Logical Methods in Computer Science
Vol. 3 (2:7) 2007, pp. 1–33
www.lmcs-online.org

Submitted Oct. 25, 2006
Published Jun. 29, 2007

THE COMPLEXITY OF MODEL CHECKING HIGHER-ORDER FIXPOINT LOGIC

ROLAND AXELSSON [a], MARTIN LANGE [b], AND RAFAŁ SOMLA [c]

[a] Department of Computer Science, University of Munich, Germany
e-mail address: Roland.Axelsson@ifi.lmu.de

[b] Department of Computer Science, University of Aarhus, Denmark
e-mail address: Martin.Lange@ifi.lmu.de

[c] IT Department, Uppsala University, Sweden
e-mail address: Rafal.Somla@it.uu.se

Abstract. Higher-Order Fixpoint Logic (HFL) is a hybrid of the simply typed λ-calculus and the modal μ-calculus. This makes it a highly expressive temporal logic that is capable of expressing various interesting correctness properties of programs that are not expressible in the modal μ-calculus.

This paper provides complexity results for its model checking problem. In particular, we consider those fragments of HFL that are built by using only types of bounded order k and arity m. We establish k-fold exponential time completeness for model checking each such fragment. For the upper bound we use fixpoint elimination to obtain reachability games that are singly-exponential in the size of the formula and k-fold exponential in the size of the underlying transition system. These games can be solved in deterministic linear time. As a simple consequence, we obtain an exponential time upper bound on the expression complexity of each such fragment.

The lower bound is established by a reduction from the word problem for alternating $(k-1)$-fold exponential space bounded Turing Machines. Since there are fixed machines of that type whose word problems are already hard with respect to k-fold exponential time, we obtain, as a corollary, k-fold exponential time completeness for the data complexity of our fragments of HFL, provided m exceeds 3. This also yields a hierarchy result in expressive power.

1. Introduction

Temporal logics are well-established tools for the specification of correctness properties and their verification in hard- and software design processes. One of the most famous temporal logics is Kozen's modal μ-calculus $\mathcal{L}_\mu$ [15] which extends multi-modal logic with extremal fixpoint quantifiers. $\mathcal{L}_\mu$ subsumes many other temporal logics like PDL [11] as well as CTL* [9], and with it CTL [8] and LTL [23]. It also has connections to other formalisms like description logics for example.

2000 ACM Subject Classification: F.3.1, F.4.1.
Key words and phrases: μ-calculus, λ-calculus, model checking, complexity.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-3 (2:7) 2007

$\mathcal{L}_\mu$ is equi-expressive to the bisimulation-invariant fragment of Monadic Second Order Logic over trees or graphs [10, 13]. Hence, properties expressed by formulas of the modal μ-calculus are only regular. There are, however, many interesting correctness properties of programs that are not regular. Examples include *uniform inevitability* [7] which states that a certain event occurs globally at the same time in all possible runs of the system; counting properties like "at any point in a run of a protocol there have never been more *send*- than *receive*-actions"; formulas saying that an unbounded number of data does not lose its order during a transmission process; or properties making structural assertions about their models like being bisimilar to a linear time model.

When program verification was introduced to computer science, programs as well as their correctness properties were mainly specified in temporal logics. Hence, verification meant to check formulas of the form $\varphi \to \psi$ for validity, or equally formulas of the form $\varphi \wedge \psi$ for satisfiability. An intrinsic problem for this approach and non-regular properties is undecidability. Note that the intersection problem for context-free languages is already undecidable [1].

One of the earliest attempts at verifying non-regular properties of programs was *Non-Regular PDL* [12] which enriches ordinary PDL by context-free programs. Non-Regular PDL is highly undecidable, hence, the logic did not receive much attention for program verification purposes. Its model checking problem, however, remains decidable on finite transition systems – it is even in P [16].

Another example is *Fixpoint Logic with Chop*, FLC, [22] which extends $\mathcal{L}_\mu$ with a sequential composition operator. It is capable of expressing many non-regular – and even non-context-free – properties, and its model checking problem on finite transition systems is decidable in deterministic exponential time [21]. It also properly subsumes Non-Regular PDL [20].

In order to achieve non-regular effects in FLC, the original $\mathcal{L}_\mu$ semantics is lifted to a function from sets of states to sets of states. This idea has been followed consequently in the introduction of *Higher-Order Fixpoint Logic*, HFL, [28] which incorporates a simply typed λ-calculus into the modal μ-calculus. This gives it even more expressive power than FLC. HFL is, for example, capable of expressing *assume-guarantee-properties*. Still, HFL's model checking problem on finite transition systems remains decidable. This has been stated in its introductory work [28]. It is also known that model checking HFL is non-elementary with the following complexity bounds [19].

- When restricted to function types of order k, the model checking problem for this fragment is hard for deterministic $(k-3)$-fold exponential space and included in determinisitic $(k+1)$-fold exponential time. It is not made explicit, though, that the arity of types needs to be fixed for that.
- The model checking problem is non-elementary on fixed (and very small) structures already. However, unbounded type orders are needed for this result.

Our aim is to close this apparent gap and to provide an analysis of the model checking problem for HFL and, thus, the problem of automatically verifying non-regular properties on finite transition systems.

We start in Sect. 2 by recalling the logic and giving a few examples of HFL-expressible properties. Sect. 3 contains a reduction from HFL's model checking problem to the problem of solving (rather large) reachability games. This improves the upper bound mentioned

above: these games can be solved in k-fold exponential time when type orders are bounded by k and arities are fixed.

Sect. 4 presents a reduction from the word problem for alternating space-bounded Turing Machines to HFL's model checking problem. This improves on the lower bounds mentioned above in two ways. For the fragment of type orders restricted to k we can match the new upper bound and establish completeness for the class of k-fold deterministic exponential time. A slight modification produces formulas that are independent of the input word to the Turing Machine. Hence, we get a result on the data complexity of HFL as a simple corollary. This, in turn, yields a hierarchy result on expressive power within HFL.

A non-elementary lower complexity bound on the problem of a logic that incorporates the simply typed λ-calculus is of course reminiscent of Statman's result which states that the normalisation problem in the simply typed λ-calculus is non-elementary [25]. But this is rather related to the equivalence problem for HFL which is known to be highly undecidable [12, 20, 28]. Since HFL is a branching time logic there is probably no simple reduction from the equivalence problem to the model checking problem. Hence, the lower bounds presented here do not necessarily follow from Statman's result.

Furthermore, Statman's result is of course irrelevant for the upper bounds presented here. There is some work on upper bounds for the number of β-reduction steps in the simply typed λ-calculus, c.f. [24]. However, this is not good enough to obtain the upper bounds we are after, c.f. Sect. 3. It also does not deal with the propositional, modal and fixpoint parts of HFL formulas.

2. Preliminaries

2.1. The Syntax of Formulas.

Definition 2.1. Let $\mathcal{P} = \{p, q, \ldots\}$ be a set of atomic propositions, $\mathcal{A} = \{a, b, \ldots\}$ be a finite set of action names, and $\mathcal{V} = \{X, Y, \ldots\}$ a set of variables. For simplicity, we fix $\mathcal{P}$, $\mathcal{A}$, and $\mathcal{V}$ for the rest of the paper.

A $v \in \{-, +, 0\}$ is called a variance. The set of HFL types is the smallest set containing the atomic type Pr and being closed under function typing with variances, i.e. if σ and τ are HFL types and v is a variance, then $\sigma^v \to \tau$ is an HFL type.

Formulas of HFL are given by the following grammar:

$$\varphi \ ::= \ q \mid X \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle a \rangle \varphi \mid \varphi \ \varphi \mid \lambda(X^v : \tau).\varphi \mid \mu(X : \tau).\varphi$$

where $q \in \mathcal{P}$, $X \in \mathcal{V}$, $a \in \mathcal{A}$, v is a variance and τ is an HFL type.

An HFL formula φ is called *fixpoint-free* if it does not contain any subformula of the form $\mu X.\psi$.

Throughout this paper we will adopt the convention given by the syntax of HFL and write function application in the style $f\ x$ rather than $f(x)$.

We use the following standard abbreviations:

$$\begin{array}{rcl@{\qquad}rcl} \mathtt{tt} & := & q \vee \neg q \text{ for some } q \in \mathcal{P} & \mathtt{ff} & := & \neg\mathtt{tt} \\ \varphi \wedge \psi & := & \neg(\neg\varphi \vee \neg\psi) & \varphi \to \psi & := & \neg\varphi \vee \psi \\ \varphi \leftrightarrow \psi & := & (\varphi \to \psi) \wedge (\psi \to \varphi) & \nu X.\varphi & := & \neg\mu X.\neg\varphi[\neg X/X] \\ [a]\psi & := & \neg\langle a \rangle\neg\psi & \langle - \rangle\varphi & := & \bigvee_{a \in \mathcal{A}} \langle a \rangle \varphi \\ [-]\varphi & := & \bigwedge_{a \in \mathcal{A}} [a]\varphi & & & \end{array}$$

$$\frac{}{\Gamma \vdash q\colon \mathsf{Pr}} \qquad \frac{v \in \{0,+\}}{\Gamma, X^v\colon \tau \vdash X\colon \tau} \qquad \frac{\Gamma^- \vdash \varphi\colon \tau}{\Gamma \vdash \neg\varphi\colon \tau}$$

$$\frac{\Gamma \vdash \varphi\colon \mathsf{Pr} \quad \Gamma \vdash \psi\colon \mathsf{Pr}}{\Gamma \vdash \varphi \vee \psi\colon \mathsf{Pr}} \qquad \frac{\Gamma \vdash \varphi\colon \mathsf{Pr}}{\Gamma \vdash \langle a\rangle\varphi\colon \mathsf{Pr}} \qquad \frac{\Gamma, X^v\colon \sigma \vdash \varphi\colon \tau}{\Gamma \vdash \lambda(X^v\colon \sigma).\varphi : (\sigma^v \to \tau)}$$

$$\frac{\Gamma \vdash \varphi : (\sigma^+ \to \tau) \quad \Gamma \vdash \psi\colon \sigma}{\Gamma \vdash (\varphi\ \psi) : \tau} \qquad \frac{\Gamma \vdash \varphi : (\sigma^- \to \tau) \quad \Gamma^- \vdash \psi\colon \sigma}{\Gamma \vdash (\varphi\ \psi) : \tau}$$

$$\frac{\Gamma \vdash \varphi : (\sigma^0 \to \tau) \quad \Gamma \vdash \psi\colon \sigma \quad \Gamma^- \vdash \psi\colon \sigma}{\Gamma \vdash (\varphi\ \psi) : \tau} \qquad \frac{\Gamma, X^+\colon \tau \vdash \varphi\colon \tau}{\Gamma \vdash \mu(X\colon \tau).\varphi : \tau}$$

Figure 1: Type inference rules for HFL.

where $\varphi[\psi/X]$ denotes the formula that results from φ by replacing simultaneously every occurrence of X by ψ.

Definition 2.2. A sequence Γ of the form $X_1^{v_1}\colon \tau_1, \ldots, X_n^{v_n}\colon \tau_n$ where X_i are variables, τ_i are types and v_i are variances is called a *context* (we assume all X_i are distinct). An HFL formula φ has type τ in context Γ if the statement $\Gamma \vdash \varphi\colon \tau$ can be inferred using the rules of Fig. 1. We say that φ is *well-formed* if $\Gamma \vdash \varphi\colon \tau$ for some Γ and τ.

For a variance v, we define its complement v^- as $+$ if $v = -$, as $-$ if $v = +$, and 0 otherwise. For a context $\Gamma = X_1^{v_1}\colon \tau_1, \ldots, X_n^{v_n}\colon \tau_n$, the complement Γ^- is defined as $X_1^{v_1^-} : \tau_1, \ldots, X_n^{v_n^-} : \tau_n$.

Definition 2.3. The Fischer-Ladner closure of an HFL formula φ_0 is the least set $FL(\varphi_0)$ that contains φ_0 and satisfies the following.

- If $\psi_1 \vee \psi_2 \in FL(\varphi_0)$ then $\{\psi_1, \psi_2\} \subseteq FL(\varphi_0)$.
- If $\neg(\psi_1 \vee \psi_2) \in FL(\varphi_0)$ then $\{\neg\psi_1, \neg\psi_2\} \subseteq FL(\varphi_0)$.
- If $\langle a\rangle\psi \in FL(\varphi_0)$ then $\psi \in FL(\varphi_0)$.
- If $\neg\langle a\rangle\psi \in FL(\varphi_0)$ then $\neg\psi \in FL(\varphi_0)$.
- If $\varphi\ \psi \in FL(\varphi_0)$ then $\{\varphi, \psi, \neg\psi\} \subseteq FL(\varphi_0)$.
- If $\neg(\varphi\ \psi) \in FL(\varphi_0)$ then $\{\neg\varphi, \psi, \neg\psi\} \subseteq FL(\varphi_0)$.
- If $\lambda X.\psi \in FL(\varphi_0)$ then $\psi \in FL(\varphi_0)$.
- If $\neg(\lambda X.\psi) \in FL(\varphi_0)$ then $\neg\psi \in FL(\varphi_0)$.
- If $\mu X.\psi \in FL(\varphi_0)$ then $\psi \in FL(\varphi_0)$.
- If $\neg(\mu X.\psi) \in FL(\varphi_0)$ then $\neg\psi[\neg X/X] \in FL(\varphi_0)$.
- If $\neg\neg\psi \in FL(\varphi_0)$ then $\psi \in FL(\varphi_0)$.
- If $\neg X \in FL(\varphi_0)$ then $X \in FL(\varphi_0)$.
- If $\neg q \in FL(\varphi_0)$ then $q \in FL(\varphi_0)$.

Note that the size of $FL(\varphi)$ as a set is at most twice the length of φ. We therefore define $|\varphi| := |FL(\varphi)|$. Another measure for the complexity of a formula is the number $v(\varphi)$ of distinct λ-bound variables occurring in φ. Formally, let $v(\varphi) := |\{X \mid \lambda X.\psi \in FL(\varphi) \text{ for some } \psi\}|$.[1]

[1]Note that we do not require α-equivalent formulas to have exactly the same computational measures.

When using least fixpoint quantifiers it is often beneficial to recall the Békič principle [2] which states that a simultaneously defined least fixpoint of a monotone function is the same as a parametrised one. We will use this to allow formulas like

$$\varphi \;:=\; \mu X_i. \begin{pmatrix} X_1 & . & \varphi_1(X_1,\ldots,X_n) \\ & \vdots & \\ X_n & . & \varphi_n(X_1,\ldots,X_n) \end{pmatrix}$$

in the syntax of HFL. This abbreviates

$$\varphi' \;:=\; \mu X_i.\varphi_i(\mu X_1.\varphi_1(X_1,\mu X_2.\varphi_2(X_1,X_2,\ldots,X_i,\ldots),\ldots,X_i,\ldots),\mu X_2\ldots,\ldots,X_i,\ldots)$$

Note that the size of φ' can be exponentially bigger than the size of φ, and this even holds for the number of their subformulas. However, it is only exponential in n, not in $|\varphi|$: $|\varphi'| = O(|\varphi| \cdot 2^n)$.

2.2. **The Semantics of Types and Formulas.**

Definition 2.4. A (labeled) transition system is a structure $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ where $\mathcal{S}$ is a finite non-empty set of states, $\xrightarrow{a}$ is a binary relation on states for each $a \in \mathcal{A}$, and $L : \mathcal{S} \to 2^{\mathcal{P}}$ is a function labeling each state with the set of propositional constants that are true in it.

The semantics of a type w.r.t. a transition system $\mathcal{T}$ is a Boolean lattice[2], inductively defined on the type as

$$[\![\mathsf{Pr}]\!]^{\mathcal{T}} = (2^{\mathcal{S}}, \sqsubseteq_{\mathsf{Pr}}) \,, \qquad [\![\sigma^v \to \tau]\!]^{\mathcal{T}} = \left(([\![\sigma]\!]^{\mathcal{T}})^v \to [\![\tau]\!]^{\mathcal{T}}, \sqsubseteq_{\sigma^v \to \tau}\right) .$$

where $\sqsubseteq_{\mathsf{Pr}}$ is simply the set inclusion order $\subseteq$. For two partial orders $\bar{\tau} = (\tau, \sqsubseteq_\tau)$ and $\bar{\sigma} = (\sigma, \sqsubseteq_\sigma)$, $\bar{\sigma} \to \bar{\tau}$ denotes the partial order of all monotone functions ordered pointwise. I.e., in this case,

$$f \sqsubseteq_{\sigma^v \to \tau} g \quad \text{iff} \quad \text{for all } x \in [\![\sigma]\!]^{\mathcal{T}} : f\ x \sqsubseteq_\tau g\ x$$

Moreover, complements in these lattices are denoted by $\bar{f}$ and defined on higher levels as $\bar{f}\ x = \overline{f\ x}$.

