Capturing Bisimulation-Invariant Complexity Classes by Polyadic Higher-Order Fixpoint Logic

by
David Kronenberger

Dipl.-Math. Florian Bruse, Advisor
Prof. Dr. Martin Lange, Reviewer
Prof. Dr. Stefan Göller, Reviewer

A thesis submitted in partial fulfillment
of the requirements for the
Degree of Master of Science
in Computer Science

UNIVERSITY OF KASSEL
Hesse, Germany

December 21, 2018

I declare that I have developed and written the enclosed thesis entirely by myself, and have not used sources or means without declaration in the text.

Kassel, December 21, 2018

. . . . . .

(**David Kronenberger**)

Contents

Chapter 1

Introduction

Descriptive complexity theory describes the complexity classes known from computational complexity theory with logics. The key advantage is that complexity classes are characterized by logical resources instead of referring to automaton models or space and time bounds. The first known result in the area of descriptive complexity is due to R. Fagin. In 1974 he showed that the well-known complexity class NP coincides with $\exists SO$, the existential fragment of second-order logic [1].

The complexity classes that are of interest in this thesis are those comprising the problems that can be solved in k-fold exponential time or using k-fold exponential space. These prominent classes are called k-EXPTIME and k-EXPSPACE, respectively. The logic we will use to capture the bisimulation-invariant k-EXPTIME, abbreviated by k-EXPTIME/$\sim$, is called Polyadic Higher-Order Fixpoint Logic and was introduced by M. Lange and E. Lozes in [2] abbreviated by PHFL.

PHFL is a modal fixpoint logic that extends Higher-Order Fixpoint Logic, abbreviated by HFL, due to M. Viswanathan and R. Viswanathan [3] with the Polyadic μ-Calculus from M. Otto [4]. HFL extends the modal μ-calculus by a simply typed λ-calculus which allows to define higher-order functions on predicates.

In this thesis, we show that the logic PHFL that uses formulas with order at most k, abbreviated with PHFL^k, captures k-EXPTIME/$\sim$ where $k > 1$. Due to the fact that the above statement is also true for $k = 0$ [4] and $k = 1$ [2] we were able to verify that PHFL^k captures k-EXPTIME/$\sim$ for any $k \geq 0$ on finite labelled transition systems. Furthermore, we will show that a restriction of PHFL called tail-recursive HFL that uses formulas with order at most $k+1$, abbreviated with PHFL$^{k+1}_{tail}$, captures bisimulation-invariant k-EXPSPACE, abbreviated k-EXPSPACE/$\sim$ where $k > 1$. In analogy to the exponential time classes, we will prove that PHFL$^{k+1}_{tail}$ captures

k-EXPSPACE/$\sim$ for any $k \geq 0$.

The results presented in this paper are divided in two parts. In the first part (Chapter 3) it is shown that the upper bounds of the expressive power of PHFL^k and PHFL$^{k+1}_{tail}$ are k-EXPTIME and k-EXSPACE, repectively. This is shown by a reduction from the model-checking problem of PHFL^k and PHFL$^{k+1}_{tail}$ to the model-checking problem of HFL^k and HFL$^{k+1}_{tail}$, respectively. Because it is known that the model checking problems of HFL^k and HFL$^{k+1}_{tail}$ are in k-EXPTIME and k-EXPSPACE, the same holds for PHFL^k and PHFL$^{k+1}_{tail}$, respectively.

In the second part (Chapter 4), we show that the logics PHFL^k and PHFL$^{k+1}_{tail}$ are as least as expressible as k-EXPTIME/$\sim$ and k-EXPSPACE/$\sim$. This can be proven by encoding the run of a Turing Machine as query. As another possibility higher-order logic extended with least fixpoints, abbreviated HO(LFP)$^{k+1}$, and higher-order logic extended with partial fixpoints, abbreviated HO(PFP)$^{k+1}$ can be used. Higher-Order logic, in contrast to first-order logic, allows quantification over sets, sets of sets and so on. Therefore, the proofs of the lower bounds are divided into two steps each.

In the first step it is shown that the lower bound of the expressive power of HO(LFP)$^{k+1}$ and HO(PFP)$^{k+1}$ are k-EXPTIME/$\sim$ and k-EXPSPACE/$\sim$ respectively. C. Freire and A. Martins showed in [5] that HO(LFP)$^{k+1}$ is at least as expressive as k-EXPTIME/$\sim$. The logic HO(PFP)$^{k+1}$ being as least as expressible as k-EXPSPACE/$\sim$ will be shown in this thesis by encoding the run of a Turing Machine as query.

In the second step we show that the bisimulation-invariant fragments of HO(LFP)$^{k+1}$ and HO(PFP)$^{k+1}$, respectively, can be encoded by PHFL^k and PHFL$^{k+1}_{tail}$, respectively. This required a lot of effort since there are no unrestricted quantifiers in PHFL and since PHFL and HO have different sets of types. Quantifiers can be simulated via the use of orders for each type of the bound variable. These orders make it possible to define successors and these are helpful to iterate over the scope of the bound variable. Note that the base type of PHFL denotes sets of elements whereas the base type of HO denotes elements. This problem is non-trivial but can be solved by using the polyadicity of PHFL.

