and formula φ the formula can be evaluated in $\mathfrak{A}$ using space doubly exponential in $|\varphi|$ and exponential in $|\mathfrak{A}|$. For, every evaluation of a fixpoint only needs enough space to store all possible trumps, and the number of trumps is bounded by $\mathrm{O}(2^{\mathfrak{A}^{|\varphi|}})$.

Theorem 5.4. *Every formula $\varphi \in$* IF-LFP *can be evaluated in a structure $\mathfrak{A}$ in space doubly exponential in $|\varphi|$ and exponential in $|\mathfrak{A}|$.*

The theorem gives an upper bound on the model checking complexity of IF-LFP. We have seen in Section 4 above that every formula of PFP is equivalent to one of IF-LFP. Further, the translation is polynomial in the size of the PFP-formula. Consequently, model-checking for IF-LFP is at least as complex as it is for PFP. As model-checking for PFP is known to be hard for exponential space – in fact even complete for exponential space – we get the following theorem.

Theorem 5.5. *The model-checking problem for* IF-LFP *is hard for exponential space.*

6 Conclusion

In this paper we studied the computational complexity of various problems related to IF-LFP. As we have seen, adding independence to least fixpoint logic increases the expressive power and complexity significantly. Another indicator for this is the translation of formulae of PFP to formulae of IF-LFP. This showed that IF-LFP is even more expressive than second-order logic – unless, of course, PSpace = ExpTime.

Looking at the various proofs given for the results, it becomes clear that the common technique used in all proofs was to use independent quantification to define functions and then show that these functions can be passed through the fixpoint induction. This suggests that there might be a more general relation between independence-friendly logic and second-order logic, namely that the two logics are actually equivalent. Showing this, however, requires a careful analysis of the role of negation in independence friendly logics and is far from obvious. This is part of ongoing work.

7 Acknowledgements

Part of this work was done while the second author was a postdoctoral fellow in Edinburgh supported by the EU Research and Training Network GAMES (Games and Automata for Synthesis and Validation).

The authors thank anonymous referees for comments which have improved the paper.

References

[AV89] S. Abiteboul and V. Vianu, Fixpoint extensions of first-order logic and Datalog-like languages. *Proc. 4th IEEE Symp. on Logic in Computer Science (LICS)*, 71–79 (1989).

[Bra99] J. C. Bradfield, Fixpoints in arithmetic, transition systems and trees. *Theoretical Informatics and Applications*, **33** 341–356 (1999).

[Bra00] J. C. Bradfield, Independence: logics and concurrency, *Proc. CSL 2000*, LNCS **1862** 247–261 (2000).

[Bra03] J. C. Bradfield, Parity of imperfection, *Proc. CSL 2003*, LNCS **2803** 72–85 (2003).

[BrF02] J. C. Bradfield and S. B. Fröschle, Independence-friendly modal logic and true concurrency, *Nordic J. Computing* **9** 102–117 (2002).

[EF99] H.-D. Ebbinghaus and J. Flum, Finite Model Theory, 2nd edition, Springer, 1999.

[End70] H. B. Enderton, Finite partially ordered quantifiers, *Z. für Math. Logik u. Grundl. Math.* **16** 393–397 (1970).

[Gra03] E. Grädel, Finite Model Theory and Descriptive Complexity, in *Finite Model Theory and Its Applications*, Springer, 2005. See http://www-mgi.informatik.rwth-aachen.de/Publications/pub/graedel/Gr-FMTbook.ps

[HiS96] J. Hintikka and G. Sandu, A revolution in logic?, *Nordic J. Philos. Logic* **1**(2) 169–183 (1996).

[Hod97] W. Hodges, Compositional semantics for a language of imperfect information, *Int. J. IGPL* **5**(4), 539–563.

[Pie00] A. Pietarinen, Games logic plays. Informational independence in game-theoretic semantics. D.Phil. thesis, Univ Sussex (2000).

[Wal70] W. J. Walkoe, Jr, Finite partially-ordered quantification. *J. Symbolic Logic* **35** 535–555 (1970).