A positive variance leaves a partial order unchanged, $\bar{\tau}^+ = (\tau, \sqsubseteq_\tau)$, a negative variance turns it upside-down to make antitone functions look well-behaved, $\bar{\tau}^- = (\tau, \sqsupseteq_\tau)$, and a neutral variance flattens it, $\bar{\tau}^0 = (\tau, \sqsubseteq_\tau \cap \sqsupseteq_\tau)$. This is not a complete lattice anymore which does not matter since variances only occur on the left of a typing arrow. Note that the space of monotone functions from a partial order to a Boolean lattice with pointwise ordering forms a Boolean lattice again.

Definition 2.5. An *environment* η is a possibly partial map on the variable set $\mathcal{V}$. For a context $\Gamma = X_1^{v_1} : \tau_1, \ldots, X_n^{v_n} : \tau_n$, we say that η respects Γ, denoted by $\eta \models \Gamma$, if $\eta(X_i) \in [\![\tau_i]\!]^{\mathcal{T}}$ for $i \in \{1,\ldots,n\}$. We write $\eta[X \mapsto f]$ for the environment that maps X to f and otherwise agrees with η. If $\eta \models \Gamma$ and $f \in [\![\tau]\!]^{\mathcal{T}}$ then $\eta[X \mapsto f] \models \Gamma, X : \tau$, where X is a variable that does not appear in Γ.

[2]In the original definition, the semantics is only said to be a complete lattice but it is in fact also Boolean. The reason for this is that negation is only allowed on the ground type Pr anyway. The game-based characterisation of HFL's model checking problem in the following section benefits from a symmetric definition w.r.t. negation. Hence, we allow negation in the syntax on arbitrary type levels. But then we have to also use the property of being Boolean of the complete lattices that form the basis for the definition of the semantics.

$$
\begin{array}{rcl}
[\![\Gamma \vdash q : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta} & = & \{s \in \mathcal{S} \mid q \in L(s)\} \\
[\![\Gamma \vdash X : \tau]\!]^{\mathcal{T}}_{\eta} & = & \eta(X) \\
[\![\Gamma \vdash \neg\varphi : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta} & = & \mathcal{S} \setminus [\![\Gamma^{-} \vdash \varphi : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \neg\varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta} & = & f \in [\![\sigma^{v} \to \tau]\!]^{\mathcal{T}} \text{ s.t. } \bar{f} = [\![\Gamma^{-} \vdash \varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \varphi \vee \psi : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta} & = & [\![\Gamma \vdash \varphi : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta} \cup [\![\Gamma \vdash \psi : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \langle a \rangle\varphi : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta} & = & \{s \in \mathcal{S} \mid s \xrightarrow{a} t \text{ for some } t \in [\![\Gamma \vdash \varphi : \mathsf{Pr}]\!]^{\mathcal{T}}_{\eta}\} \\
[\![\Gamma \vdash \lambda(X^{v} : \sigma).\varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta} & = & f \in [\![\sigma^{v} \to \tau]\!]^{\mathcal{T}} \text{ s.t. } \forall x \in [\![\sigma]\!]^{\mathcal{T}} \\
 & & \quad f\ x = [\![\Gamma, X^{v} : \sigma \vdash \varphi : \tau]\!]^{\mathcal{T}}_{\eta[X \mapsto x]} \\
[\![\Gamma \vdash \varphi\ \psi : \tau]\!]^{\mathcal{T}}_{\eta} & = & [\![\Gamma \vdash \varphi : \sigma^{v} \to \tau]\!]^{\mathcal{T}}_{\eta}\ [\![\Gamma' \vdash \psi : \sigma]\!]^{\mathcal{T}}_{\eta} \\
[\![\Gamma \vdash \mu(X : \tau)\varphi : \tau]\!]^{\mathcal{T}}_{\eta} & = & \bigsqcap\{x \in [\![\tau]\!]^{\mathcal{T}} \mid [\![\Gamma, X^{+} : \tau \vdash \varphi : \tau]\!]^{\mathcal{T}}_{\eta[X \mapsto x]} \sqsubseteq_{\tau} x\}
\end{array}
$$

Figure 2: Semantics of HFL

For any well-typed term $\Gamma \vdash \varphi : \tau$ and environment $\eta \models \Gamma$, Fig. 2 defines the semantics of φ inductively to be an element of $[\![\tau]\!]^{\mathcal{T}}$. In the clause for function application $(\varphi\ \psi)$ the context Γ' is Γ if $v \in \{+, 0\}$, and is Γ^{-} if $v = -$.

The model checking problem for HFL is the following: Given an HFL sentence $\varphi : \mathsf{Pr}$, a transition system $\mathcal{T}$ and one of its states s, decide whether or not $s \in [\![\varphi]\!]^{\mathcal{T}}$.

In the following we will identify a type τ and its underlying complete lattice $[\![\tau]\!]^{\mathcal{T}}$ induced by a transition system $\mathcal{T}$ with state set $\mathcal{S}$. In order to simplify notation we fix $\mathcal{T}$ for the remainder of this section. We will also simply write $|\tau|$ instead of $|[\![\tau]\!]^{\mathcal{T}}|$ for the size of the lattice induced by τ.

Definition 2.6. We consider fragments of formulas that can be built using restricted types only. Note that because of right-associativity of the function arrow, every HFL type is isomorphic to a $\tau = \tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}$ where $m \in \mathbb{N}$. Clearly, for $m = 0$ we simply have $\tau = \mathsf{Pr}$. We stratify types w.r.t. their *order*, i.e. the degree of using proper functions as arguments to other functions, as well as *maximal arity*, i.e. the number of arguments a function has. Order can be seen as depth, and maximal arity as the width of a type. Both are defined recursively as follows.

$$
\begin{array}{rcl}
ord(\tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}) & := & \max\{1 + ord(\tau_i) \mid i = 1, \ldots, m\} \\
mar(\tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}) & := & \max(\{m\} \cup \{mar(\tau_i) \mid i = 1, \ldots, m\})
\end{array}
$$

where we assume $\max \emptyset = 0$. Now let, for $k \geq 1$ and $m \geq 1$,

$$
\begin{array}{rcl}
\mathrm{HFL}^{k,m} & := & \{\ \varphi \in \mathrm{HFL} \mid \vdash \varphi : \mathsf{Pr} \text{ using types } \tau \text{ with } ord(\tau) \leq k \text{ and } mar(\tau) \leq m \text{ only }\} \\
\mathrm{HFL}^{k} & := & \bigcup_{m \in \mathbb{N}} \mathrm{HFL}^{k,m}
\end{array}
$$

Note that no formula can have maximal type order $k > 0$ but maximal type arity $m = 0$. The combination $k = 0$ and $m > 0$ is also impossible. Hence, we define

$$\mathrm{HFL}^{0} = \{\ \varphi \in \mathrm{HFL} \mid \vdash \varphi : \mathsf{Pr} \text{ using types } \tau \text{ with } ord(\tau) = 0 \text{ only }\}$$

We extend these measures to formulas in a straightforward way: $ord(\varphi) = k$ and $mar(\varphi) = m$ iff k and m are the least k' and m' s.t. φ can be shown to have some type using types τ with $ord(\tau) \leq k'$ and $mar(\tau) \leq m'$ only.

Proposition 2.7. $\mathrm{HFL}^0 = \mathcal{L}_\mu$.

Proof. An HFL^0 formula cannot have any subformula of the form $\lambda X.\psi$ or $\varphi\ \psi$. But deleting these two clauses from the definition of HFL's syntax yields exactly the syntax of $\mathcal{L}_\mu$. It is not hard to see that this is faithful, i.e. the semantics of this logic regarded as a fragment of HFL is the same as the semantics of $\mathcal{L}_\mu$. □

2.3. **Examples of Properties Expressible in HFL.**

Example 2.8. HFL can express the non-regular (but context-free) property "on any path the number of *out*'s seen at any time never exceeds the number of *in*'s seen so far." Let

$$\varphi \ := \ \mu(X : \mathsf{Pr} \to \mathsf{Pr}).(\lambda(Z : \mathsf{Pr}).\langle out\rangle Z \vee \langle in\rangle(X\ (X\ Z)))\mathtt{tt}$$

This formula is best understood by comparing it to the CFG $X \to out \mid in\, X\, X$. It generates the language L of all words $w \in \{in, out\}^*\{out\}$ s.t. $|w|_{in} = |w|_{out}$ and for all prefixes v of w we have: $|v|_{in} \geq |v|_{out}$. This language contains exactly those prefixes of buffer runs that are violating due to a buffer underflow. Then $\mathcal{T}, s \models \varphi$ iff there is a finite path through $\mathcal{T}$ starting in s that is labeled with a word in L, and $\neg\varphi$ consequently describes the property mentioned above.

Example 2.9. Another property that is easily seen not to be expressible by a finite tree automaton and, hence, not by a formula of the modal μ-calculus either is *bisimilarity to a word*. Note that a transition system $\mathcal{T}$ with starting state s is not bisimilar to a linear word model iff there are two distinct actions a and b s.t. there are two (not necessarily distinct) states t_1 and t_2 at the same distance from s s.t. $t_1 \xrightarrow{a} t_1'$ and $t_2 \xrightarrow{b} t_2'$ for some t_1', t_2'. This is expressed by the HFL formula

$$\neg\Big(\bigvee_{a\neq b}\big(\mu(F : \mathsf{Pr} \to \mathsf{Pr} \to \mathsf{Pr}).\lambda(X : \mathsf{Pr}).\lambda(Y : \mathsf{Pr}).(X \wedge Y) \vee (F\ \langle -\rangle X\ \langle -\rangle Y)\big)\ \langle a\rangle\mathtt{tt}\ \langle b\rangle\mathtt{tt}\Big)$$

This formula is best understood by regarding the least fixpoint definition F as a functional program. It takes two arguments X and Y and checks whether both hold now or calls itself recursively with the arguments being checked in two (possibly different) successors of the state that it is evaluated in.

Note that here, bisimulation does not consider the labels of states but only the actions along transitions. It is not hard to change the formula accordingly to incorporate state labels as well.

Example 2.10. Let $\mathbf{2}_0^n := n$ and $\mathbf{2}_{m+1}^n := 2^{\mathbf{2}_m^n}$. For any $m \in \mathbb{N}$, there is a short HFL formula φ_m expressing the fact that there is a maximal path of length $\mathbf{2}_m^1$ (number of states on this path) through a transition system. It can be constructed using a typed version of the Church numeral 2. Let $\tau_0 = \mathsf{Pr}$ and $\tau_{i+1} = \tau_i \to \tau_i$. For $i \geq 1$ define ψ_i of type τ_{i+1} as $\lambda(F : \tau_i).\lambda(X : \tau_{i-1}).F\,(F\,X)$. Then

$$\varphi_m := \psi_m\ \psi_{m-1}\ \ldots\ \psi_1\ \big(\lambda(X : \mathsf{Pr}).\langle -\rangle X\big)\ [-]\mathtt{ff}\ .$$

Note that for any $m \in \mathbb{N}$, φ_m is of size linear in m. This indicates that HFL is able to express computations of Turing Machines of arbitrary elementary complexity. Sect. 4 will show that this is indeed the case.

2.4. **Complexity Classes and Alternating Turing Machines.** We will assume familiarity with the concept of a deterministic Turing Machine but quickly recall the less known model of an alternating Turing Machine.

Let $\mathrm{DTime}(f(n))$ be the class of languages that can be recognised by a deterministic Turing Machine in at most $f(n)$ many steps on any input of length n. The k-th level of the exponential time hierarchy for $k \in \mathbb{N}$ is

$$k\textsc{ExpTime} := \bigcup_{p \text{ polynomial}} \mathrm{DTime}(\mathbf{2}_k^{p(n)})$$

Then $\textsc{Elementary} := \bigcup_{k\in\mathbb{N}} k\textsc{ExpTime}$ is the class of problems that can be solved in elementary time. Note that $\textsc{Elementary}$ does not have complete problems because their existence would lead to a collapse of the hierarchy which is not the case.

Definition 2.11. An *alternating Turing Machine* is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, \delta, q_{acc}, q_{rej})$ s.t. its state set Q is partitioned into *existential* states $Q_\exists$, *universal* states $Q_\forall$ and the halting states $\{q_{acc}, q_{rej}\}$. The starting state q_0 is either existential or universal. The input alphabet Σ is a subset of the tape alphabet Γ containing a special blank symbol $\square$. The transition relation is of type $Q \times \Gamma \times Q \times \Gamma \times \{-1, 0, +1\}$.

$\mathcal{M}$ is called $f(n)$-space bounded for some function $f(n)$ if it never uses more than $f(n)$ many tape cells in a computation on a word of length n. A configuration of such an $\mathcal{M}$ is a triple $C \in Q \times \{0, \ldots, f(n) - 1\} \times \Gamma^{f(n)}$ representing the current state, the position of the tape head and the content of the tape. The starting configuration is $C_0 := (q_0, 0, w\square \ldots \square)$. A configuration (q, i, v) is called

- *existential* if $q \in Q_\exists$,
- *universal* if $q \in Q_\forall$,
- *accepting* if $q = q_{acc}$,
- *rejecting* if $q = q_{rej}$.

The computation of $\mathcal{M}$ on w is a tree whose root is C_0 s.t. an existential configuration has exactly one successor configuration in the tree, all possible successor configurations of a universal configurations are present in the tree, and leaves are exactly those configurations that are accepting or rejecting. The successor relation on configurations is the usual one built on the transition relation δ.

W.l.o.g. we can assume that every path of any computation tree of $\mathcal{M}$ on any w will eventually reach an accepting or rejecting configuration. I.e. computation trees are always finite. This can be achieved for example by running an additional clock which causes a transition to the rejecting state when a configuration has been reached repeatedly.

A computation is called *accepting* if all of its leaves are accepting. The machine $\mathcal{M}$ accepts the word $w \in L(\mathcal{M})$, if there is an accepting computation tree of $\mathcal{M}$ on w.

Let $\mathrm{ASpace}(f(n))$ be the class of languages that can be recognised by an $f(n)$-space bounded alternating Turing Machine.

$$k\mathrm{AExpSpace} := \bigcup_{p \text{ polynomial}} \mathrm{ASpace}(\mathbf{2}_k^{p(n)})$$

There is a direct correspondence between the levels of the elementary time hierarchy and classes defined by alternating space-bounded Turing Machines. For all $k \geq 1$ we have $k\text{ExpTime} = (k-1)\text{AExpSpace}$ [4]. We will make use of a related result.

Theorem 2.12 ([4]). *For every $k \geq 1$ there is a polynomial $p(n)$ and some alternating $\mathbf{2}_{k-1}^{p(n)}$-space bounded Turing Machine $\mathcal{A}_k$ s.t. $L(\mathcal{A}_k)$ over a binary alphabet is kExpTime-hard.*

Finally, we need to introduce the class UP – a subclass of NP. UP consists of all problems that are solvable by a non-deterministic polynomial time bounded Turing Machine with at most one accepting computation. As usual, co-UP denotes the complement of UP. Later we will briefly mention the class UP∩co-UP. Note that UP∩co-UP does not have complete problems either.

3. The Upper Bound

We will take two steps in order to obtain a kExpTime upper bound on the model checking problem for $\text{HFL}^{k,m}$ for every $m \in \mathbb{N}$. First we eliminate fixpoint constructs from the formula w.r.t. the underlying transition system. This results in a possibly k-fold exponentially larger modal formula with λ-abstractions and function applications. We then reduce the model checking problem for such formulas to the reachability game problem in graphs of roughly the same size.

The combination of the elimination step and the reduction step is necessary to achieve the kExpTime upper bound. It would be easy to eliminate the λ-calculus part from a fixpoint-free formula using β-reduction. However, the best known upper bounds on the number of reduction steps in the simply typed λ-calculus are approximately of the order $\mathbf{2}_{k+1}^{O(n)}$ [24] which would only yield a $(2k+1)$ExpTime upper bound.

The reason for avoiding the additional $k+1$ exponents is that β-reduction is a purely syntactical procedure. We incorporate semantics into these reachability games by evaluating λ-bound variables to real functions of finite domain and co-domain rather than unwinding the entire syntactical definition of that function as a program in the simply typed λ-calculus. Note that such a function can be represented by more than one λ-term. Whereas equivalence of fixpoint-free HFL formulas is difficult to decide – in fact, it is undecidable in general and might require β-reduction on a fixed transition system – it is easy to decide for unique semantical representations of these functions.

On the other hand, extending the reachability games to games that capture full HFL formulas including fixpoint quantifiers and variables is not easy either, see the example after the definition of the games below.

3.1. Fixpoint Elimination.

Lemma 3.1. *For all* HFL *types τ and all transition systems $\mathcal{T}$ with n states we have: $|\tau| \leq \mathbf{2}_{ord(\tau)+1}^{n \cdot (mar(\tau)+ord(\tau))^{ord(\tau)}}$.*

Proof. We prove this by induction on the structure of τ. Note that there are 2^n many different elements of type Pr, and $ord(\mathsf{Pr}) = 0 = mar(\mathsf{Pr})$ which immediately yields the base case.