Structure of the Thesis

The structure of the thesis is as follows. In Section 2.1 we review queries and all necessary definitions for the term bisimulation-invariance. Section 2.2 explains the term fixpoints and some variants of them. In Section 2.3 we use fixpoints to define PHFL and the tail-recursive fragment of it. The

following section explains descriptive complexity in more detail and defines k-EXPTIME/$\sim$ and k-EXPSPACE/$\sim$. In the last section of Chapter 2 we define the intermediate logics HO(LFP)$^{k+1}$ and HO(PFP)$^{k+1}$. In Chapter 3 we show that the upper bounds of the expressive power of PHFL^k and PHFL$^{k+1}_{tail}$ are k-EXPTIME and k-EXSPACE, repectively. In the last chapter we show that the lower bounds of the expressive power of PHFL^k and PHFL$^{k+1}_{tail}$ are k-EXPTIME/$\sim$ and k-EXPSPACE/$\sim$, respectively. In Section 4.1 we identify an order on the base type of HO and explain that the proof can be simplified to using so called reduced labelled transition systems. In Section 4.2 we show how the quantification of any type can be encoded by PHFL formulas. In the next section we define the encoding for any bisimulation-invariant HO(LFP)^k formula in PHFL^k and verifies its correctness. As a consequence the lower bound of the expressive power of PHFL^k is k-EXPTIME/$\sim$. In the last section we define the encoding of the partial fixpoint operator of HO(PFP)$^{k+1}$ in PHFL$^{k+1}_{tail}$ in a similar manner as the previous section. Finally, the correctness of this encoding is shown and similarly to PHFL^k this means that the lower bound of the expressive power of PHFL$^{k+1}_{tail}$ is k-EXPSPACE/$\sim$.

Chapter 2

Preliminaries

This chapter introduces all necessary definitions to prove that $\mathrm{PHFL}^k = k$-EXPTIME/$\sim$ and $\mathrm{PHFL}^{k+1}_{tail} = k$-EXPSPACE/$\sim$. The notions are mainly from [6], [7], [4], [5] and [2].

We assume that the reader is already familiar with basic notions of first order logic and computational complexity. In the first section we define a graph called labelled transition systems and a relation on it called bisimulation. Additionally, we define queries and a characterization of queries called bisimulation invariant. In the next section we give some information on fixpoints that are used in the section that follows. There the logic PHFL is defined. In Section 2.4 we present the descriptive complexity and the complexity classes k-EXPTIME/$\sim$ and k-EXPSPACE/$\sim$. In the last section we define the higher-order logic and combinations with LFP and PFP.

2.1 Bisimulation Invariance

First of all, we need the definition of *labelled transition systems*. A labelled transition system is a graph with labelled vertices and edges. Formally, it is the following.

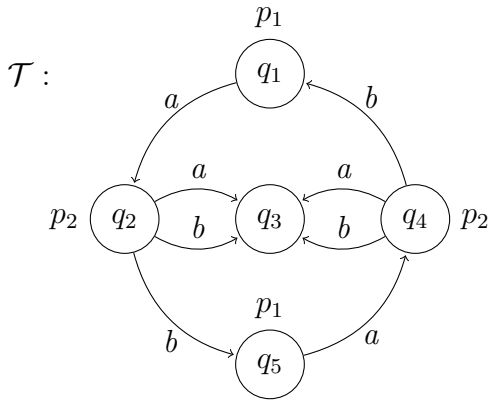
Definition 2.1. A **labelled transition system** (**LTS**) is a quintuple $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$, where

- Q is a set of states,
- Σ is a finite set of actions,
- P is a finite set of propositions,
- $\Delta \subseteq Q \times \Sigma \times Q$ is the labelled transition relation and
- $v : Q \to 2^P$ is a function that maps each state to a set of propositions.

For all $q_1, q_2 \in Q$ and all $a \in \Sigma$ we write $q_1 \xrightarrow{a} q_2$ for $(q_1, a, q_2) \in \Delta$. From

now on, we assume that every LTS is a finite LTS. Finite LTS are those where the set of states is finite.

Example 2.2. *As mentioned above, an LTS can be seen as a graph with labelled vertices and edges. For example, an LTS is* $\mathcal{T} = (\{q_1, q_2, q_3, q_4, q_5\}, \{a, b\}, \{p_1, p_2\}, \Delta, v)$*, where* $\Delta = \{(q_1, a, q_2), (q_2, a, q_3), (q_2, b, q_3), (q_2, b, q_5), (q_5, a, q_4), (q_4, a, q_3), (q_4, b, q_3), (q_4, b, q_1)\}$*,* $v(q_1) = v(q_2) = v(q_4) = v(q_5) = \{p_1\}$ *and* $v(q_3) = \emptyset$*.* $\mathcal{T}$ *can be visualized as follows.*



On these systems or rather on the states of the systems it is possible to define relations. The following relation describes states that have a sufficiently similar behaviour.

Definition 2.3. Let be $\mathcal{T}_1 = (Q_1, \Sigma, P, \Delta_1, v_1)$ and $\mathcal{T}_2 = (Q_2, \Sigma, P, \Delta_2, v_2)$ two LTS. A **bisimulation** is a binary relation $R \subseteq Q_1 \times Q_2$ that satisfies for all $(q_1, q_2) \in R$:

- $v_1(q_1) = v_2(q_2)$,
- for all $a \in \Sigma$ and all $q_1' \in Q_1$, if $q_1 \xrightarrow{a} q_1'$, then there is a state $q_2' \in Q_2$, such that $q_2 \xrightarrow{a} q_2'$ and $(q_1', q_2') \in R$ and
- for all $a \in \Sigma$ and all $q_2' \in Q_2$, if $q_2 \xrightarrow{a} q_2'$, then there is a state $q_1' \in Q_1$, such that $q_1 \xrightarrow{a} q_1'$ and $(q_1', q_2') \in R$.

We call two states $q_1 \in Q_1$, $q_2 \in Q_2$ **bisimilar**, noted as $(\mathcal{T}_1, q_1) \sim (\mathcal{T}_2, q_2)$, if there is a bisimulation R such that $(q_1, q_2) \in R$.

It is easy to prove that $\sim$ is an equivalence relation.

Example 2.4. *Let* $\mathcal{T}$ *be like in Example 2.2. In this example, we compare states of the same LTS. It holds that* $(\mathcal{T}, q_1) \sim (\mathcal{T}, q_5)$ *and* $(\mathcal{T}, q_2) \sim (\mathcal{T}, q_4)$ *because* $R = \{(q_1, q_5), (q_2, q_4), (q_3, q_3)\}$ *is a bisimulation relationand includes* (q_1, q_5) *and* (q_2, q_4)*. Furthermore, it holds that* $(\mathcal{T}, q_5) \sim (\mathcal{T}, q_1)$ *and* $(\mathcal{T}, q_4) \sim (\mathcal{T}, q_2)$*. Moreover, all reflexive pairs are bisimilar. Note that* $(\mathcal{T}, q_1) \not\sim (\mathcal{T}, q_3)$ *because* $v(q_1) \neq v(q_3)$ *and* $(\mathcal{T}, q_1) \not\sim (\mathcal{T}, q_2)$ *because* q_2 *has an outgoing b-transition but* q_1 *has no b-transitions.*

Because $\sim$ is an equivalence relation that is compatible with the propositions and actions of LTS it can be used to build a factor structure over an LTS.

Definition 2.5. Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS and $(q_1, \ldots, q_n) \in Q^n$ an n-tuple then we call $(\mathcal{T}, q_1, \ldots, q_n)$ **reduced** with respect to $q_1, \ldots, q_n$ iff all states of $\mathcal{T}$ are reachable by at least one q_i and $\sim$ coincides with equality, where $i \in \{1, \ldots, n\}$. For all LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ and all tuples of states $(q_1, \ldots, q_n) \in Q^n$ we associate to $(\mathcal{T}, q_1, \ldots, q_n)$ the reduced tuple $RED(\mathcal{T}, q_1, \ldots, q_n)$ by factoring $\mathcal{T}$ with respect to $\sim$, named as $\mathcal{T}_\sim$, and pruning all states of $\mathcal{T}_\sim$ that cannot be reached from at least one q_i. We call the resulting graph the **reduced LTS** of $\mathcal{T}$ with respect to $(q_1, \ldots, q_n)$.

Furthermore, we can describe properties of LTS. *Queries* are one way to describe these.

Definition 2.6. [4] An **r-adic query** $\mathcal{Q}$ is a mapping that associates to each LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ a subset $\mathcal{Q}^\mathcal{T}$ of Q^r such that any isomorphism $f : \mathcal{T} \to \mathcal{T}'$ between $\mathcal{T}$ and another LTS $\mathcal{T}'$ maps tuples not in $\mathcal{Q}^\mathcal{T}$ to tuples not in $\mathcal{Q}^{\mathcal{T}'}$ and vice versa.

One characterization of queries is called *bisimulation invariant*. This characterization describes those queries that cannot distinguish bisimilar states. In [4] this property is defined over so called *Kripke structures*. A Kripke structure is an LTS with only one type of actions.

Definition 2.7. Let $\mathcal{T}$, $\mathcal{T}'$ be two LTS with $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ and $\mathcal{T}' = (Q', \Sigma, P, \Delta', v')$. Moreover, let be $(q_1, \ldots, q_r) \in Q^r$ and $(q_1{}', \ldots, q_r{}') \in Q^{r\prime}$.

A query $\mathcal{Q}$ is called **bisimulation invariant** if $(\mathcal{T}, q_i) \sim (\mathcal{T}', q_i')$ for all $1 \leq i \leq r$ implies that $(q_1, \ldots, q_r) \in \mathcal{Q}^\mathcal{T}$ iff $(q_1{}', \ldots, q_r{}') \in \mathcal{Q}^{\mathcal{T}'}$.

Example 2.8. *[2] Let $\mathcal{Q}$ be a 2-adic query that maps a LTS to the set of pairs of states that are bisimilar. As example let $\mathcal{T}$ the LTS from Example 2.2. The query $\mathcal{Q}$ maps $\mathcal{T}$ to the following set of pairs:*

$$\mathcal{Q}^\mathcal{T} = \{(q_1, q_1), \ldots, (q_5, q_5), (q_1, q_5), (q_5, q_1), (q_2, q_4), (q_4, q_2)\}$$

As mentioned in Example 2.4, $\sim$ is an equivalence relation. It follows that $\mathcal{Q}$ is bisimulation invariant.

2.2 Fixpoints

To define the polyadic higher-order fixpoint logic and the higher-order logic with least and partial fixpoints, we examine fixpoints in general in this section. The first fixpoint we consider is the least fixpoint.

Definition 2.9. Let $F\colon A \to A$ be an operator on a finite set A, then $x \in A$ is called a **fixpoint** of F if $F(x) = x$. Let x be a fixpoint of F and $\sqsubseteq$ an partial order on A, then x is called the **least fixpoint** of F, abbreviated as $LFP(F)$, if for all other fixpoints y of F the condition $x \sqsubseteq y$ holds. A fixpoint x is called the **greatest fixpoint** if $y \sqsubseteq x$ for all fixpoints y of F.