For the other cases let $\tau = \tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}$. With uncurrying it is easy to regard this as a function that takes m arguments of corresponding type and delivers something of type Pr. Then we have

$$\begin{aligned}
|\tau| &= |\tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}| = |\mathsf{Pr}|^{\prod_{i=1}^m |\tau_i|} = 2^{n \cdot \prod_{i=1}^m |\tau_i|} \\
&= 2^{n \cdot \prod_{i=1}^m \mathbf{2}_{ord(\tau_i)+1}^{n \cdot (mar(\tau_i)+ord(\tau_i))^{ord(\tau_i)}}} && \text{by the hypothesis} \\
&\leq 2^{n \cdot \prod_{i=1}^m \mathbf{2}_{ord(\tau)}^{n \cdot (mar(\tau_i)+ord(\tau)-1)^{ord(\tau)-1}}} && \text{because of } ord(\tau_i) \leq ord(\tau) - 1 \\
&\leq 2^{n \cdot \prod_{i=1}^m \mathbf{2}_{ord(\tau)}^{n \cdot (mar(\tau)+ord(\tau)-1)^{ord(\tau)-1}}} && \text{because of } mar(\tau_i) \leq mar(\tau) \\
&= 2^{n \cdot (\mathbf{2}_{ord(\tau)}^{n \cdot (mar(\tau)+ord(\tau)-1)^{ord(\tau)-1}})^m} \\
&= 2^{n \cdot 2^{m \cdot \mathbf{2}_{ord(\tau)-1}^{n \cdot (mar(\tau)+ord(\tau)-1)^{ord(\tau)-1}}}} && \text{because of } ord(\tau) \geq 1 \\
&\leq 2^{n \cdot 2^{\mathbf{2}_{ord(\tau)-1}^{n \cdot m \cdot (mar(\tau)+ord(\tau)-1)^{ord(\tau)-1}}}} \\
&\leq 2^{n \cdot 2^{\mathbf{2}_{ord(\tau)-1}^{n \cdot (mar(\tau)+ord(\tau)-1)^{ord(\tau)}}}} && \text{because of } m \leq mar(\tau), ord(\tau) \geq 1 \\
&= 2^{n \cdot \mathbf{2}_{ord(\tau)}^{n \cdot (mar(\tau)+ord(\tau)-1)^{ord(\tau)}}} \\
&\leq 2^{\mathbf{2}_{ord(\tau)}^{n \cdot (mar(\tau)+ord(\tau)-1)^{ord(\tau)} + \log n}} && \text{because of } ord(\tau) \geq 1 \\
&\leq 2^{\mathbf{2}_{ord(\tau)}^{n \cdot ((mar(\tau)+ord(\tau)-1)^{ord(\tau)}+1)}} \\
&\leq 2^{\mathbf{2}_{ord(\tau)}^{n \cdot (mar(\tau)+ord(\tau))^{ord(\tau)}}} && \text{because of } ord(\tau) \geq 1 \\
&= \mathbf{2}_{ord(\tau)+1}^{n \cdot (mar(\tau)+ord(\tau))^{ord(\tau)}}
\end{aligned}$$

which proves the claim. □

Let $types(k, m) := \{\tau \mid ord(\tau) \leq k, mar(\tau) \leq m\}$ denote the set of types of restricted order and maximal arity. As mentioned above we have $|types(0, 0)| = 1$, and $|types(k, 0)| = |types(0, m)| = 0$ for any $k, m \geq 1$.

Lemma 3.2. *For all $k \geq 1$ and $m \geq 1$ we have $|types(k, m)| \leq m^{k \cdot (m^{k-1})}$.*

Proof. By induction on k. First consider the case of $k = 1$. All types of order 1 and maximal arity m are of the form

$$\tau := \underbrace{\mathsf{Pr} \to \ldots \to \mathsf{Pr}}_{i \text{ times}} \to \mathsf{Pr}$$

with $1 \leq i \leq m$. Clearly, their number is bounded by $m = m^{1 \cdot (m^0)}$.

Now consider any $k > 1$. Remember that any HFL type is isomorphic to one of the form $\tau = \tau_1 \to \ldots \to \tau_i \to \mathsf{Pr}$. Note that $ord(\tau_j) < ord(\tau)$ for all $j = 1, \ldots, i$, and $1 \leq i \leq m$. Then we have

$$|types(k, m)| = \sum_{i=1}^{m} |types(k-1, i)|^i \leq m \cdot |types(k-1, m)|^m \leq m \cdot (m^{(k-1)m^{k-2}})^m$$

$$= \; m \cdot m^{(k-1)\cdot m\cdot m^{k-2}} \; = \; m \cdot m^{(k-1)\cdot m^{k-1}} \; = \; m^{(k-1)\cdot m^{k-1}+1} \; \leq \; m^{k\cdot m^{k-1}}$$

using the hypothesis for $k-1$. □

Lemma 3.3. *For any $k \geq 1$ and any $m \geq 1$ there are at most $m^{k\cdot(m^{k-1})} \cdot \mathbf{2}_{k+1}^{n\cdot(k+m)^k}$ many different functions f of type τ with $ord(\tau) \leq k$ and $mar(\tau) \leq m$ over a transition system with n states.*

Proof. Immediately from Lemmas 3.1 and 3.2. □

Definition 3.4. Let τ be any HFL type. We write $h(\tau)$ for the *height* of the lattice $[\![\tau]\!]^{\mathcal{T}}$ over a fixed transition system $\mathcal{T}$. It is the length of a maximal chain

$$f_0 \;\sqsubset_\tau\; f_1 \;\sqsubset_\tau\; f_2 \;\sqsubset_\tau\; \ldots$$

of elements that are properly increasing w.r.t. $\sqsubseteq_\tau$. In general this is an ordinal number, but if $|\mathcal{T}| < \infty$ then $h(\tau) \in \mathbb{N}$ for all HFL types τ.

Lemma 3.5. *For all* HFL *types τ and all transition systems $\mathcal{T}$ with n states we have: $h(\tau) \leq (n+1)(\mathbf{2}_{ord(\tau)}^{n(mar(\tau)+ord(\tau)-1)^{ord(\tau)-1}})^{mar(\tau)}$.*

Proof. First consider the case of $ord(\tau) = 0$. Then $\tau = \mathsf{Pr}$, and it is well-known that the power set lattice of n elements has height $n+1 = (n+1)\cdot(\mathbf{2}_0^{n(0+0-1)^{0-1}})^0$.

Now suppose $ord(\tau) > 0$ and $\tau = \tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}$ for some $m \geq 1$. Let $N := (\mathbf{2}_{ord(\tau)}^{n(mar(\tau)+ord(\tau)-1)^{ord(\tau)-1}})^{mar(\tau)}$. According to Lemma 3.1 there are at most N many different tuples $\bar{x} \in [\![\tau_1]\!]^{\mathcal{T}} \times \ldots \times [\![\tau_m]\!]^{\mathcal{T}}$ because $ord(\tau_i) \leq ord(\tau) - 1$ for all $i = 1, \ldots, m$.

Using uncurrying we can regard each $f \in [\![\tau]\!]^{\mathcal{T}}$ as a function that maps each such $\bar{x}$ to an element of Pr. Now suppose the claim is wrong. Then there is a chain

$$f_0 \;\sqsubset_\tau\; f_1 \;\sqsubset_\tau\; f_2 \;\sqsubset_\tau\; \ldots \;\sqsubset_\tau\; f_{(n+1)N+1}$$

of functions of type τ. Since each one is strictly greater than the preceeding one there is a sequence $\bar{x}_i$ of tuples s.t. for $i = 0, \ldots, (n+1)N$ we have $f_i\ \bar{x}_i \subsetneq f_{i+1}\ \bar{x}_i$. But remember that there are only N tuples altogether. By the pidgeon hole principle, one of them must occur at least $(n+1)+1$ many times. Thus, there are $0 \leq i_1 < \ldots < i_{n+2} \leq (n+1)N+1$ s.t. $\bar{x}_{i_1} = \bar{x}_{i_2} = \ldots = \bar{x}_{i_{n+2}}$. Let $\bar{x}$ simply denote this element.

By transitivity of the partial order $\sqsubseteq_\tau$ we then have

$$f_{i_1}\ \bar{x} \;\subsetneq\; f_{i_2}\ \bar{x} \;\subsetneq\; \ldots \;\subsetneq\; f_{i_{n+2}}\ \bar{x}$$

which contradicts the fact that the height of Pr is only $n+1$. Hence, the height of τ must be bounded by $(n+1)N$. □

Definition 3.6. Let $\mu X.\varphi$ be an HFL formula of type $\tau = \tau_1 \to \ldots \to \tau_m \to \mathsf{Pr}$. We define finite approximants of this fixpoint formula for all $\alpha \in \mathbb{N}$ as follows:

$$\mu^0 X.\varphi \;:=\; \lambda(Z_1 : \tau_1^0)\ldots\lambda(Z_m : \tau_m^0).\mathtt{ff} \quad , \qquad \mu^{\alpha+1}X.\varphi \;:=\; \varphi[\mu^\alpha X.\varphi/X]$$

The next result is an immediate consequence of the Knaster-Tarski theorem [27].

Lemma 3.7. *Let $\mathcal{T}$ be a transition system with state set $\mathcal{S}$ s.t. $|\mathcal{S}| < \infty$. For all* HFL *formulas $\mu(X:\tau).\varphi$ and all environments ρ we have: $[\![\mu(X:\tau).\varphi]\!]_\rho^{\mathcal{T}} = [\![\mu^{h(\tau)}X.\varphi]\!]_\rho^{\mathcal{T}}$.*

The following lemma concerns the size of formulas after fixpoint elimination.

Lemma 3.8. *Let $\mathcal{T}$ be a finite transition system with n states, $k, m \geq 1$. For every closed $\mathrm{HFL}^{k,m}$ formula φ there is a fixpoint-free and closed $\varphi' \in \mathrm{HFL}^{k,m}$ s.t. $[\![\varphi]\!]^{\mathcal{T}} = [\![\varphi']\!]^{\mathcal{T}}$, $v(\varphi') \leq v(\varphi) + m$, and $|\varphi'| \leq |\varphi| \cdot (n+1)^{|\varphi|} \cdot (\mathbf{2}_k^{n(m+k-1)^{k-1}})^{m \cdot |\varphi|}$.*

Proof. First we prove the existence of such a φ' by induction on the number f of different fixpoint subformulas of φ. If this is 0 then simply take $\varphi' := \varphi$.

Suppose $f > 0$. Then φ contains at least one subformula $\mu(X:\tau).\psi$ of some type τ s.t. ψ is fixpoint-free. According to Lemma 3.7 this $\mu(X:\tau).\psi$ is equivalent to $\mu^{h(\tau)}X.\psi$ over $\mathcal{T}$. Furthermore, $\mu^{h(\tau)}X.\psi$ is fixpoint-free. Let $\varphi'' := \varphi[\mu^{h(\tau)}X.\psi / \mu(X:\tau).\psi]$. Since φ'' contains less fixpoint subformulas as φ we can use the induction hypothesis to obtain a φ' that is equivalent to φ'' over $\mathcal{T}$. Lemma 3.7 shows that φ'' is equivalent to φ over $\mathcal{T}$, hence we have $[\![\varphi]\!]^{\mathcal{T}} = [\![\varphi']\!]^{\mathcal{T}}$. Note that fixpoint elimination does not create free variables, i.e. φ' is also closed.

What remains to be shown are the corresponding bounds on the size and number of variables of φ'. First consider $v(\varphi')$. The only λ-bound variables in φ' are those that are already λ-bound in φ plus at most m variables for subformulas of the form $\mu^0 X.\psi = \lambda Z_1 \ldots \lambda Z_{m'}.\mathtt{ff}$ for some $m' \leq m$. Note that the approximants reuse λ-bound variables which is semantically sound because the value of an i-th approximant as a function cannot depend on an argument of the j-th approximant for some $j \neq i$. The only free variables in each approximant should be those that are free in $\mu(X:\tau).\psi$ already.

Finally, let $N := (n+1)(\mathbf{2}_k^{n(m+k-1)^{k-1}})^m$. We show by induction on the number f of fixpoint subformulas in φ that the size of φ' is bounded by $(N+1)^f \cdot |\varphi|$. It should be clear that this implies the claim of the lemma.

This is clearly true for $f = 0$. Now let $f > 0$, and first consider the formula $\varphi'' = \varphi[\mu^{h(\tau)}X.\psi / \mu(X:\tau).\psi]$ as constructed above. Note that $ord(\tau) \leq k$, and $mar(\tau) \leq m$, and, according to Lemma 3.5, $h(\tau) \leq N$. Therefore, we can estimate the size of the approximant that replaces the fixpoint formula as $|\mu^{h(\tau)}X.\psi| \leq N \cdot |\psi| + m + 1$. This is because the size of the 0-th approximant is $m+1$ and the size of the $(i+1)$-st is always $|\psi|$ plus the size of the i-th. Then we have

$$|\varphi''| \;=\; |\varphi| - |\psi| + N \cdot |\psi| + m + 1 \;=\; |\varphi| + (N-1) \cdot |\psi| + m + 1 \;\leq\; N \cdot |\varphi| + m + 1 \;\leq\; (N+1) \cdot |\varphi|$$

because the size of a formula φ must be strictly greater than the maximal arity of any of its subformulas. Now the number of fixpoint formulas in φ'' is $f - 1$. By the induction hypothesis we obtain

$$|\varphi'| \;\leq\; (N+1)^{f-1} \cdot |\varphi''| \;\leq\; (N+1)^{f-1} \cdot (N+1) \cdot |\varphi| \;=\; (N+1)^f \cdot |\varphi|$$

for the size of the formula φ' without any fixpoint subformulas. □

3.2. **Reachability Games.**

Definition 3.9. A *reachability game* between players $\exists$ and $\forall$ is a pointed and directed graph $\mathcal{G} = (V_\exists, V_\forall, E, v_0, W_\exists, W_\forall)$ with node set $V := V_\exists \cup V_\forall \cup W_\exists \cup W_\forall$ for some mutually disjoint $V_\exists, V_\forall, W_\exists, W_\forall$, edge relation $E \subseteq (V_\exists \cup V_\forall) \times V$ and designated starting node $v_0 \in V$. Define $|\mathcal{G}| := |E|$ as the size of the game.

The sets $V_\exists$ and $V_\forall$ contain those nodes in which player $\exists$, resp. player $\forall$ makes a choice. The sets $W_\exists$ and $W_\forall$ are terminal nodes in which player $\exists$, resp. player $\forall$ wins. We therefore

require that only nodes in $W_\exists$ or $W_\forall$ are terminal, i.e. for all $v \in V \setminus (W_\exists \cup W_\forall)$ there is a $w \in V$ with $(v, w) \in E$.

A *play* is a sequence $v_0, v_1, \ldots$ starting in v_0 and constructed as follows. If the play has visited nodes $v_0, \ldots, v_i$ for some $i \in \mathbb{N}$ and $v_i \in V_p$ for some $p \in \{\exists, \forall\}$ then player p chooses a node w s.t. $(v_i, w) \in E$ and $v_{i+1} := w$.

A play $v_0, \ldots, v_n$ is won by player p if $v_n \in W_p$. A reachability game is called *determined* if every play has a unique winner. Given the prerequisite $W_\exists \cap W_\forall = \emptyset$, determinacy of a reachability game simply means that infinite plays are not possible.

A strategy[3] for player p is a function $\sigma : V_p \to V$. A play $v_0, \ldots, v_n$ *conforms* to a strategy σ for player p if for all $i = 0, \ldots, n-1$ with $v_i \in V_p$: $v_{i+1} = \sigma(v_i)$. Such a strategy σ is called *winning strategy* if player p wins every play that conforms to σ.

The problem of solving a determined reachability game is: given such a game $\mathcal{G}$, decide whether or not player $\exists$ has a winning strategy for $\mathcal{G}$.

It is well-known that reachability games can be solved in linear time using dynamic programming for instance [30].

Theorem 3.10. *Solving a reachability game $\mathcal{G}$ can be done in time $O(|\mathcal{G}|)$.*

3.3. **Model Checking Games for Fixpoint-Free HFL.** In this section we define reachability games that capture exactly the satisfaction relation for fixpoint-free HFL formulas.