From the Knaster-Tarski Theorem [8] we know that if an operator $F\colon A \to A$ is monotone and A is a complete lattice regarding to $\sqsubseteq$ then the least and greatest fixpoints of F exists. F is monotone if for all $x, y \in A$ if $x \sqsubseteq y$ then $F(x) \sqsubseteq F(y)$ holds.

Example 2.10. *Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS and $F : Q^2 \to Q^2$ an operator on Q^2 defined as*

$$\begin{aligned} F(X) = &\{(q,p) \in Q^2 \mid v(q) \neq v(p)\} \cup \\ &\{(q,p) \in Q^2 \mid \textit{there exists } a \in \Sigma \textit{ with } q \xrightarrow{a} q' \textit{ such that for all } p' \in Q \\ &\textit{it holds that } p \xrightarrow{a} p' \textit{ implies } (q',p') \in X\} \cup \\ &\{(q,p) \in Q^2 \mid \textit{there exists } a \in \Sigma \textit{ with } p \xrightarrow{a} p' \textit{ such that for all } q' \in Q \\ &\textit{it holds that } q \xrightarrow{a} q' \textit{ implies } (q',p') \in X\} \end{aligned}$$

Then $LFP(F)$ represents all those pairs of states (q,p) such that $q \not\sim p$.

The following theorem shows a possibility to calculate the least fixpoint of a monotone operator if the operating set is a complete lattice with respect to its order.

Theorem 2.11 (Kleene Fixed-Point Theorem [9])**.** *Let $F : A \to A$ be a monotone operator on A and A regarding to $\sqsubseteq$ a complete lattice, then there exists a finite sequence $X_0, \ldots, X_m$ such that the first part X_0 is the smallest element in A with respect to $\sqsubseteq$, the $(i+1)$-th part X_{i+1} is $F(X_i)$ and $X_m = X_{m+1}$.*

Next, we define the partial fixpoint. Since the LFP restricts the operator to be monotone, the partial fixpoint need no restriction on the operator.

Definition 2.12. Let $F\colon A \to A$ be an operator on a finite set A, then the **partial fixpoint** of F, abbreviated as $PFP(F)$, is defined as follows:

$$PFP(F) \coloneqq \begin{cases} F^{i+1}(\varnothing) = F^i(\varnothing), & \text{if such } i \in \{0, \ldots, |A|\} \text{ exists} \\ \varnothing, & \text{otherwise,} \end{cases}$$

where $F^0(\varnothing) = \varnothing$, $F^1(\varnothing) = F(\varnothing)$, $F^2(\varnothing) = F(F(\varnothing))$, and so on.

Note that for monotone F holds $PFP(F)$ equals $LFP(F)$.

Example 2.13. *Let* $F : \mathcal{P}(\{1,\ldots,n\}) \to \mathcal{P}(\{1,\ldots,n\})$ *be an operator on* $\mathcal{P}(\{1,\ldots,n\})$ *defined as*

$$\begin{aligned} F(X) = \{x \in \{1,\ldots,n\} \mid\ & x \in X \text{ and there exists } y \in \{1,\ldots,n\} \text{ such that} \\ & y < x \text{ and } y \notin X \text{ or} \\ & x \notin X \text{ and for all } y \in \{1,\ldots,n\} \text{ holds if} \\ & y < x \text{ then } y \in X\}. \end{aligned}$$

If we see $X \in \mathcal{P}(\{1,\ldots,n\})$ *as a binary string* b*, where a* 1 *at the* i*-th position means that* i *is in* X*, then* $F(X)$ *returns the set* Y *such that the binary string* b' *of* Y *is* $b' = b + 1 \mod n$*. Then* $PFP(F)$ *returns* $\varnothing$ *because for every* i *it holds that* $F^i(\varnothing) \neq F^{i+1}(\varnothing)$*.*

2.3 Polyadic Higher Order Fixpoint Logic

In this section, we present a logic with name Polyadic Higher Order Fixpoint Logic, abbreviated with PHFL, that was introduced by M. Lange and E. Lozes in [2]. It is defined over LTS (see Definition 2.1) and extends the polyadic modal μ-calculus [4] by higher-order fixpoints analogue to M. Viswanathan and R. Viswanathan who extended the modal μ-calculus [10] with higher-order fixpoints [3] and λ-calculus. The logic of M. Viswanathan and R .Viswanathan with name higher order fixed point logic is a combination of propositional logic, modal operators and a simply typed λ-calculus with fixed point operators.

2.3.1 Types of PHFL

Before defining formulas of PHFL we need to introduce the PHFL types. These definitions are guided by [3] and [2].

Definition 2.14. PHFL types are given by the grammar

$$\sigma, \tau ::= \bullet \mid \sigma^v \to \tau,$$

where v is called *variance*. The **variances** of PHFL are defined by the grammar

$$v ::= + \mid - \mid 0.$$

All types will be interpreted as a partially ordered sets. Keep in mind that partial orders are relations that are reflexive, transitive and antisymmetric. Let $\mathcal{A} = (A, \leq_A)$ and $\mathcal{B} = (B, \leq_B)$ be two partial orders. Then $\mathcal{A} \to \mathcal{B}$ is the partial order of monotone functions ordered pointwise, i.e.

$$\mathcal{A} \to \mathcal{B} = \{f \colon A \to B \mid \text{ for all } x, y \in A.\, x \leq_A y \text{ implies } f(x) \leq_B f(y)\}$$

and the ordering relation is given by

$$f \leq_{\mathcal{A}\to\mathcal{B}} g \text{ iff } \text{ for all } x \in \mathcal{A}.\, f(x) \leq_{\mathcal{B}} g(x).$$