Let φ_0 be a closed and fixpoint-free HFL formula of type Pr and $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ a labeled transition system with a designated starting state $s_0 \in \mathcal{S}$. The game $\mathcal{G}_\mathcal{T}(s_0, \varphi_0)$ is played between players $\exists$ and $\forall$ in order to determine whether or not $\mathcal{T}, s_0 \models \varphi_0$ holds. A configuration of the game is written

$$s, f_1, \ldots, f_k, \eta \vdash \psi$$

s.t. $\psi \in FL(\varphi_0)$ is of some type $\tau_1 \to \ldots \to \tau_k \to \mathsf{Pr}$, $s \in \mathcal{S}$, and $f_i \in [\![\tau_i]\!]^\mathcal{T}$ for all $i = 1, \ldots, k$. Note that $k = 0$ is possible. Finally, η is a (partial) finite map that assigns an element $f \in [\![\tau]\!]^\mathcal{T}$ to each free variable X of type τ in ψ .

The intended meaning of such a configuration is: player $\exists$ tries to show $s \in [\![\psi]\!]^\mathcal{T}_\eta \; f_1 \ldots f_n$ whereas player $\forall$ tries to show the opposite. Since the semantics of formulas is defined recursively, the play usually proceeds from one such configuration to another containing a direct subformula. For instance, if the formula in the current configuration is a disjunction then player $\exists$ chooses one of the disjuncts because disjunctions are easy to prove but hard to refute in this way. Consequently, player $\forall$ performs a choice on conjunctions (negated disjunctions). A similar argument applies to configurations with modal operators. In case of function application we employ a small protocol of choices between these two players which simply reflects the semantics of function application in higher-order logic, etc.

A *play* of $\mathcal{G}_\mathcal{T}(s_0, \varphi_0)$ is a finite sequence $C_0, C_1, \ldots$ of configurations constructed as follows. $C_0 := s_0, \eta_0 \vdash \varphi_0$ where η_0 is undefined on all arguments.

If $C_0, \ldots, C_{n-1}$ have already been constructed, then C_n is obtained by case distinction on C_{n-1}.

(1) If $C_{n-1} = s, \eta \vdash \psi_1 \vee \psi_2$ then player $\exists$ chooses an $i \in \{1, 2\}$ and $C_n := s, \eta \vdash \psi_i$.
(2) If $C_{n-1} = s, \eta \vdash \neg(\psi_1 \vee \psi_2)$ then player $\forall$ chooses an $i \in \{1, 2\}$ and $C_n := s, \eta \vdash \neg\psi_i$.

[3]Here we restrict ourselves to memory-less strategies which are well-known to suffice for reachability games.

(3) If $C_{n-1} = s, \eta \vdash \langle a \rangle \psi$ then player $\exists$ chooses a $t \in \mathcal{S}$ s.t. $s \xrightarrow{a} t$ and $C_n := t, \eta \vdash \psi$.
(4) If $C_{n-1} = s, \eta \vdash \neg\langle a \rangle \psi$ then player $\forall$ chooses a $t \in \mathcal{S}$ s.t. $s \xrightarrow{a} t$ and $C_n := t, \eta \vdash \neg\psi$.
(5) If $C_{n-1} = s, f_1, \ldots, f_k, \eta \vdash \neg\neg\psi$ then $C_n := s, f_1, \ldots, f_k, \eta \vdash \psi$.
(6) If $C_{n-1} = s, f_1, \ldots, f_k, \eta \vdash \varphi\ \psi$ and ψ is of type σ then player $\exists$ chooses a $g \in [\![\sigma]\!]^{\mathcal{T}}$. Next player $\forall$ has two options.
   - He either continues with $C_n := s, g, f_1, \ldots, f_k, \eta \vdash \varphi$.
   - Or let $\sigma = \sigma_1 \to \ldots \to \sigma_m \to \mathsf{Pr}$. Player $\forall$ chooses values $h_i \in [\![\sigma_i]\!]^{\mathcal{T}}$ for $i = 1, \ldots, m$, and either
     - selects a $t \in g\ h_1 \ldots h_m$, and the play continues with $t, h_1, \ldots, h_m, \eta \vdash \psi$, or
     - selects a $t \notin g\ h_1 \ldots h_m$, and the play continues with $t, h_1, \ldots, h_m, \eta \vdash \neg\psi$.
(7) If $C_{n-1} = s, f_1, \ldots, f_k, \eta \vdash \neg(\varphi\ \psi)$ and ψ is of type σ then player $\exists$ chooses a $g \in [\![\sigma]\!]^{\mathcal{T}}$. Next player $\forall$ has two options.
   - He either continues with $C_n := s, g, f_1, \ldots, f_k, \eta \vdash \neg\varphi$.
   - Or let $\sigma = \sigma_1 \to \ldots \to \sigma_m \to \mathsf{Pr}$. Player $\forall$ chooses values $h_i \in [\![\sigma_i]\!]^{\mathcal{T}}$ for $i = 1, \ldots, m$, and either
     - selects a $t \in g\ h_1 \ldots h_m$, and the play continues with $t, h_1, \ldots, h_m, \eta \vdash \psi$, or
     - selects a $t \notin g\ h_1 \ldots h_m$, and the play continues with $t, h_1, \ldots, h_m, \eta \vdash \neg\psi$.
(8) If $C_{n-1} = s, f_1, \ldots, f_k, \eta \vdash \lambda X.\psi$ then $C_n := s, f_2, \ldots, f_k, \eta[X \mapsto f_1] \vdash \psi$.
(9) If $C_{n-1} = s, f_1, \ldots, f_k, \eta \vdash \neg\lambda X.\psi$ then $C_n := s, f_2, \ldots, f_k, \eta[X \mapsto f_1] \vdash \neg\psi$.

The game rules (5),(8) and (9) are deterministic. Neither player has to make a real choice there.

A play $C_0, C_1, \ldots, C_n$ is won by player $\exists$, if

(1) $C_n = s, \eta \vdash q$ and $s \in L(q)$, or
(2) $C_n = s, \eta \vdash \neg q$ and $s \notin L(q)$, or
(3) $C_n = s, f_1, \ldots, f_k, \eta \vdash X$ and $s \in \eta(X)\ f_1 \ldots f_k$, or
(4) $C_n = s, f_1, \ldots, f_k, \eta \vdash \neg X$ and $s \notin \eta(X)\ f_1 \ldots f_k$, or
(5) $C_n = s, \eta \vdash \neg\langle a \rangle \psi$ and there is no $t \in \mathcal{S}$ with $s \xrightarrow{a} t$.

Player $\forall$ wins this play, if

(6) $C_n = s, \eta \vdash q$ and $s \notin L(q)$, or
(7) $C_n = s, \eta \vdash \neg q$ and $s \in L(q)$, or
(8) $C_n = s, f_1, \ldots, f_k, \eta \vdash X$ and $s \notin \eta(X)\ f_1 \ldots f_k$, or
(9) $C_n = s, f_1, \ldots, f_k, \eta \vdash \neg X$ and $s \in \eta(X)\ f_1 \ldots f_k$, or
(10) $C_n = s, \eta \vdash \langle a \rangle \psi$ and there is no $t \in \mathcal{S}$ with $s \xrightarrow{a} t$.

We remark that these games do not easily extend to formulas with fixpoint quantifiers and variables via the characterisation of the model checking problem for the modal μ-calculus as a parity game [26]. The natural extension would add simple unfolding rules for fixpoint constructs which lead to infinite plays. The type of the outermost fixpoint variable that gets unfolded infinitely often in such a play would determine the winner.

However, this is neither sound nor complete. Consider the formula $(\nu(X : \mathsf{Pr} \to \mathsf{Pr}).\mu(Y : \mathsf{Pr}).X\ Y)\ \mathtt{ff}$. It is equivalent to $\mathtt{tt}$, hence, player $\exists$ should have a winning strategy for the game on this formula and any transition system. But player $\forall$ can enforce a play via rule (6) in which the outermost variable that gets unfolded infinitely often is Y which is of type μ.

This shows that the straight-forward extension to non-fixpoint-free formulas is not complete. Because of the presence of negation it is also not sound. Another explanation for the failure of such games is given by the model checking games for FLC [17] which incorporate a stack and a visibly pushdown winning condition in order to model that the variable X (the function) is more important than the variable Y (the argument) in the example above.

Lemma 3.11. *Every play of $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ has a unique winner.*

Proof. All rules properly reduce the size of the formula component in a configuration. Hence, there are no infinite plays, and a play is finished when either one of the players cannot perform a choice or there is no rule that applies to the current configuration anymore.

Note that for as long as rules still apply there are only two situations in which a player can get stuck: Either the current configuration is $s, \eta \vdash \langle a \rangle \psi$ or it is $s, \eta \vdash \neg\langle a \rangle \psi$ and there is no $t \in \mathcal{S}$ s.t. $s \xrightarrow{a} t$. These cases are covered by winning conditions (5) and (10).

All other rules always guarantee one player a possible choice. The only rules for which this is not obvious are (6) and (7). First note that $[\![\sigma]\!]^{\mathcal{T}}$ is non-empty for any type σ. Hence, player $\exists$ can always choose some g. Then let, for some arguments $h_1, \ldots, h_m$ chosen by player $\forall$, $T := g\ h_1 \ldots h_m$. Note that it is impossible to have $T = \emptyset$ and at the same time $\mathcal{S} \setminus T = \emptyset$ for as long as $\mathcal{S} \neq \emptyset$ for the underlying state space $\mathcal{S}$. Hence, player $\forall$ cannot get stuck in this rule either.

If a play finishes because no rule applies then the formula in the current configuration must either be atomic or a negation of an atomic formula, i.e. of one of the forms $q, \neg q, X, \neg X$ for some $q \in \mathcal{P}$, $X \in \mathcal{V}$. In any case, one of the winning conditions (1)–(4) and (6)–(9) applies.

This shows that every play has at least one winner. Finally, it is not hard to see that the winning conditions are mutually exclusive, i.e. every play has at most one winner. □

Theorem 3.12. *Let φ_0 be closed, fixpoint-free, and of type* Pr*. If $s_0 \in [\![\varphi_0]\!]^{\mathcal{T}}$ then player $\exists$ has a winning strategy for the game $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$.*

Proof. We call a configuration $C = t, f_1, \ldots, f_k, \eta \vdash \psi$ of the game $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$ *true* if $t \in [\![\psi]\!]^{\mathcal{T}}_{\eta}\ f_1 \ldots f_k$. Otherwise we call C *false*.

Suppose $s_0 \in [\![\varphi_0]\!]$, i.e. the starting configuration $s_0, \eta_0 \vdash \varphi_0$ of $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$ is true. Player $\exists$'s strategy will consist of preservering truth along a play. We will show by case distinction on the last rule played that player $\exists$ can enforce a play in which every configuration is true. I.e. if a configuration that is true requires her to make a choice then she can choose a successor configuration which is also true. If such a configuration requires player $\forall$ to make a choice then regardless of what he selects, the successor will always be true.

Cases (1) and (2), the Boolean operators. If a play has reached a configuration $t, \eta \vdash \psi_1 \vee \psi_2$ that is true then there is an $i \in \{1, 2\}$ s.t. $t, \eta \vdash \psi_i$ is true. Player $\exists$ chooses this i. Note that player $\forall$ will ultimately preserve truth if he makes a choice in a configuration $t, \eta \vdash \neg(\psi_1 \vee \psi_2)$.

Cases (3) and (4), the modal operators. Similarly, player $\exists$ can preserve truth in a configuration of the form $t, \eta \vdash \langle a \rangle \psi$, and player $\forall$ must preserve truth in a configuration of the form $t, \eta \vdash \neg\langle a \rangle \psi$.

Case (5), double negation. Preservation of truth is trivial.

Case (6), positive application. Suppose the play has reached a configuration $t, f_1, \ldots, f_k, \eta \vdash \varphi\ \psi$ that is true. Let $g := [\![\psi]\!]^{\mathcal{T}}_{\eta}$. Note that g always exists, hence, player $\exists$ can choose it. By β-equivalence we have

$$t \in [\![\varphi\ \psi]\!]^{\mathcal{T}}_{\eta}\ f_1 \ldots f_k \ \Rightarrow\ t \in [\![\varphi]\!]^{\mathcal{T}}_{\eta}\ g\ f_1 \ldots f_k$$

which shows that truth is preserved if player $\forall$ selects his first option.

Suppose he selects his second option with arguments $h_1, \ldots, h_m$ for g instead. Since $g = [\![\psi]\!]^{\mathcal{T}}_{\eta}$ we obviously have for all $t \in \mathcal{S}$: $t \in g\ h_1 \ldots h_m$ iff $t \in [\![\psi]\!]^{\mathcal{T}}_{\eta}\ h_1 \ldots h_m$. This shows that truth is preserved regardless of which way player $\forall$ leads.

Case (7), negative application. This is the same as the case above. Note that – by the semantics of the negation operator – we have $\neg(\varphi\ \psi) \equiv (\neg\varphi)\ \psi$.

Cases (8) and (9), λ-abstraction. This is only an equivalence-preserving β-reduction. Hence, truth is preserved. For case (9) remember that the complement of a function is defined pointwise.

It remains to be seen that this truth-preserving strategy guarantees player $\exists$ to win any play. I.e. assume that player $\forall$ uses his best strategy against player $\exists$'s truth-preserving strategy and consider the unique play $C_0, \ldots, C_n$ that results from playing against each other. By the argumentation above, we know that C_i is true for all $i = 0, \ldots, n$. A quick inspection of player $\forall$'s winning conditions (6)–(10) shows that he cannot be the winner of this play because all of them require the play at hand to end in a configuration that is not true.

According to Lemma 3.11, player $\exists$ wins every play in which she uses the truth-preserving strategy. Hence, this is a winning strategy. □

Theorem 3.13. *Let φ_0 be closed, fixpoint-free, and of type* Pr. *If $s_0 \notin [\![\varphi_0]\!]^{\mathcal{T}}$ then player $\forall$ has a winning strategy for the game $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$.*

Proof. Similar to the proof of Theorem 3.12. The starting configuration of $\mathcal{G}_{\mathcal{T}}(s_0, \varphi_0)$ is false. An analysis of the game rules shows that player $\forall$ can preserve falsity with his choices, and player $\exists$ must preserve falsity.

This is shown for rules (1), (2), (3), (4) and (5) in the same way as above in the proof of Thm. 3.12. Here we only consider the case (6). The case of rule (7) is shown analogously.

Suppose the current configuration is $s, f_1, \ldots, f_k, \eta \vdash \varphi\ \psi$, s.t. ψ has type σ and player $\exists$ has chosen some $g \in [\![\sigma]\!]^{\mathcal{T}}$. We need to distinguish two subcases.

If $s \notin [\![\varphi]\!]^{\mathcal{T}}_{\eta}\ g\ f_1 \ldots f_k$ then player $\forall$ can easily preserve falsity by choosing the successor configuration $s, g, f_1, \ldots, f_k, \eta \vdash \varphi$.

If $s \in [\![\varphi]\!]^{\mathcal{T}}_{\eta}\ g\ f_1 \ldots f_k$ then we must have $g \neq_\sigma [\![\psi]\!]^{\mathcal{T}}_{\eta}$ for equality would, by β-equivalence, contradict the assumption that the current configuration is false. Remember that $\neq_\sigma$ is inequality w.r.t. the pointwise order $\sqsubseteq_\sigma$. Now suppose $\sigma = \sigma_1 \to \ldots \to \sigma_m \to$ Pr. Hence, there must be $h_i \in [\![\sigma_i]\!]^{\mathcal{T}}$ for $i = 1, \ldots, m$ s.t. $g\ h_1 \ldots h_m \neq [\![\psi]\!]^{\mathcal{T}}_{\eta}\ h_1 \ldots h_m$. First of all, player $\forall$ can choose these arguments $h_1, \ldots, h_m$. Next, let $T := g\ h_1 \ldots h_m$ and $T' := [\![\psi]\!]^{\mathcal{T}}_{\eta}\ h_1 \ldots h_m$. Note that $T, T' \subseteq \mathcal{S}$. Hence, $T \neq T'$ means $T \not\subseteq T'$ or $T' \not\subseteq T$.

In the first case there is a $t \in g\ h_1 \ldots h_m$ s.t. $t \notin [\![\psi]\!]^{\mathcal{T}}_{\eta}\ h_1 \ldots h_m$. Player $\forall$ can choose this t and continue with the configuration $t, h_1, \ldots, h_m, \eta \vdash \psi$ which is false.

In the second case note that $T' \not\subseteq T$ iff $\mathcal{S} \setminus T \not\subseteq \mathcal{S} \setminus T'$. Hence, there is a $t \notin g\ h_1 \dots h_m$ s.t. $t \notin \mathcal{S} \setminus ([\![\psi]\!]^{\mathcal{T}}_{\eta}\ h_1 \dots h_m)$. Again, player $\forall$ can choose this t and continue with the false configuration $t, h_1, \dots, h_m, \eta \vdash \neg\psi$.

The proof is finished just like the proof of Thm. 3.12. With this strategy, player $\forall$ can always enforce a play that ends in a false configuration, but player $\exists$ can only win plays that end in true configurations. □

Putting these two theorems together shows that these games correctly characterise the satisfaction relation for fixpoint-free HFL.

Corollary 3.14. *For all transition systems $\mathcal{T}$ all of their states s, and all fixpoint-free* HFL *formulas φ of type* Pr *we have: $\mathcal{T}, s \models \varphi$ iff player $\exists$ wins the game $\mathcal{G}_{\mathcal{T}}(s, \varphi)$.*

Lemma 3.15. *For any $k, m \geq 1$, any $\mathcal{T} = (\mathcal{S}, \{\xrightarrow{a} \mid a \in \mathcal{A}\}, L)$ with $|\mathcal{S}| = n$, any $s \in \mathcal{S}$, and any* $\mathrm{HFL}^{k,m}$ *formula φ, $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ is a reachability game of size at most*

$$4n^2 \cdot |\varphi|^2 \cdot (m^{(k-1)m^{k-2}} \cdot \mathbf{2}_k^{n(k-1+m)^{k-1}})^{2(m+v(\varphi))}\ .$$

Proof. It should be clear from the definition of the game that $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ can indeed be regarded as a reachability game $(V_\exists, V_\forall, v_0, W_\exists, W_\forall)$. Its node set $V := V_\exists \cup V_\forall \cup W_\exists \cup W_\forall$ consists of all possible configurations in $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ plus auxiliary configurations that represent the choices done by either player in rules (6) and (7) which require an alternating sequence of choices of fixed depth 3. However, this can at most double the number of nodes in comparison to the number of configurations.

The starting node v_0 is the starting configuration $s, \eta \vdash \varphi$ for an everywhere undefined η. The partition of the nodes is given by the definition of the game rules and winning conditions above: $V_\exists$, resp. $V_\forall$ are all those configurations that require player $\exists$, resp. $\forall$ to make a choice – including the auxiliary configurations for the choices in between rules. The edges of the game are simply given by the game rules. $W_\exists$, resp. $W_\forall$ are all those configurations that end a play according to one of the winning conditions. Lemma 3.11 shows that these games are determined.

What remains to be seen is that the size of $\mathcal{G}_{\mathcal{T}}(s, \varphi)$ is bounded accordingly. There are at most n different states $t \in \mathcal{S}$, and at most $|\varphi|$ many formulas $\psi \in FL(\varphi)$. The maximal width of a configuration, the parameter m' in $t, f_1, \dots, f_{m'}, \eta \vdash \psi$ is bounded by m since here ψ has a type of arity m'. According to Lemma 3.3 there are at most $m^{(k-1)m^{k-2}} \mathbf{2}_k^{n(k-1+m)^{k-1}}$ many different functions f_i of type order $k-1$. None of these can be of type order k because they only occur as arguments to formulas of strictly higher order. We simply define $m^{(k-1)m^{k-2}} := m$ if $k = 1$ rather than introducing max-operators in these terms.

Finally, we need to estimate the number of different environments η. These map at most each λ-bound variable X of type τ in φ to an element of $[\![\tau]\!]^{\mathcal{T}}$. Again, if X occurs bound in φ, then there is a $\lambda X.\psi \in FL(\varphi)$ of type σ, and we have $ord(\sigma) \geq ord(\tau) + 1$. Hence, there are at most $m^{(k-1)m^{k-2}} \mathbf{2}_k^{n(k-1+m)^{k-1}}$ many possible values for each such X, and thus at most $(m^{(k-1)m^{k-2}} \mathbf{2}_k^{n(k-1+m)^{k-1}})^{v(\varphi)}$ many different environments η.

Putting this together we obtain

$$2 \cdot n \cdot (m^{(k-1)m^{k-2}} \cdot \mathbf{2}_k^{n(k-1+m)^{k-1}})^m \cdot (m^{(k-1)m^{k-2}} \cdot \mathbf{2}_k^{n(k-1+m)^{k-1}})^{v(\varphi)} \cdot |\varphi| \ =$$
$$2n \cdot |\varphi| \cdot (m^{(k-1)m^{k-2}} \cdot \mathbf{2}_k^{n(k-1+m)^{k-1}})^{m+v(\varphi)}$$

as an upper bound on the number of nodes in $\mathcal{G}_{\mathcal{T}}(s,\varphi)$. The number of edges in this directed graph can of course be at most quadratic in the number of nodes which finishes the proof. □

3.4. **The Model Checking Complexity.**

Theorem 3.16. *The model checking problem on a transition system $\mathcal{T}$ of size n and an* $\mathrm{HFL}^{k,m}$ *formula φ can be solved in time $2^{O(|\varphi|^k \cdot \log(n\cdot|\varphi|))} \cdot (\mathbf{2}_k^{n(m+k-1)^{k-1}})^{O(|\varphi|^2)}$ for any $k, m \geq 1$.*

Proof. Let $\mathcal{S}$ be the state space of $\mathcal{T}$, $n := |\mathcal{S}|$, and $\varphi \in \mathrm{HFL}^{k,m}$ for some $k, m \geq 1$ be closed. According to Lemma 3.8 there is a fixpoint-free φ' s.t.

- $v(\varphi') \leq v(\varphi) + m$,
- $|\varphi'| \leq |\varphi| \cdot (n+1)^{|\varphi|} \cdot (\mathbf{2}_k^{n(m+k-1)^{k-1}})^{m\cdot|\varphi|}$, and
- for all $s \in \mathcal{S}$ we have $\mathcal{T}, s \models \varphi$ iff $\mathcal{T}, s \models \varphi'$.

Now take any $s \in \mathcal{S}$. Consider the reachability game $\mathcal{G}_{\mathcal{T}}(s,\varphi')$. According to Lemma 3.15 its size is at most

$$\begin{aligned} & 4n^2 \cdot |\varphi'|^2 \cdot (m^{(k-1)m^{k-2}} \cdot \mathbf{2}_k^{n(k-1+m)^{k-1}})^{2(m+v(\varphi'))} \\ = \; & 4n^2 \cdot (|\varphi| \cdot (n+1)^{|\varphi|} \cdot (\mathbf{2}_k^{n(m+k-1)^{k-1}})^{m\cdot|\varphi|})^2 \cdot (m^{(k-1)m^{k-2}} \cdot \mathbf{2}_k^{n(k-1+m)^{k-1}})^{2(2m+v(\varphi))} \end{aligned}$$

by replacing $|\varphi'|$ according to Lemma 3.8. This can be approximated from above by

$$\begin{aligned} & 4 \cdot (n+1)^{2|\varphi|+2} \cdot |\varphi|^{3|\varphi|^k+2} \cdot (\mathbf{2}_k^{n(m+k-1)^{k-1}})^{2|\varphi|^2+6|\varphi|} \\ = \; & n^{O(|\varphi|)} \cdot |\varphi|^{O(|\varphi|^k)} \cdot (\mathbf{2}_k^{n(m+k-1)^{k-1}})^{O(|\varphi|^2)}) \;\; = \;\; 2^{O(|\varphi|^k \cdot \log(n\cdot|\varphi|))} \cdot (\mathbf{2}_k^{n(m+k-1)^{k-1}})^{O(|\varphi|^2)} \end{aligned}$$

because $|\varphi|$ is an upper bound on m, k and $v(\varphi)$. By Cor. 3.14 we have $\mathcal{T}, s \models \varphi'$ iff player $\exists$ has a winning strategy for $\mathcal{G}_{\mathcal{T}}(s,\varphi')$. And by Thm. 3.10 the asymptotic time needed to solve this game equals its size. □

Corollary 3.17. *For any $k, m \geq 1$ the* $\mathrm{HFL}^{k,m}$ *model checking problem is in* kExpTime.

Corollary 3.18. *For any $k, m \geq 1$ the model checking problem for* $\mathrm{HFL}^{k,m}$ *on a fixed transition system is in* ExpTime.

Proof. If k, m and n are fixed constants then so is $\mathbf{2}_k^{n(k-1+m)^{k-1}}$. Hence, model checking in this case can be done in time $2^{O(|\varphi|^2+|\varphi|^k \cdot \log|\varphi|)}$. □

4. The Lower Bound

We will show that the upper bound in Cor. 3.17 is optimal by reducing the word problem for alternating space bounded Turing Machines to the model checking problem for HFL.

Let $F_0(p(n)) := 2^{p(n)}$ and $F_{k+1}(p(n)) := 2^{p(n)\cdot F_k(p(n))}$ for any polynomial $p(n)$. A simple induction shows $F_k(p(n)) \geq \mathbf{2}_{k+1}^{p(n)}$ for all $k, n \in \mathbb{N}$. Clearly, the space used by a $\mathbf{2}_k^{p(n)}$-space bounded Turing Machine is also bounded by $F_{k-1}(p(n))$ for $k \geq 1$. This slight shift in indices makes the encoding of large numbers in the next section easier. On the other hand, it only allows us to consider alternating $\mathbf{2}_k^{p(n)}$-space bounded Turing Machines when $k \geq 1$. Hence, we will only obtain kExpTime-hardness results for $k \geq 2$. Fortunately, the results for the $\mathrm{HFL}^{1,m}$ fragments follow from known lower bounds for FLC [18].

	$\|F_{k-1}(p(n))-1\|^{k-1}$	$\dots$	$\|1\|^{k-1}$	$\|0\|^{k-1}$
$\|0\|^k$	$\|0\|^0$	$\dots$	$\|0\|^0$	$\|0\|^0$
$\|1\|^k$	$\|0\|^0$	$\dots$	$\|0\|^0$	$\|1\|^0$
$\|2\|^k$	$\|0\|^0$	$\dots$	$\|0\|^0$	$\|2\|^0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\|F_1(p(n))-1\|^k$	$\|0\|^0$	$\dots$	$\|0\|^0$	$\|F_0(p(n))-1\|^0$
$\|F_1(p(n))\|^k$	$\|0\|^0$	$\dots$	$\|1\|^0$	$\|0\|^0$
$\|F_1(p(n))+1\|^k$	$\|0\|^0$	$\dots$	$\|1\|^0$	$\|1\|^0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\|F_k(p(n))-1\|^k$	$\|F_0(p(n))-1\|^0$	$\dots$	$\|F_0(p(n))-1\|^0$	$\|F_0(p(n))-1\|^0$

Figure 3: Encoding large numbers as lexicographically ordered functions.

4.1. **Representing Large Numbers in** HFL**.** Let $\tau_0 := \mathsf{Pr}$ and $\tau_{k+1} := \tau_k \to \mathsf{Pr}$ for all $k \in \mathbb{N}$. Note that on a transition system of exactly $p(n)$ states we have $|\tau_k| = F_k(p(n))$ for all $k \in \mathbb{N}$. In order to model the position of the head and the sequence of the cells of a tape of size $F_k(p(n))$ we therefore use a transition system $\mathcal{T}$ with $p(n)$ many states, and an encoding of the natural numbers $\{0, \dots, F_k(p(n)) - 1\}$ via HFL functions[4] of type τ_k over $\mathcal{T}$. This is done by induction on k. Let n and the polynomial $p(n)$ be fixed.

For $k = 0$ we assume that $\mathcal{T}$ contains $p(n)$ many states called $0, \dots, p(n)-1$. A number i between 0 and $F_0(p(n)) - 1$ is now represented by the subset $S_i = \{j \mid \text{the } j\text{-th bit of } i \text{ is } 1\}$ which has type τ_0. Let $\|i\|^0$ for $i \in \{0, \dots, F_0(p(n)) - 1\}$ denote the function of type τ_0 that represents the natural number i in this way.

Now let $k > 0$. By assumption there are HFL functions $\|0\|^{k-1}, \dots, \|F_{k-1}(p(n))-1\|^{k-1}$ of type τ_{k-1} that represent the numbers $0, \dots, F_{k-1}(p(n)) - 1$. Clearly, these are linearly ordered by the standard ordering on the numbers that they represent. We now need to find a representation of the numbers $0, \dots, F_k(p(n))-1$ via HFL functions of type $\tau_k = \tau_{k-1} \to \mathsf{Pr}$.

These functions have a finite and linearly ordered domain as well as co-domain. Hence, we can regard them as lexicographically ordered words of length $F_{k-1}(p(n))$ over the alphabet $\{\|0\|^0, \dots, \|F_0(p(n))-1\|^0\}$, or simply as base-$F_0(p(n))$ numerals with $F_{k-1}(p(n))$ digits. Now $\|i\|^k$ simply is the i-th function in this lexicographic ordering as depicted in Fig. 3. The leftmost column contains the symbolic name $\|i\|^k$ for the i-th function in that ordering. The upper row contains the ordered list of all possible arguments x for any such $\|i\|^k$ while the entries below denote the values $\|i\|^k\ x$.

[4]We will also use the term "function" for an object of type τ_0 which is a set strictly speaking, hence, a function of order 0.

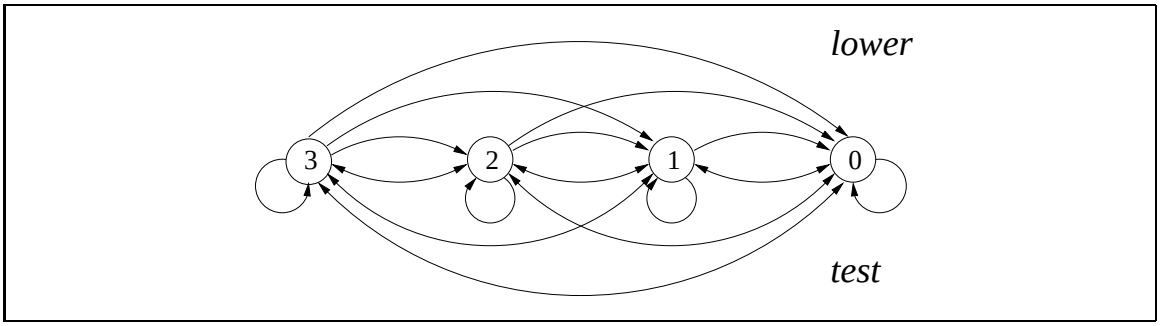

Figure 4: The transition system $\mathcal{T}_{\mathcal{M},w}$ for $p(n) = 4$.

4.2. **The Reduction.** For the remainder of this section we fix an alternating $F_k(p(n))$-space bounded Turing Machine $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, \delta, q_{acc}, q_{rej})$ and an input word w of length n. W.l.o.g. we assume $p(n) > n$ for all $n \in \mathbb{N}$. According to Thm. 2.12 we can also assume $\Sigma = \Gamma$ and $|\Gamma| = 2$.

Of course, symbols are just purely syntactic objects. However, later we need to encode these two symbols as propositions in transition systems, and we will use the propositions $\mathtt{tt}$ and $\mathtt{ff}$ to do so. Hence, we can simplify notation slightly by assuming $\Gamma = \{\mathtt{tt}, \mathtt{ff}\}$ as two different alphabet letters with no attached meaning. W.l.o.g. we assume that the special blank symbol $\square$ is encoded by a sequence of the symbol $\mathtt{ff}$ of some suitable length.

The goal is to construct a transition system $\mathcal{T}_{\mathcal{M},w}$ and an HFL^{k+2} formula $\Phi^k_{\mathcal{M},w}$ both of polynomial size, s.t. $\mathcal{T}_{\mathcal{M},w}, s \models \Phi^k_{\mathcal{M},w}$ iff $w \in L(\mathcal{M})$ for some state s. The types of the subformulas of $\Phi^k_{\mathcal{M},w}$ that we present in the following can easily be inferred. We will therefore omit type annotations.

We begin with the construction of the transition system. Let $\mathcal{P} := \emptyset$, i.e. no state of $\mathcal{T}_{\mathcal{M},w}$ carries a label. There are two modal accessibility relations with labels *lower* and *test*. Let $\mathcal{T}_{\mathcal{M},w} = (\mathcal{S}, \{\xrightarrow{lower}, \xrightarrow{test}\}, L)$ where $\mathcal{S} = \{0, \ldots, p(n) - 1\}$, and L maps every state to the empty set. The *lower*-relation simply resembles the less-than-relation on natural numbers: $i \xrightarrow{lower} j$ iff $j < i$. The *test*-relation forms a clique: $i \xrightarrow{test} j$ for all $i, j \in \mathcal{S}$. It is used to form global statements. Note that for all states i, j, and all formulas ψ: $i \models [test]\psi$ iff $j \models [test]\psi$. Fig. 4 depicts $\mathcal{T}_{\mathcal{M},w}$ for $p(n) = 4$. The transitions above the states are the $\xrightarrow{lower}$-relation. Consequently, they only lead from the left to the right. The transitions below the states are the $\xrightarrow{test}$-relation.

For the remainder of this section we fix $\mathcal{T}_{\mathcal{M},w}$ as the transition system over which formulas are interpreted and write $[\![\cdot]\!]$ instead of $[\![\cdot]\!]^{\mathcal{T}_{\mathcal{M},w}}$.