Definition 2.15. Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS and $d \in \mathbb{N}$ with $d > 0$, then $[\![\tau]\!]_{\mathcal{T}}$ denotes the semantics of type τ which is defined as follows:

$$[\![\tau]\!]_{\mathcal{T}} = \begin{cases} (\mathcal{P}(Q^d), \subseteq), & \text{if } \tau = \bullet \\ (([\![\sigma_1]\!]_{\mathcal{T}})^v \to [\![\sigma_2]\!]_{\mathcal{T}}, \leq_{([\![\sigma_1]\!]_{\mathcal{T}})^v \to [\![\sigma_2]\!]_{\mathcal{T}}}), & \text{if } \tau = \sigma_1^v \to \sigma_2, \end{cases}$$

where for any partial order $\mathcal{A} = (A, \leq_A)$, $\mathcal{A}^v = (A, \leq_A^v)$ is a partial order with $\leq_A^+ = \leq_A$, $\leq_A^- = \{(a, b) \mid (b, a) \in \leq_A\}$ and $\leq_A^0 = \leq_A^+ \cap \leq_A^-$.

The partial orders $[\![\tau]\!]_{\mathcal{T}}$ for any PHFL type τ are complete lattices. That means that we have meets and joins, denoted by $\sqcap_{[\![\tau]\!]_{\mathcal{T}}}$ and $\sqcup_{[\![\tau]\!]_{\mathcal{T}}}$ respectively, and least and greatest elements, denoted by $\bot_{[\![\tau]\!]_{\mathcal{T}}}$ and $\top_{[\![\tau]\!]_{\mathcal{T}}}$ respectively for any subset of $[\![\tau]\!]_{\mathcal{T}}$. This ensures that the least and greatest fixpoint over all monotone PHFL types exist [8]. See Section 2.2 for further information about fixpoints.

Definition 2.16. The **maximal arity** $ma(\tau)$ and the **order** $ord(\tau)$ of a PHFL type τ are defined inductively on τ as follows:

$$ma(\tau) = \begin{cases} 1, & \text{if } \tau = \bullet \\ max(\{n\} \cup \{ma(\tau_i) \mid 1, \ldots, n\}), & \text{if } \tau = \tau_1 \to \cdots \to \tau_n \to \bullet \end{cases}$$

$$ord(\tau) = \begin{cases} 0, & \text{if } \tau = \bullet \\ max(\{1 + ord(\sigma_1), ord(\sigma_2)\}), & \text{if } \tau = \sigma_1 \to \sigma_2 \end{cases}$$

2.3.2 Syntax of PHFL

Next, we want to define the syntax of PHFL formulas.

Definition 2.17. Let P be a set of propositions, Σ a set of actions, $\mathcal{V} = \{X_1, X_2, \dots\}$ be a countably infinite set of variables and $d \in \mathbb{N}$ with $d > 0$, then d**-adic PHFL formulas** $\Phi, \Psi, \dots$ are defined by the grammar

$$\Phi, \Psi ::= \top \mid p_i \mid \Phi \vee \Psi \mid \neg\Phi \mid \langle a \rangle_i \Phi \mid \{\mathbf{j}\}\Phi \mid X \mid \lambda(X^v : \tau).\Phi \mid \Phi\,\Psi \mid \mu(X : \tau).\Phi$$

where

- $\mathbf{j} = (e(1), \dots, e(d))$ and $e : \{1, \dots, d\} \to \{1, \dots, d\}$,
- $i \in \{1, \dots, d\}$
- v is a variance,
- τ is a type,
- $p \in P$,
- $a \in \Sigma$ and
- $X \in \mathcal{V}$.

For convenience, we use some other further standard connectives and operators like $\Phi \wedge \Psi$, $[a]_i\Phi$, $\nu X : \tau.\Phi$ or $\Phi \Leftrightarrow \Psi$. The formulas can be thought of as a game played by two players moving d many pebbles along the transitions of an LTS. The two players are called Prover and Refuter. That means p_i reflects, whether the position of the i-th pebble satisfies property p. $\langle a \rangle_i \Phi$ means the Prover has to move the i-th pebble along an a-transition and check whether Φ holds there. The formula $\{\mathbf{j}\}\Phi$ describes that all pebbles

Figure 2.1: Derivation Rules for PHFL formulas.

$$\frac{}{\Gamma \vdash \top : \bullet} \qquad \frac{}{\Gamma \vdash p_i : \bullet} \qquad \frac{\Gamma \vdash \Phi : \bullet}{\Gamma \vdash \langle a \rangle_i \Phi : \bullet} \qquad \frac{\Gamma \vdash \Phi : \bullet}{\Gamma \vdash \{\mathbf{j}\}\Phi : \bullet} \qquad \frac{\Gamma^- \vdash \Phi : \bullet}{\Gamma \vdash \neg\Phi : \bullet}$$

$$\frac{\Gamma \vdash \Phi : \bullet \qquad \Gamma \vdash \Psi : \bullet}{\Gamma \vdash \Phi \vee \Psi : \bullet} \qquad \frac{v \in \{+, 0\}}{\Gamma, X^v : \tau \vdash X : \tau} \qquad \frac{\Gamma, X^v : \sigma \vdash \Phi : \tau}{\Gamma \vdash \lambda(X^v : \tau).\Phi : \sigma^v \to \tau}$$

$$\frac{\Gamma \vdash \Phi : \sigma^+ \to \tau \qquad \Gamma \vdash \Psi : \sigma}{\Gamma \vdash \Phi\,\Psi : \tau} \qquad \frac{\Gamma \vdash \Phi : \sigma^- \to \tau \qquad \Gamma^- \vdash \Psi : \sigma}{\Gamma \vdash \Phi\,\Psi : \tau}$$