Remember that any function of type τ_0 represents a number in binary coding over $\mathcal{T}_{\mathcal{M},w}$: $\|i\|^0 = \{j \mid \text{the } j\text{-th bit of } i \text{ is } 1\}$. Furthermore, the transitions in $\mathcal{T}_{\mathcal{M},w}$ allow a bit to assess the values of all lower bits. The $\mathrm{HFL}^{1,1}$ formula

$$inc^0 \;:=\; \lambda X.X \leftrightarrow \langle lower\rangle \neg X$$

models the increment among the number representations $\|0\|^0, \ldots, \|F_0(p(n)) - 1\|^0$. Increment of a binary counter sets a bit of the input to 0 if itself and all lower bits are 1. A bit is set to 1 if it currently is 0 and all lower bits are 1. A bit is preserved if a lower-valued bit is unset. Applied to $\|F_0(p(n)) - 1\|^0$ this yields $\|0\|^0$ again. Similarly, we can model the

decrement among these values as

$$dec^0 \;:=\; \lambda X.X \leftrightarrow \langle lower\rangle X$$

Lemma 4.1. *For all $i \in \{0,\dots,F_0(p(n))-1\}$ we have:*

a) $[\![inc^0]\!]\,\|i\|^0 \;=\; \|i+1 \mod F_0(p(n))\|^0$,

b) $[\![dec^0]\!]\,\|i\|^0 \;=\; \|i-1 \mod F_0(p(n))\|^0$.

Proof. We will only show part (a) since part (b) is entirely analogous. Take any $i \in \{0,\dots,F_0(p(n))\}$ and let $m := p(n)-1 = (\log F_0(p(n)))-1$. Furthermore, let $b_m \dots b_0 \in \{0,1\}^{m+1}$ be the binary representation of the number i. According to the encoding described in the previous section, we have $\|i\|^0 = \{j \mid b_j = 1\}$.

Now take any state j and suppose $j \in [\![inc^0]\!]\,\|i\|^0$. The body of the λ-abstracted formula inc^0 is a bi-implication which can be seen as an abbreviation of a disjunction of two conjunctions. Hence, there are two possibilities.

- Either $j \models X \wedge \langle lower\rangle \neg X$ with X interpreted as $\|i\|^0$. This means that bit j is set in i and there is a lower bit that is not set in i. Hence, bit j is also set in $i+1 \mod F_0(p(n))$.
- Or $j \models \neg X \wedge [lower]X$ under the same interpretation of X. Then the j-th bit is the lowest bit which is unset in i. Hence, it gets set in $i+1 \mod F_0(p(n))$.

This shows that only those bits are included in the increment process that should be included. The converse direction – all necessary bits are included – is shown in the same way by case analysis. Suppose $j \notin [\![inc^0]\!]\,\|i\|^0$, i.e. $j \not\models X \leftrightarrow \langle lower\rangle \neg X$ with X interpreted by $\|i\|^0$.

- Either $j \models X \wedge [lower]X$. Then the j-th bit is among all those least bits that are set in i. Hence, it gets unset in $i+1 \mod F_0(p(n))$.
- Or $j \models \neg X \wedge \langle lower\rangle \neg X$. Then the j-th bit is not set in i, but there is a lower bit that is not set either. Hence, it preserves its value and remains unset in $i+1 \mod F_0(p(n))$.

□

In order to define the increment and decrement of numbers $\|i\|^k$ in lexicographic ordering for some $k > 0$ we need to have equality, less-than and greater-than tests on lower types τ_{k-1}. They can be implemented as functions of type $\tau_{k-1} \to \tau_{k-1} \to \mathsf{Pr}$. Equality simply makes use of the fact that two numbers are equal iff they have the same binary representation.

$$eq^0 \;:=\; \lambda I.\lambda J.[test](I \leftrightarrow J)$$

The other comparing functions need to access single bits. Remember that $i < j$ iff there is a bit that is unset in i and set in j s.t. i and j agree on all higher bits. We therefore first define formulas bit_i for $i = 0,\dots,p(n)-1$ s.t. bit_i axiomatises the state i, i.e. $j \models bit_i$ iff $i = j$. This can be done recursively as

$$bit_0 \;:=\; [lower]\mathtt{ff}\;, \qquad bit_{i+1} \;:=\; \langle lower\rangle bit_i \wedge [lower](\bigvee_{j\le i} bit_j)$$

Note that $|bit_i| = O(i^2)$ only.

$$lt^0 \;:=\; \lambda I.\lambda J.\;\bigvee_{k=0}^{p(n)-1} [test]\Big((bit_k \to \neg I \wedge J)) \wedge \bigwedge_{h>k} \big((bit_h \to (I \leftrightarrow J)\big)\Big)$$

$$gt^0 \;:=\; \lambda I.\lambda J.(lt^0\ J\ I)$$

Lemma 4.2. *For all $i, j \in \{0, \ldots, F_0(p(n)) - 1\}$ we have:*

a) $[\![eq^0]\!]\ \|i\|^0\ \|j\|^0 \;=\; \begin{cases} \mathcal{S}, & \text{if } i = j \\ \emptyset, & \text{o.w.} \end{cases}$

b) $[\![lt^0]\!]\ \|i\|^0\ \|j\|^0 \;=\; \begin{cases} \mathcal{S}, & \text{if } i < j \\ \emptyset, & \text{o.w.} \end{cases}$

c) $[\![gt^0]\!]\ \|i\|^0\ \|j\|^0 \;=\; \begin{cases} \mathcal{S}, & \text{if } i > j \\ \emptyset, & \text{o.w.} \end{cases}$

Proof. (a) The binary representation of a number is unique. Hence, $i = j$ iff $\|i\|^0 = \|j\|^0$ iff for all $s \in \mathcal{S}$: $s \in \|i\|^0 \Leftrightarrow s \in \|j\|^0$. The rest follows from the fact that $[test](I \leftrightarrow J)$ either holds in all states or in none of $\mathcal{S}$.

(b) Similarly, we have $i < j$ iff there is a bit that is unset in $\|i\|^0$ but set in $\|j\|^0$, and $\|i\|^0$ and $\|j\|^0$ agree on all higher bits. Again, each disjunct of the form $[test]\ldots$ is satisfied by either all or no states, and so is the entire disjunction.

(c) Follows directly from (b). □

We will call an HFL formula ψ of some type $\sigma_1 \to \ldots \to \sigma_m \to \mathsf{Pr}$ *2-valued* iff for all $x_1 \in [\![\sigma_1]\!], \ldots, x_m \in [\![\sigma_m]\!]$ we have $[\![\psi]\!]\ x_1 \ldots x_m = \mathcal{S}$ or $[\![\psi]\!]\ x_1 \ldots x_m = \emptyset$. For example, eq^0, lt^0, and gt^0 are 2-valued. Such functions will be used to model predicates, i.e. functions whose return value should be either *true* or *false*.

Before we can extend the incrementation and decrementation functions to types τ_k for some $k > 0$ we need to define some auxiliary functions and macros.

For HFL formulas β, ψ_1, ψ_2 of type Pr let

$$\texttt{if}\ \beta\ \texttt{then}\ \psi_1\ \texttt{else}\ \psi_2 \;:=\; (\beta \wedge \psi_1) \vee (\neg\beta \wedge \psi_2)$$

Note that if $[\![\beta]\!]$ is either $\mathcal{S}$ or $\emptyset$ then we have

$$[\![\texttt{if}\ \beta\ \texttt{then}\ \psi_1\ \texttt{else}\ \psi_2]\!] \;=\; \begin{cases} [\![\psi_1]\!], & \text{if } [\![\beta]\!] = \mathcal{S} \\ [\![\psi_2]\!], & \text{if } [\![\beta]\!] = \emptyset \end{cases}$$

For any $k \in \mathbb{N}$ we can easily define formulas min^k and max^k that encode the minimal and maximal element in the range of $0, \ldots, F_k(p(n)) - 1$.

$$min^0 \;:=\; \texttt{ff} \qquad\qquad max^0 \;:=\; \texttt{tt}$$

$$min^{k+1} \;:=\; \lambda X.min^0 \qquad\qquad max^{k+1} \;:=\; \lambda X.max^0$$

We will define by simultaneous induction on k the following formulas.

- $exists^k : (\tau_k \to \mathsf{Pr}) \to \mathsf{Pr}$
  It takes a predicate P on the number representations $\|0\|^k, \ldots, \|F_k(p(n)) - 1)\|^k$ and decides whether or not there is an i s.t. $P\ \|i\|^k$ holds. If P is 2-valued then so is $exists^k$. It is defined as

$$exists^k \;:=\; \lambda P.\Big(\big(\mu Z.\lambda X.(P\ X) \vee Z\ (inc^k\ X)\big)\ min^k\Big)$$

- $\mathit{forall}^k : (\tau_k \to \mathsf{Pr}) \to \mathsf{Pr}$
  Similarly, this function checks whether $P\ \|i\|^k$ holds for all such i.
  $$\mathit{forall}^k \ := \lambda P.\neg\big((\mathit{exists}^k)\ (\neg P)\big)$$
- $\mathit{eq}^k : \tau_k \to \tau_k \to \mathsf{Pr}$
  This is a 2-valued function which decides whether two given representations from $\|0\|^k, \ldots, \|F_k(p(n)) - 1\|^k$ encode the same number. Note that for $k = 0$ this has already been defined above.
  $$\mathit{eq}^k \ := \ \lambda I.\lambda J.\mathit{forall}^{k-1}\ \big(\lambda X.\mathit{eq}^0\ (I\ X)\ (J\ X)\big)$$
- $\mathit{lt}^k : \tau_k \to \tau_k \to \mathsf{Pr}$
  This 2-valued function decides for two number representations whether the less-than-relationship holds between the two encoded numbers. Again, the case of $k = 0$ has been dealt with above.
  $$\mathit{lt}^k \ := \ \lambda I.\lambda J.\mathit{exists}^{k-1}\ \Big(\lambda X.\big(\mathit{lt}^0\ (I\ X)\ (J\ X)\big)\ \wedge \mathit{forall}^{k-1}\ \Big(\lambda Y.(\mathit{gt}^{k-1}\ X\ Y) \to \big(\mathit{eq}^0\ (I\ X)\ (J\ Y)\big)\Big)\Big)$$
- $\mathit{gt}^k : \tau_k \to \tau_k \to \mathsf{Pr}$
  Using the last one we can easily decide for two number representations whether the greater-than-relationship holds between the two encoded numbers.
  $$\mathit{gt}^k \ := \ \lambda I.\lambda J.(\mathit{lt}^k\ J\ I)$$
- $\mathit{inc}^k : \tau_k \to \tau_k$
  This function models increment in the range of $0, \ldots, F_k(p(n)) - 1$ for $k > 0$.
  $$\mathit{inc}^k \ := \ \lambda I.\lambda X.\texttt{if}\ \ \mathit{exists}^{k-1}\ \big(\lambda Y.(\mathit{lt}^{k-1}\ Y\ X) \wedge \neg(\mathit{eq}^0\ (I\ Y)\ \mathit{max}^0)\big)\ \ \texttt{then}\ I\ X\ \ \texttt{else}\ \mathit{inc}^0\ (I\ X)$$
  Incrementation is done in the same way as with the binary represenation in the case of $k = 0$ above: inc^k applied to $\|i\|^k$ yields the function that agrees with $\|i\|^k$ on all arguments for which there is a smaller one whose value is not maximal, i.e. still less than $F_0(p(n)) - 1$. If all smaller arguments including itself have already reached the maximal value then they are reset to the minimal, i.e. $\|0\|^0$. Note that inc^0 also models the increment modulo $F_0(p(n))$.
- $\mathit{dec}^k : \tau_k \to \tau_k$
  Similarly, this models decrement in the range of $0, \ldots, F_k(p(n)) - 1$ for $k > 0$.
  $$\mathit{dec}^k \ := \ \lambda I.\lambda X.\texttt{if}\ \ \mathit{exists}^{k-1}\ \big(\lambda Y.(\mathit{lt}^{k-1}\ Y\ X) \wedge \neg(\mathit{eq}^0\ (I\ Y)\ \mathit{min}^0)\big)\ \ \texttt{then}\ I\ X\ \ \texttt{else}\ \mathit{dec}^0\ (I\ X)$$

These definitions are well-defined. For $k > 0$, inc^k and dec^k need lt^{k-1}, and exists^{k-1}. The latter only needs inc^{k-1}. The former needs exists^{k-2} and forall^{k-2}, etc.

Remark 4.3. For all $k \in \mathbb{N}$ we have:

$$\begin{array}{lllllll}
ord(exists^k) & = & ord(forall^k) & = & & & k+2 \\
ord(eq^k) & = & ord(lt^k) & = & ord(gt^k) & = & k+1 \\
ord(inc^k) & = & ord(dec^k) & = & & & k+1 \\
mar(exists^k) & = & mar(forall^k) & = & & & 1 \\
mar(eq^k) & = & mar(lt^k) & = & mar(gt^k) & = & 2 \\
mar(inc^k) & = & mar(dec^k) & = & & & 2
\end{array}$$

The following lemmas provide exact specifications for the functions above and prove that their implementations comply to these specifications. They are all proved by simultaneous induction on k.

Lemma 4.4. *For any function ψ of type $\tau_k \to \mathsf{Pr}$ s.t. $[\![\psi]\!]$ is two-valued we have:*

$$a) \quad [\![exists^k\ \psi]\!] = \begin{cases} \mathcal{S}, & \text{if } \exists i \in \{0,\ldots,F_k(p(n))-1\} \text{ s.t. } [\![\psi]\!]\ \|i\|^k = \mathcal{S} \\ \emptyset, & \text{o.w.} \end{cases}$$

$$b) \quad [\![forall^k\ \psi]\!] = \begin{cases} \mathcal{S}, & \text{if } \forall i \in \{0,\ldots,F_k(p(n))-1\} \text{ s.t. } [\![\psi]\!]\ \|i\|^k = \mathcal{S} \\ \emptyset, & \text{o.w.} \end{cases}$$

Proof. We will only prove part (a), since (b) follows from it by simple propositional reasoning. Note that for any formula ψ we have

$$exists^k\ \psi \ \equiv \bigvee_{i\in\mathbb{N}} p\ (\underbrace{inc^k(inc^k(\ldots(inc^k}_{i \text{ times}}\ min^k)\ldots)))$$

by fixpoint unfolding. The rest follows from the correctness Lemmas 4.1 and 4.6 for inc^k and the fact that p is assumed to be 2-valued. Clearly, the disjunction over disjuncts that all are either true or false or false. □

Lemma 4.5. *For all $k > 0$ and $i, j \in \{0,\ldots,F_k(p(n))-1\}$ we have:*

$$\text{a)} \ [\![eq^k]\!]\ \|i\|^k\ \|j\|^k = \begin{cases} \mathcal{S}, & \text{if } i = j \\ \emptyset, & \text{o.w.} \end{cases}$$

$$\text{b)} \ [\![lt^k]\!]\ \|i\|^k\ \|j\|^k = \begin{cases} \mathcal{S}, & \text{if } i < j \\ \emptyset, & \text{o.w.} \end{cases}$$

$$\text{c)} \ [\![gt^k]\!]\ \|i\|^k\ \|j\|^k = \begin{cases} \mathcal{S}, & \text{if } i > j \\ \emptyset, & \text{o.w.} \end{cases}$$

Proof. (a) This follows immediately from the definition of eq^k and Lemmas 4.4 and 4.2. Note that $i = j$ iff they encode the same functions according to the representation of the previous section. Function equality, however, can easily be tested using the *forall* macro to iterate through all possible arguments and the eq^0 function to compare the corresponding values.

(b) We have $i < j$ iff $\|i\|^k$ is lexicographically smaller than $\|j\|^k$ according to the encoding of the previous section. Now this is the case iff there is an argument x s.t. the value of $\|i\|^k$ on x is smaller than the value of $\|j\|^k$ on x, and for all arguments that are greater than x, these two functions agree. Hence, correctness of lt^k follows from Lemmas 4.4, 4.2 and part (c) on $k-1$.

(c) Follows from (a) and (b) by propositional reasoning. □

Lemma 4.6. *For all $k > 0$ and $i \in \{0, \ldots, F_k(p(n)) - 1\}$ we have:*

a) $[\![inc^k]\!]\, \|i\|^k \;=\; \|i + 1 \mod F_k(p(n))\|^k$,
b) $[\![dec^k]\!]\, \|i\|^k \;=\; \|i - 1 \mod F_k(p(n))\|^k$.

Proof. Again, we will only prove part (a) since part (b) is entirely analogous. Let $i \in \{0, \ldots, F_k(p(n)) - 1\}$, and $i' := i + 1 \mod F_k(p(n))$. Remember that according to the previous section, $\|i'\|^k$ is the lexicographically next function after $\|i\|^k$. Hence, it is the function that takes an argument x and returns $\|i\|^k\, x$ if there is a smaller argument y s.t. $\|i\|^k\, y$ is not the maximal value. If there is no such smaller y then it returns the value of $\|i\|^k$ on x increased by one. This makes use of the fact that inc^0 increases modulo $F_0(p(n))$. Hence, on all lower-valued arguments the function values are reset to $\|0\|^0$ again. Therefore, correctness follows from Lemmas 4.1, 4.4, 4.2, and 4.5. □

This provides all the necessary tools to model the behaviour of the space-bounded alternating Turing Machine $\mathcal{M}$. In particular, inc^k and dec^k can be used to model the movements of the tape head on a tape of size $F_k(p(n))$.