$$\frac{\Gamma \vdash \Phi : \sigma^0 \to \tau \qquad \Gamma \vdash \Psi : \sigma \qquad \Gamma^- \vdash \Psi : \sigma}{\Gamma \vdash \Phi\,\Psi : \tau} \qquad \frac{\Gamma, X^+ : \tau \vdash \Phi : \tau}{\Gamma \vdash \mu(X : \tau).\Phi : \tau}$$

are moved from their current position to the position indicated by tuple **j** and after that Φ has to be satisfied. The player who needs to move the pebbles changes on negations. $\lambda(X^v\colon \tau).\Phi$ is interpreted as a function that expects arguments from $(\llbracket\tau\rrbracket_{\mathcal{T}})^v$. We see that the formulas also have types. Therefore, we have ensure that a formula is well-typed.

Definition 2.18. Let $X_1, \ldots, X_n$ variables, Φ a PHFL formula, $v_1, \ldots, v_n$ variances and $\tau, \tau_1, \ldots, \tau_n$ types, then $\Gamma = X_1^{v_1}\colon \tau_1, \ldots X_n^{v_n}\colon \tau_n$ is called a **type environment** and $\Gamma \vdash \Phi\colon \tau$ is called a **type judgement**. Let $\Gamma^- = X_1^{v_1^-}\colon \tau_1, \ldots X_n^{v_n^-}\colon \tau_n$ be a type environment then $\Gamma^- = X_1^{v_1^-}\colon \tau_1, \ldots X_n^{v_n^-}\colon \tau_n$, where $-^- = +$, $+^- = -$ and $0^- = 0$.

A type judgment is called *derivable* if it generates a derivation tree according to the rules of Figure 2.1. This type system makes sure that we do not create senseless formulas like $\langle a\rangle_i p_j p_k$. Furthermore, it controls the occurrences of negations and therefore guarantees monotonicity. A formula

Figure 2.2: Semantics of PHFL formulas.

$$
\begin{aligned}
\llbracket\Gamma \vdash \top\colon \bullet\rrbracket^\eta_{\mathcal{T}} &= Q^d \\
\llbracket\Gamma \vdash p_i\colon \bullet\rrbracket^\eta_{\mathcal{T}} &= \{(q_1, \ldots, q_d) \in Q^d \mid p \in v(q_i)\} \\
\llbracket\Gamma \vdash \langle a\rangle_i\Phi\colon \bullet\rrbracket^\eta_{\mathcal{T}} &= \{(q_1, \ldots, q_d) \in Q^d \mid \text{ there exists} \\
&\qquad (q'_1, \ldots, q'_d) \in \llbracket\Gamma \vdash \Phi\colon \bullet\rrbracket^\eta_{\mathcal{T}} \text{ such that} \\
&\qquad q_i \xrightarrow{a} q'_i \text{ and for all } j \neq i \text{ holds } q_j = q'_j\} \\
\llbracket\Gamma \vdash \Phi \vee \Psi\colon \bullet\rrbracket^\eta_{\mathcal{T}} &= \llbracket\Gamma \vdash \Phi\colon \bullet\rrbracket^\eta_{\mathcal{T}} \sqcup_\bullet \llbracket\Gamma \vdash \Psi\colon \bullet\rrbracket^\eta_{\mathcal{T}} \\
\llbracket\Gamma \vdash \neg\Phi\colon \bullet\rrbracket^\eta_{\mathcal{T}} &= Q^d \setminus \llbracket\Gamma^- \vdash \Phi\colon \bullet\rrbracket^\eta_{\mathcal{T}} \\
\llbracket\Gamma \vdash \{\mathbf{j}\}\Phi\colon \bullet\rrbracket^\eta_{\mathcal{T}} &= \{(q_1, \ldots, q_d) \in Q^d \mid \\
&\qquad (q_{j_1}, \ldots, q_{j_d}) \in \llbracket\Gamma \vdash \Phi\colon \bullet\rrbracket^\eta_{\mathcal{T}}\} \\
\llbracket\Gamma, X\colon \tau \vdash X\colon \tau\rrbracket^\eta_{\mathcal{T}} &= \eta(X) \\
\llbracket\Gamma \vdash \mu(X\colon \tau).\,\Phi\colon \tau\rrbracket^\eta_{\mathcal{T}} &= \bigsqcap\nolimits_{\llbracket\tau\rrbracket_{\mathcal{T}}} \{\mathcal{X} \in \llbracket\tau\rrbracket_{\mathcal{T}} \mid \\
&\qquad \llbracket\Gamma, X^+\colon \tau \vdash \Phi\colon \tau\rrbracket^{\eta[X\mapsto\mathcal{X}]}_{\mathcal{T}} \leq_{\llbracket\tau\rrbracket_{\mathcal{T}}} \mathcal{X}\} \\
\llbracket\Gamma \vdash \lambda(X^v\colon \sigma).\,\Phi\colon \sigma^v \to \tau\rrbracket^\eta_{\mathcal{T}} &= F \in \llbracket\sigma^v \to \tau\rrbracket_{\mathcal{T}} \text{ such that for all } \mathcal{X} \in \llbracket\sigma\rrbracket_{\mathcal{T}}. \\
&\qquad F(\mathcal{X}) = \llbracket\Gamma, X^v\colon \sigma \vdash \Phi\colon \tau\rrbracket^{\eta[X\mapsto\mathcal{X}]}_{\mathcal{T}} \\
\llbracket\Gamma \vdash \Phi\,\Psi\colon \tau\rrbracket^\eta_{\mathcal{T}} &= \llbracket\Gamma \vdash \Phi\colon \sigma^v \to \tau\rrbracket^\eta_{\mathcal{T}}(\llbracket\Gamma \vdash \Psi\colon \sigma\rrbracket^\eta_{\mathcal{T}})
\end{aligned}
$$

Φ is called *well-typed* if the type judgement $\varnothing \vdash \Phi : \tau$ is derivable for some type τ. Note that we are only interested in well-typed formulas in this thesis. For those formulas where the variable types are clear from context, we omit the type of the variables.