Remember that a configuration of $\mathcal{M}$ in the computation on w is a triple (q, h, t) where $q \in Q$, $h \in \{0, \ldots, F_k(p(n)) - 1\}$, and $t : \{0, \ldots, F_k(p(n)) - 1\} \to \Gamma$. We will use the HFL type τ_k to model head positions h, and the type τ_{k+1} to model tape contents t. The state component of a configuration will be encoded in the formula. The two alphabet symbols $\mathtt{tt}$ and $\mathtt{ff}$ will be interpreted by the whole, resp. empty set of states, i.e. like the propositions $\mathtt{tt}$ and $\mathtt{ff}$.

First of all we need to define formulas that encode the starting configuration. Formula $head_0^k$ encodes position 0 on a tape of length $F_k(p(n))$. This is simply $head_0^k := min^k$.

Remark 4.7. $ord(head_0^k) = k$, $mar(head_0^k) = 1$ if $k \geq 1$ and 0 otherwise.

In order to encode the tape content of the starting configuration we need yet another auxiliary macro. Let $m \in \mathbb{N}$ and $j_1, \ldots, j_m$ be HFL formulas of type τ_k, and $\psi, \psi_1, \ldots, \psi_m$ be HFL formulas of type τ_0. We write

$$\mathtt{case}^k\ j_1 : \psi_1, \ldots, j_m : \psi_m\ \mathtt{else}\ \psi$$

to abbreviate

$$\lambda I.\Big(\big(\bigvee_{h=1}^{m} (eq^k\ I\ j_h) \wedge \psi_h \big) \vee \big(\psi \wedge \bigwedge_{h=1}^{m} \neg (eq^k\ I\ j_h) \vee \neg \psi_h \big) \Big)$$

Lemma 4.5 immediately gives us the following. Given formulas $j_1, \ldots, j_m$ of type τ_k that represent pairwise different numbers from $\{0, \ldots, F_k(p(n)) - 1\}$, and formulas $\psi, \psi_1, \ldots, \psi_m$, as well as a number $i \in \{0, \ldots, F_k(p(n)) - 1\}$, we have

$$[\![\big(\mathtt{case}^k\ j_1 : \psi_1, \ldots, j_m : \psi_m\ \mathtt{else}\ \psi\big)]\!]\, \|i\|^k \;=\; \begin{cases} [\![\psi_h]\!], & \text{if } \|i\|^k = [\![j_h]\!] \\ [\![\psi]\!], & \text{o.w.} \end{cases}$$

In order to define the tape content of the starting configuration of length $F_k(p(n))$ let $w = a_0 \ldots a_{n-1}$ with $a_i \in \{\mathtt{tt}, \mathtt{ff}\}$. We will use the $\mathtt{case}$-construct to define the initial tape content by case distinction. In order to do so, we need to explicitly address the first n tape cells via a formula χ_i^k s.t. $[\![\chi_i^k]\!] = \|i\|^k$ for all i, k. This can be done recursively using the

auxiliary formulas bit_i from above.

$$\chi_i^0 \;:=\; \bigvee_{j=0}^{p(n)-1} \begin{cases} bit_j\ , & \text{the } j\text{-th bit of } i \text{ is set} \\ \neg bit_j\ , & \text{the } j\text{-th bit of } i \text{ is unset} \end{cases}$$

Note that here we need to represent a number in the range of $0, \ldots, F_0(p(n)) - 1$ by the union over all its bit values, hence a disjunction rather than a conjunction which might seem more intuitive.

For $k > 0$ also recall that we have assumed $p(n) > n$ for all $n \in \mathbb{N}$, in particular $n \leq 2^{p(n)}$. This ensures an easy encoding of the small numbers $0, \ldots, n-1$ as functions of type τ_{k+1}. Function $\|i\|^k$, for $i \in \{0, \ldots, n-1\}$, maps $\|0\|^{k-1}$ to $\|i\|^0$ and all other arguments to $\|0\|^0$ – cf. Fig. 3. Hence, for $k > 0$, let

$$\chi_i^k \;:=\; \texttt{case}\ min^{k-1} : \chi_i^0\ \texttt{else}\ min^0$$

This allows us to represent the starting configuration of $\mathcal{M}$ on w as a simple case distinction.

$$tape_0^k \;:=\; \texttt{case}^k\ \chi_0^k : a_0, \ldots, \chi_{n-1}^k : a_{n-1}\ \texttt{else ff}$$

Here we utilise the fact that we encode the alphabet symbols `tt` and `ff` using the propositions `tt` and `ff` and the blank tape by a sequence of the symbol `ff`.

Remark 4.8. $ord(tape_0^k) = k+1$, $mar(tape_0^k) = 2$.

Next we need formulas that encode the manipulation of configurations. In particular, we will have to model the head movement, and define formulas for reading and updating the symbol at a certain tape position. Remember that in an $F_k(p(n))$-space bounded configuration, the head position can be encoded using type τ_k, and the tape content can be encoded using type $\tau_k \to \mathsf{Pr} = \tau_{k+1}$. We need to define the following functions for any $a \in \Gamma$.

- $read_a^k : \tau_{k+1} \to \tau_k \to \mathsf{Pr}$
  Applied to an encoded tape content and head position it tests whether or not the symbol under the head on that tape is a. It is also a 2-valued predicate. Remember that there only are the two symbols `tt` and `ff` with corresponding encoding.

$$read_{\texttt{tt}}^k \;:=\; \lambda T.\lambda H.(T\ H)$$
$$read_{\texttt{ff}}^k \;:=\; \lambda T.\lambda H.\neg(T\ H)$$

- $write_a^k : \tau_{k+1} \to \tau_k \to \tau_{k+1}$
  Given an encoded tape content t and a head position h it returns the tape content that contains a at position h and complies with t on all other positions.

$$write_a^k \;:=\; \lambda T.\lambda H.\lambda H'.\texttt{if}\ (eq^k\ H\ H')\ \texttt{then}\ a\ \texttt{else}\ (T\ H')$$

Remark 4.9. $ord(read_a^k) = k+2$, $ord(write_a^k) = k+2$, $mar(read_a^k) = 2$, $mar(write_a^k) = 3$.

In the following we write $\|t\|^k$ for the encoding of the tape t of length $F_k(p(n))$ as a function of type $\tau_k \to \mathsf{Pr}$. Equally, the head position h in a configuration is encoded by $\|h\|^k$. We also write $t[h := a]$ for the update of t with a at position h. The next two lemmas show that the above functions are correct. Their proofs are straight-forward. The latter relies on the correctness of the `if-then-else`-construct.

Lemma 4.10. *For all $k \in \mathbb{N}$, all $x \in \Gamma$, all tape contents t, and all head positions h we have*

$$[\![read^k_x]\!]\ \|t\|^k\ \|h\|^k \;=\; \begin{cases} \mathcal{S}, & \text{if the symbol in } t \text{ at position } h \text{ is } x \\ \emptyset, & o.w. \end{cases}$$

Lemma 4.11. *For all $k \in \mathbb{N}$, all $x \in \Gamma$, all tape contents t, t', and all positions h we have: $[\![write^k_x]\!]\ \|t\|^k\ \|h\|^k = \|t'\|^k$ iff $t' = t[h := x]$.*

The movement of the tape head is easily modeled using three functions $move^k_d : \tau_k \to \tau_k$ for $d \in \{-1, 0, +1\}$.

$$move^k_{-1} \;:=\; dec^k \;,\quad move^k_0 \;:=\; \lambda H.H \;,\quad move^k_{+1} \;:=\; inc^k$$

Finally, we use the characterisation of acceptance in an alternating Turing Machine as a reachability game to construct the formula $\Phi^k_{\mathcal{M},w}$. Let $Q = \{q_0, \ldots, q_m, q_{acc}, q_{rej}\}$. We will simultaneously define for each state $q \in Q$ an eponymous function $q : \tau_{k+1} \to \tau_k \to \mathsf{Pr}$ that – given a tape content t and a head position h – signals as a 2-valued predicate whether or not $\mathcal{M}$ accepts starting in the configuration (q, h, t). Let, for all $q \in Q$,

$$\Psi^k_{\mathcal{M},q} \;:=\; \mu q. \begin{pmatrix} q_0 & . & \lambda T.\lambda H.\Psi_0 \\ & \vdots & \\ q_m & . & \lambda T.\lambda H.\Psi_m \\ q_{acc} & . & \lambda T.\lambda H.\mathtt{tt} \\ q_{rej} & . & \lambda T.\lambda H.\mathtt{ff} \end{pmatrix}$$

where for all $i = 0, \ldots, m$:

$$\Psi_i \;:=\; \bigvee_{a \in \Gamma} (read^k_a\ T\ H) \wedge \begin{cases} \bigvee\limits_{(q',b,d) \in \delta(q_i,a)} q'\ (write^k_b\ T\ H)\ (move^k_d\ H) & , \text{ if } q \in Q_\exists \\ \bigwedge\limits_{(q',b,d) \in \delta(q_i,a)} q'\ (write^k_b\ T\ H)\ (move^k_d\ H) & , \text{ if } q \in Q_\forall \end{cases}$$

Then define $\Phi^k_{\mathcal{M},w} := \Psi^k_{\mathcal{M},q_0}\ tape^k_0\ head^k_0$.

The following result about the order-restricted fragment into which $\Phi^k_{\mathcal{M},w}$ falls is easily obtained by collecting all the preceding remarks about the orders and maximal arities of all its subformulas. Note that those of highest type-order are $read^k_a$, $write^k_a$, and q for each $q \in Q$. All of them have order $k + 2$.

Lemma 4.12. *For all $k \in \mathbb{N}$: $\Phi^k_{\mathcal{M},w} \in \mathrm{HFL}^{k+2,3}$.*

Theorem 4.13. *For all $k \geq 1$, all $w \in \Gamma^*$ and all $F_k(p(n))$-space bounded alternating Turing Machines $\mathcal{M}$ we have:*

$$[\![\Phi^k_{\mathcal{M},w}]\!]^{\mathcal{T}_{\mathcal{M},w}} \;=\; \begin{cases} \mathcal{S}, & \text{if } w \in L(\mathcal{M}) \\ \emptyset, & o.w. \end{cases}$$

Proof. Let $\mathcal{M} = (Q, \Sigma, \Gamma, q_0, \delta, q_{acc}, q_{rej})$. Suppose $w \in L(\mathcal{M})$. Then there is an accepting run of $\mathcal{M}$ on w. Remember that $\mathcal{M}$ is alternating. Hence, this run can be represented as a tree T with starting configuration $(q_0, 0, w\mathtt{ff}\ldots\mathtt{ff})$ as the root, s.t.

- every existential configuration has exactly one successor in the tree,
- for every universal configuration the set of its successors in the tree forms the set of all its successor configurations,

- all leaves are accepting configurations.

We now show $[\![\Phi^k_{\mathcal{M},w}]\!]^{\mathcal{T}_{\mathcal{M},w}} = \mathcal{S}$ by induction on the height $h(T)$ of T. Since $\Phi^k_{\mathcal{M},w}$ is a simultaneously defined fixpoint function applied to two arguments we need a stronger inductive hypothesis. We will show that for all $q \in Q$ and all $t : \tau_k \to \mathsf{Pr}$, all $h : \tau_k$ encoding a tape content and a head position: $[\![\Psi^k_{\mathcal{M},q}]\!]\ \|t\|^k\ \|h\|^k = \mathcal{S}$ if $\mathcal{M}$ accepts starting in the configuration given by (q, h, t).

The base case is $h(T) = 1$ which means that the root is an accepting configuration. Hence, $q = q_{acc}$, and the claim is easily seen to be true by two applications of β-reduction.

If $h(T) > 1$ then we need to distinguish two cases. First, assume that $q \in Q_\exists$. Then there is exactly one successor configuration (q', t', h') in T which results from (q, t, h) by one Turing Machine step according to δ. Clearly, $\mathcal{M}$ accepts starting in (q', t', h') and, by hypothesis, we have $[\![\Psi^k_{\mathcal{M},q'}]\!]\ \|t'\|^k\ \|h'\|^k = \mathcal{S}$. One unfolding of the fixpoint formula together with Lemmas 4.1, 4.6–4.11 show that we also have $[\![\Psi^k_{\mathcal{M},q}]\!]\ \|t\|^k\ \|h\|^k = \mathcal{S}$.

The case of $q \in Q_\forall$ is similar. Here, there are possibly several accepting subtrees of T. But the hypothesis applies to all of them and intersection over $\mathcal{S}$ several times is still $\mathcal{S}$.

This shows completeness. Soundness can be proved along the same lines because of determinacy. Note that if $w \notin L(\mathcal{M})$ then this is witnessed by a computation tree in which every universal configuration has only one successor, every existential one retains all of its successors, and all leaves are rejecting. □

4.3. **Lower Bounds on the Model Checking Complexity.**

Theorem 4.14. *For all $k \geq 2$ and all $m \geq 3$ the model checking problem for* $\mathrm{HFL}^{k,m}$ *is* kExpTime*-hard when $|\mathcal{P}| \geq 0$, $|\mathcal{A}| \geq 2$.*

Proof. Let $k \geq 2$. According to Thm. 2.12 there is a $(k-1)$AExpSpace machine $\mathcal{M}$ s.t. $L(\mathcal{M})$ is kExpTime-hard [4]. Using padding we can assume the space required by $\mathcal{M}$ on an input word of length n to be bounded by $F_{k-2}(p(n))$ for some polynomial $p(n) > n$. Thm. 4.13 yields a reduction from $w \in \Gamma^*$ to labeled transition systems $\mathcal{T}_{\mathcal{M},w}$ and a formula $\Phi^{k-2}_{\mathcal{M},w}$ s.t. $w \in L(\mathcal{M})$ iff $\mathcal{T}_{\mathcal{M},w}, s \models \Phi^{k-2}_{\mathcal{M},w}$ for any state s.

According to Lemma 4.12 we have $\Phi^{k-2}_{\mathcal{M},w} \in \mathrm{HFL}^{k,3}$. Furthermore, $|\mathcal{T}_{\mathcal{M},w}|$ is clearly polynomial in n. The size of $\Phi^{k-2}_{\mathcal{M},w}$ is also polynomial in n, but this formula is only an abbreviation using the simultaneous fixpoint definition in $\Phi^k_{\mathcal{M},q_0}$ and we need to consider the Fisher-Ladner closure of its unabbreviated counterpart. But remember the definition of $\Phi^{k-2}_{\mathcal{M},w}$ as $\Psi^k_{\mathcal{M},q_0}\ tape^k_0\ head^k_0$. Unfolding only affects the subformula $\Psi^k_{\mathcal{M},q_0}$ whose size is independent of n. Hence, $|\Phi^{k-2}_{\mathcal{M},w}|$ is also polynomial in n. □

For the fragment HFL^1 a similar result follows from the known ExpTime lower bound for FLC [18] and the embedding of FLC into $\mathrm{HFL}^{1,1}$ [28].

Proposition 4.15 ([18, 28])**.** *There is an* $\mathrm{HFL}^{1,1}$ *formula over a singleton $\mathcal{P}$ and an $\mathcal{A}$ of size 2 whose set of models is* ExpTime*-hard.*

The condition $|\mathcal{A}| \geq 2$ results from the fact that the reduction to FLC model checking is from the pushdown game problem. The number of different modal accessibility relations is $p + 1$ where p is the size of the alphabet in the pushdown games. A close inspection of the ExpTime lower bound proof for this problem [29] shows that $p = 1$ is sufficient.

It has already been observed that model checking HFL on fixed and very small transition systems is non-elementary [19]. We repeat this observation here since it follows from the construction above in a very neat way. Remember that $\log^* n = i$ iff the i-fold iteration of the function $\lambda m.\lceil \log m \rceil$ starting in n yields 1.

Theorem 4.16. *The model checking problem for* HFL *on the fixed transition system of size* 1*, no transitions and no labels is non-elementary when maximal type arities are at least* 2*.*

Proof. Note that Thm. 4.13 uses $p(n)$ many states to encode $F_k(p(n))$ many numbers for any $k \geq 1$. But $F_k(p(n)) = 2^{p(n) \cdot F_{k-1}(p(n))}$, thus $F_{k+1}(\log p(n)) \geq F_k(p(n))$. This means that the reduction in Thm. 4.13 also works with $\log p(n)$ many states, but yields a formula in HFL^{k+1} rather than HFL^k. Iterating this shows that one state suffices for the reduction, but the result is only in $\mathrm{HFL}^{k+\log^* p(n)}$.