2.3.3 Semantics of PHFL

To define the semantics of PHFL formulas we need a mapping η that associates to each variable occurring in a formula an element of its type semantics, i.e. $\eta(X) \in [\![\tau]\!]_{\mathcal{T}}$ for X of type τ. Let Φ be a well-typed formula of type τ, $\mathcal{T}$ an LTS and η a variable mapping, then the semantics $[\![\Gamma \vdash \Phi\colon \tau]\!]^{\eta}_{\mathcal{T}}$ are defined inductively on Φ as explained in Figure 2.2. The semantics of $[\![\Gamma \vdash \Phi\colon \tau]\!]^{\eta}_{\mathcal{T}}$ returns an element of $[\![\tau]\!]_{\mathcal{T}}$. Note that $\eta[X \mapsto \mathcal{X}]$ is a mapping η' that is equal to η but $\eta'(X) = \mathcal{X}$.

In this thesis we are interested in PHFL formulas that have a specific order. Thus, a formula Φ has order k if $k = max(\{ord(\tau) \mid \Psi\colon \tau \textit{ is a subformula of } \Phi\})$. The type of a well-typed formula follows from the derivation tree. The set of formulas that have at most order k is denoted by PHFL^k.

Example 2.19. *[2] The following 2-adic PHFL^0 formula $\Phi_\sim$ describes bisimilarity i.e. it denotes those pairs (q_1, q_2) such that $q_1 \sim q_2$ and vice versa.*

$$\Phi_\sim = \nu(X\colon \bullet).\ \bigwedge_{a\in\Sigma} [a]_1\langle a\rangle_2 X \wedge [a]_2\langle a\rangle_1 X \wedge \bigwedge_{p\in P} p_1 \Leftrightarrow p_2$$

Example 2.20. *[2] The following 2-adic PHFL^1 formula Φ describes trace equivalence of two states in a LTS i.e. it denotes those pairs (q_1, q_2) for which q_1 has the same traces as q_2 and vice versa.*

$$\Phi = \big(\nu(F\colon \bullet^0 \to (\bullet^0 \to \bullet)).\ \lambda(X\colon \bullet).\ \lambda(Y\colon \bullet).\ (X \Leftrightarrow Y) \wedge \bigwedge_{a\in\Sigma} F\langle a\rangle_1 X \langle a\rangle_2 Y\big)\top\top$$

Definition 2.21. Let $\mathcal{T}$ be an LTS, $\mathbf{s}$ a state tuple, η a variable mapping, Γ a type environment and φ a PHFL^k formula of type $\bullet$ then we call $\mathcal{T}$ with $\mathbf{s}$ a model of φ, written as $\mathcal{T}, \mathbf{s} \models \varphi$ iff $\mathbf{s} \in [\![\Gamma \vdash \varphi\colon \bullet]\!]^{\eta}_{\mathcal{T}}$.

Remark 2.22. To simplify PHFL formulas with fixpoint and λ operators we will use the **fixpoint unfolding principle** and **β-reduction**. As an example for β-reduction, suppose we apply the function $\lambda(X\colon \bullet).\ X \vee p_1$ to the value $\top$. To calculate the result, we substitute $\top$ for every occurrence of $\top$, and so the application of the function $(\lambda(X\colon \bullet).\ X \vee p_1)\top$ is reduced to the result

$\top \vee p_1$. As an example for fixpoint unfolding principle, suppose we have the formula $\Phi \coloneqq \mu(F\colon \bullet \to \bullet).\, \lambda(X\colon \bullet).\, X \vee F\langle a\rangle_1 X$. By using the fixpoint unfolding principle on Φ this results into $X \vee \langle a\rangle_1 X \vee \langle a\rangle_1\langle a\rangle_1 X \vee \dots$ which can an be simplified to $\bigvee_{i\geq 0}\langle a\rangle_1{}^i X$. Because we work on finite structures and the semantics of PHFL are invariant under β- reduction we can use those procedures without loss of generality.

Finally, we take a look at r-adic queries that are associated to a closed d-adic formula Φ. A PHFL formula Φ is closed if the type judgement $\varnothing \vdash \Phi\colon \bullet$ is derivable.

Definition 2.23. Given $d \in \mathbb{N}$ with $d > 0$, a type environment Γ, a variable assignment η and a closed d-adic PHFL formula Φ we call $\mathcal{Q}^r_\Phi$ the r-adic query defined by Φ if for all LTS $\mathcal{T}$, all variable mappings η and all $(q_1, \dots, q_r) \in \mathcal{Q}^r_\Phi{}^{\mathcal{T}}$ there is a $(s_1, \dots, s_d) \in [\![\Gamma \vdash \Phi\colon \bullet]\!]^\eta_{\mathcal{T}}$ such that $q_i = s_i$ for all $i \in \{1, \dots, min(\{r, d\})\}$.