Finally, note that by the construction above, this single state 0 does not have any *lower*-transitions. The transition $0 \xrightarrow{test} 0$ is redundant because we have $0 \models [test]\psi$ iff $0 \models \psi$ for any formula ψ. □

There is an apparent intuitive mismatch between this and Cor. 3.18 which both make a statement about the expression complexity of HFL on the smallest possible transition system. For every fixed k, m, this is in ExpTime. However, when k is unbounded it becomes non-elementary. Even though this gap is huge in terms of complexity classes it is just tiny in terms of the HFL types that are necessary to achieve a non-elementary complexity: the type levels only have to be increased by $\log^* p(n)$. Note that $\log^* m \leq 6$ for any natural number m that is representable using electron spins as bits when the entire observable universe was densely packed with electrons. The cause for the apparent intuitive mismatch is simply an underestimation of the exponential time hierarchy. Equally, a tower of height 6 is sufficient to exceed the numbers representable using the electron spins in this way.

The above two theorems raise the question after a lower bound for the data complexity of HFL. In the following we will modify the reduction to yield a formula $\Phi^k_{\mathcal{M}}$ that only depends on the alternating Turing Machine $\mathcal{M}$ rather than both the machine and the input word. Remember that, according to Thm. 2.12, there is – for any $k \geq 1$ – such a machine with a word problem that is kExpTime-hard.

The idea for the modification is simple. It is only the subformula $tape^k_0$ that depends on the input word. First, let

$$tape^k_{empty} \;:=\; \lambda X.\mathtt{ff}$$

model the tape that contains the blank symbol □ only.

Note that $\mathcal{T}_{\mathcal{M},w}$ has $p(n) > n$ many states. Hence, we can use these states together with a single proposition q to model the input word $w = a_0, \ldots, a_{n-1} \in \{\mathtt{tt}, \mathtt{ff}\}^*$.

$$L(i) \;=\; \begin{cases} \{q\}\,, & \text{if } i \leq n \text{ and } a_i = \mathtt{tt} \\ \emptyset\,, & \text{o.w.} \end{cases}$$

Let $\mathcal{T}'_{\mathcal{M},w}$ be the result of this. Note that it differs from $\mathcal{T}_{\mathcal{M},w}$ only through the additional labels on the states. An example with $p(n) = 4$ and $w = \mathtt{tt}\ \mathtt{tt}\ \mathtt{ff}\ \mathtt{tt}$ is shown in Fig. 5. For better readability we depict the relation $\xrightarrow{test}$ only schematically.

All we need now is a formula that traverses through these states and uses the information obtained from each label to generate the original encoding $tape^k_0$ of the real input tape. This

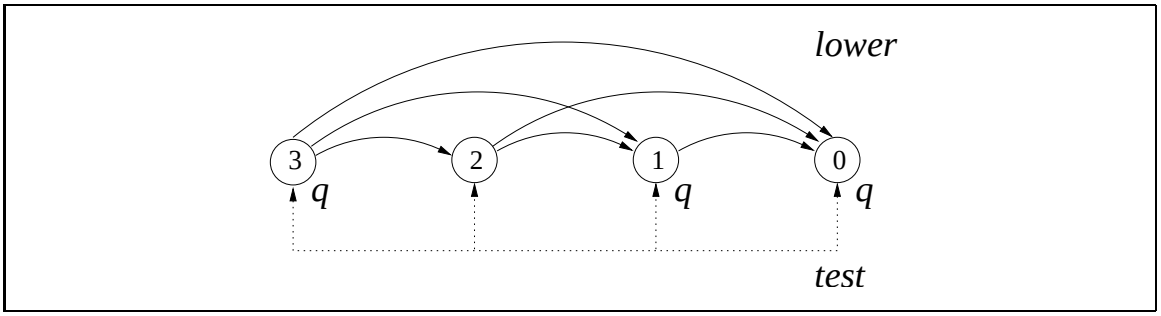

Figure 5: The transition system $\mathcal{T}'_{\mathcal{M},w}$ for $p(n) = 4$ and $w = \mathtt{tt}\ \mathtt{tt}\ \mathtt{ff}\ \mathtt{tt}$.

is done by the function $build^k : \tau_{k+1} \to \tau_k \to \mathsf{Pr} \to \mathsf{Pr} \to \tau_{k+1}$, defined as

$$\begin{aligned} build^k \;:=\; & \mu Z.\lambda T.\lambda H.\lambda C.\lambda Y. \\ & \quad \mathtt{if}\ [test](\chi \leftrightarrow bit_{p(n)})\ \mathtt{then}\ T \\ & \quad \mathtt{else\ if}\ [test](C \to q) \\ & \qquad \mathtt{then}\ Z\ (write^k_{\mathtt{tt}}\ T\ H)\ (inc^{k-1}\ H)\ (\langle lower\rangle C \wedge [lower]Y)\ (C \vee Y) \\ & \qquad \mathtt{else}\ Z\ (write^k_{\mathtt{ff}}\ T\ H)\ (inc^{k-1}\ H)\ (\langle lower\rangle C \wedge [lower]Y)\ (C \vee Y) \end{aligned}$$

The parameters T and H contain the current tape content and the position at which the next symbol is going to be written. From this perspective it is not surprising that $write^k$ and inc^k are applied to them before a recursive call of Z. The parameters C and Y are used to identify the next state in $\mathcal{T}_{\mathcal{M},w}$ which is checked for the label q. Remember the recursive definition of the formulas bit_i which is exactly what is reproduced here. Note that $ord(build^k) = k + 2$ and $mar(build^k) = 5$. Finally, let

$$tape^k_{built} \;:=\; build^k\ tape^k_{empty}\ min^k\ bit_0\ \mathtt{ff}$$

Lemma 4.17. *For all $\mathcal{T}'_{\mathcal{M},w}$ which, in addtion to $\mathcal{T}_{\mathcal{M},w}$ carry the input word w through labels as defined above we have $[\![tape^k_{built}]\!]^{\mathcal{T}'_{\mathcal{M},w}} = [\![tape^k_0]\!]^{\mathcal{T}_{\mathcal{M},w}}$.*

Proof. Assume that $\|t\|^k$ encodes a tape content t and $\|h\|^k$ a head position on this tape. Then we have for all $i \in \{0, \ldots, p(n) - 1\}$:

$$[\![build^k]\!]\ \|t\|^k\ \|h\|^k\ [\![bit_i]\!]\ [\![\textstyle\bigvee_{j=0}^{i-1} bit_j]\!] \;=\; \begin{cases} \|t\|^k\,, & \text{if } i = p(n) \\ [\![build^k]\!]\ \|t[h := a]\|^k\ \|h+1\|^k\ [\![bit_{i+1}]\!]\ [\![\bigvee_{j=0}^{i} bit_j]\!]\,, & \text{if } i < p(n) \end{cases}$$

where $a = \mathtt{tt}$ if $L(i) = q$ and $a = \mathtt{ff}$ if $L(i) = \emptyset$. Thus, when applied to the initial values encoding the blank tape, leftmost head position, the state representing bit 0 and the empty disjunction, this least fixpoint recursion eventually yields the tape onto which the word w at hand is written. This makes use of the fact that $p(n) > n$, i.e. the fixpoint recursion takes at least one more step after reading the entire input word before it terminates. □

Theorem 4.18. *For all $k \geq 2$ and all $m \geq 5$ there is an* $\mathrm{HFL}^{k,m}$ *formula over a singleton $\mathcal{P}$ and an $\mathcal{A}$ of size 2 whose set of models is k*ExpTime*-hard.*

complexity		combined	data	expression
HFL	$\in$	$\mathrm{DTime}(\mathbf{2}_k^{n\cdot\lvert\varphi\rvert^{O(\lvert\varphi\rvert)}})$	kExpTime	$\mathrm{DTime}(\mathbf{2}_k^{\lvert\varphi\rvert^{O(\lvert\varphi\rvert)}})$
	hard	Elementary		Elementary
HFL^0	$\in$	UP$\cap$co-UP	P	UP$\cap$co-UP
	hard	P		P
$\mathrm{HFL}^{1,m}$	$\in$	ExpTime (when $p \geq 3$)	ExpTime (when $p \geq 3$)	ExpTime
	hard			P
$\mathrm{HFL}^{k,m}$, $k \geq 2$	$\in$	kExpTime (when $m \geq 3$, $p \geq 2$)	kExpTime (when $m \geq 5, p \geq 3$ or $m \geq 4, p \geq 4$)	ExpTime
	hard			P

Figure 6: A summary of the model checking complexity results.

Proof. Let $\Phi^k_{\mathcal{M}} := \Psi^k_{\mathcal{M},q_0}\ \mathit{tape}^k_{\mathit{built}}\ \mathit{head}^k_0$. Clearly, $\Phi^k_{\mathcal{M}}$ only depends on $\mathcal{M}$ and not on its input word w. Furthermore, we have $\Phi^k_{\mathcal{M}} \in \mathrm{HFL}^{k+2,5}$. The hardness result then follows from Lemma 4.17 and Thm. 4.13 along the same lines as the proof of Thm. 4.14. □

It is possible to reduce the maximal arity to 4 at the cost of an extra accessibility relation in the model. If there are transitions $i \xrightarrow{\mathit{pred}} j$ iff $j = i - 1$ then the formulas bit_i can be defined more simply as $\mathit{bit}_0 := [\mathit{pred}]\mathtt{ff}$ and $\mathit{bit}_{i+1} := \langle \mathit{pred}\rangle \mathit{bit}_i$, and the parameter Y in build^k is unnecessary.

5. Conclusions

The table in Fig. 6 shows the complexity of the model checking problem for HFL. We distinguish the *combined complexity* (both transition system and formulas as input), the *expression complexity* (model checking on a fixed transition system), and the *data complexity* (model checking with a fixed formula). Note that lower bounds from either expression or data complexity trivially transfer to the combined complexity while upper bounds for that trivially transfer back to both of them. In any case, n denotes the size of the transition system, φ is the input formula, k the maximal type order and m the maximal type arity of one of its subformulas, and $p := |\mathcal{P}| + |\mathcal{A}|$ is the number of underlying propositions and modal acessibility relations. Note that there are standard translations for modal logics that reduce one at the cost of increasing the other whilst preserving satisfiability. These could be incorporated directly into the reduction for the lower bound.

The entries stretching over two columns denote completeness results for the corresponding complexity class. The restrictions of the form $m \geq 2$ etc. of course only apply to the respective lower bound.

The estimation on the time complexity of model checking general HFL uses the fact that the maximal type order as well as the maximal type arity of a subformula of φ are both bounded by $|\varphi|$.

Recall that ELEMENTARY does not have complete problems under polynomial time reductions. The upper bounds on the expression and combined complexity for general HFL model checking are therefore as close as possible to the corresponding lower bound.

The gaps between P and UP∩co-UP simply restate open questions about the exact model checking complexity of the modal μ-calculus. The best upper bound known there in terms of complexity classes is UP∩co-UP so far [14]. The polynomial time lower bound for its expression complexity is taken from an unpublished manuscript [6]. Despite a lot of effort this gap remains open up to date.

The only question about the complexity of model checking HFL that is left unanswered but might be feasible is the gap in the expression complexity of $\mathrm{HFL}^{k,m}$ for any fixed $k, m \geq 1$. It remains to be seen whether there are fixed transition systems $\mathcal{T}_{k,m}$, s.t. for all k, m, the set of $\mathrm{HFL}^{k,m}$ formulas that are satisfied by $\mathcal{T}_{k,m}$ is EXPTIME-hard.

Finally, the kEXPTIME-completeness of $\mathrm{HFL}^{k,m}$'s data complexity immediately implies a hierarchy result regarding expressive power.

Corollary 5.1. *For all $k \in \mathbb{N}$ we have:* $\mathrm{HFL}^k \lneq \mathrm{HFL}^{k+1}$.

Proof. For $k = 0$ this is known already because of $\mathrm{HFL}^0 = \mathcal{L}_\mu \lneq \mathrm{FLC} \leq \mathrm{HFL}^{1,1}$ [22, 28]. Now take any $k \geq 1$. According to Thm. 4.14, there is a formula $\varphi \in \mathrm{HFL}^{k+1,5}$ whose set of models is $(k+1)$EXPTIME-hard. Now suppose that there is also a $\psi \in \mathrm{HFL}^{k,m}$ for some $m \geq 1$ s.t. $\psi \equiv \varphi$. Note that φ is fixed, and so is ψ. According to Thm. 3.17, this same set of models would also be included in kEXPTIME which contradicts the complexity-theoretic time-hierarchy theorem of $k\mathrm{EXPTIME} \subsetneq (k+1)\mathrm{EXPTIME}$. □

REFERENCES

[1] Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonologie, Sprachwissenschaft und Kommunikationsforschung*, 14:113–124, 1961.

[2] H. Békič. *Programming Languages and Their Definition, Selected Papers*, volume 177 of *LNCS*. Springer, 1984.

[3] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[4] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[5] E. M. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.

[6] S. Dziembowski, M. Jurdziński, and D. Niwiński. On the expression complexity of the modal μ-calculus model checking. Unpublished manuscript, 1996.

[7] E. A. Emerson. Uniform inevitability is tree automaton ineffable. *Information Processing Letters*, 24(2):77–79, January 1987.

[8] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.

[9] E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, January 1986.

[10] E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd Symp. on Foundations of Computer Science*, pages 368–377, San Juan, Puerto Rico, October 1991. IEEE.

[11] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, April 1979.

[12] D. Harel, A. Pnueli, and J. Stavi. Propositional dynamic logic of nonregular programs. *Journal of Computer and System Sciences*, 26(2):222–243, April 1983.
[13] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional μ-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *Proc. 7th Conf. on Concurrency Theory, CONCUR'96*, volume 1119 of *LNCS*, pages 263–277, Pisa, Italy, August 1996. Springer.
[14] M. Jurdziński. Deciding the winner in parity games is in $UP \cap$co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.
[15] D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, December 1983.
[16] M. Lange. Model checking propositional dynamic logic with all extras. *Journal of Applied Logic*, 4(1):39–49, 2005.
[17] M. Lange. The alternation hierarchy in fixpoint logic with chop is strict too. *Information and Computation*, 204(9):1346–1367, 2006.
[18] M. Lange. Three notes on the complexity of model checking fixpoint logic with chop. *R.A.I.R.O. – Theoretical Informatics and Applications*, ??(??):??–??, 2006. (to appear).
[19] M. Lange and R. Somla. The complexity of model checking higher order fixpoint logic. In *Proc. 30th Int. Symp. on Math. Foundations of Computer Science, MFCS'05*, volume 3618 of *LNCS*, pages 640–651. Springer, 2005.
[20] M. Lange and R. Somla. Propositional dynamic logic of context-free programs and fixpoint logic with chop. *Information Processing Letters*, 100(2):72–75, 2006.
[21] M. Lange and C. Stirling. Model checking fixed point logic with chop. In M. Nielsen and U. H. Engberg, editors, *Proc. 5th Conf. on Foundations of Software Science and Computation Structures, FOSSACS'02*, volume 2303 of *LNCS*, pages 250–263, Grenoble, France, April 2002. Springer.
[22] M. Müller-Olm. A modal fixpoint logic with chop. In C. Meinel and S. Tison, editors, *Proc. 16th Symp. on Theoretical Aspects of Computer Science, STACS'99*, volume 1563 of *LNCS*, pages 510–520, Trier, Germany, 1999. Springer.
[23] A. Pnueli. The temporal logic of programs. In *Proc. 18th Symp. on Foundations of Computer Science, FOCS'77*, pages 46–57, Providence, RI, USA, October 1977. IEEE.
[24] H. Schwichtenberg. An upper bound for reduction sequences in typed λ-calculus. *Archives of Mathematical Logic*, 30:405–408, 1991.
[25] R. Statman. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9:73–81, 1979.
[26] C. Stirling. Local model checking games. In I. Lee and S. A. Smolka, editors, *Proc. 6th Conf. on Concurrency Theory, CONCUR'95*, volume 962 of *LNCS*, pages 1–11, Berlin, Germany, 1995. Springer.
[27] A. Tarski. A lattice-theoretical fixpoint theorem and its application. *Pacific Journal of Mathematics*, 5:285–309, 1955.
[28] M. Viswanathan and R. Viswanathan. A higher order modal fixed point logic. In Ph. Gardner and N. Yoshida, editors, *Proc. 15th Int. Conf. on Concurrency Theory, CONCUR'04*, volume 3170 of *LNCS*, pages 512–528, London, UK, 2004. Springer.
[29] I. Walukiewicz. Pushdown processes: Games and model-checking. *Information and Computation*, 164(2):234–263, 2001.
[30] E. Zermelo. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. In *Proc. 5th Int. Congress of Mathematicians*, volume II, pages 501–504. Cambridge University Press, 1913.