For example $\mathcal{Q}^2_{\Phi_\sim}$ is the same query as in Example 2.8, where $\Phi_\sim$ is the formula from Example 2.19.

Figure 2.3: Derivation Rules for PHFL formulas that are tail-recursive.

$$\frac{}{\bar{Y} \vdash tail(p_i, \bar{X})} \qquad \frac{X \in \bar{X} \cup \bar{Y}}{\bar{Y} \vdash tail(X, \bar{X})} \qquad \frac{\bar{Y} \vdash tail(\Phi, \varnothing)}{\bar{Y} \vdash tail(\neg\Phi, \bar{X})}$$

$$\frac{\bar{Y} \vdash tail(\Phi, \bar{X})}{\bar{Y} \vdash tail(\{\mathbf{j}\}\Phi, \bar{X})} \qquad \frac{\bar{Y} \vdash tail(\Phi, \bar{X}) \qquad \bar{Y} \vdash tail(\Psi, \bar{X})}{\bar{Y} \vdash tail(\Phi \vee \Psi, \bar{X})}$$

$$\frac{\bar{Y} \vdash tail(\Phi, \bar{X})}{\bar{Y} \vdash tail(\langle a\rangle_i \Phi, \bar{X})} \qquad \frac{\bar{Y} \vdash tail(\Phi, \varnothing)}{\bar{Y} \vdash tail([a]_i \Phi, \bar{X})} \qquad \frac{\bar{Y} \cup \{Z\} \vdash tail(\Phi, \bar{X})}{\bar{Y} \vdash tail(\lambda Z^v\colon \tau.\Phi, \bar{X})}$$

$$\frac{\bar{Y} \vdash tail(\Phi, \varnothing) \qquad \bar{Y} \vdash tail(\Psi, \bar{X})}{\bar{Y} \vdash tail(\Phi \wedge \Psi, \bar{X})} \qquad \frac{\bar{Y} \vdash tail(\Phi, \bar{X}) \qquad \bar{Y} \vdash tail(\Psi, \varnothing)}{\bar{Y} \vdash tail(\Phi\, \Psi, \bar{X})}$$

$$\frac{\bar{Y} \vdash tail(\Phi, \bar{X} \cup \{Z\})}{\bar{Y} \vdash tail(\mu Z\colon \tau.\Phi, \bar{X})} \qquad \frac{\bar{Y} \vdash tail(\Phi, \bar{X} \cup \{Z\})}{\bar{Y} \vdash tail(\nu Z\colon \tau.\Phi, \bar{X})}$$

2.3.4 Tail-Recursive PHFL

Next, we want to define a fragment of PHFL formulas. This fragment is called tail-recursive and ensures that some combinations of subformulas do not appear in a PHFL formula. Therefore, let the logical connective $\wedge$, the modality operators $[a]_i$ and the greatest fixpoint operator ν be further primitives of PHFL formula syntax. Intuitively, tail-recursive PHFL formulas are PHFL formulas where all fixpoint variables neither occur freely under the operators $\neg$ and $[a]_i$ nor in Ψ of formulas of the type $\Phi\,\Psi$ or $\Psi \wedge \Phi$.

Definition 2.24. A closed PHFL formula Φ is called **tail-recursive** if $\varnothing \vdash tail(\Phi, \varnothing)$ is derivable from the rules of Figure 2.3.

The set of all tail-recursive PHFL formulas that have at most order k is denoted by PHFL^k_{tail}.

Example 2.25. *[2] Looking at Figure 2.3, we can see that* $\Phi_1 = \mu X.[a]_1 X$ *is not tail recursive, because* X *occurs under* $[a]_1$*. Moreover,* $\Phi_2 = \mu F.\lambda X.(FX) \wedge (F(FX))$ *is not tail-recursive because* F *occurs on the left side of the logical operator* $\wedge$*. The second reason why* Φ_2 *is not tail-recursive is because* F *also occurs in* FX *of subformula* $F(FX)$*. Note that all formulas that do not use fixpoints are obviously tail-recursive. An example of a formula that uses a fixpoint and is also tail-recursive is given by*

$$\Phi_3 = \big(\mu F.\, \lambda G.\, G\, p_1 \bigvee_{a \in \Sigma} F(\lambda Z.\, \langle a \rangle_1 G\, Z)\big) \lambda X.\, X$$

and denotes those states that can reach a state where property p *holds.*

2.4 Descriptive Complexity

The main aim of **descriptive complexity** is to describe the complexity classes known from computational complexity theory with logics. While computational complexity theory distinguishes time and space classes, descriptive complexity theory characterizes classes with logical resources instead of a reference to automaton models or space and time bounds.

The first known result in the area of descriptive complexity comes from Fagin. In 1974 he showed that the complexity class NP coincides with $\exists SO$ [1], the existential fragment of second-order logic.

To describe complexity classes with logics we have to explain what complexity classes are from the viewpoint of computational complexity theory. One way to describe complexity classes is with the help of *Turing Machines* [11]. A Turing Machine is a theoretical model of a machine with

a reading head moving over a tape with symbols. The reading head of those machines is positioned always over one cell of this tape and scans the symbol of the current cell. After scanning the symbol, the machine is able to override the symbol and move the reading head one cell left, right or stand still. Formally, a Turing Machine is the following.

Definition 2.26. A **Deterministic Turing Machine**(DTM) is a seven-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F, R)$, where

- Q is the finite set of states,
- Σ is the input alphabet,
- Γ is the working alphabet with $\Sigma \subset \Gamma$,
- $\delta : (Q \setminus (F \cup R)) \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$ is the transition function,
- $q_0 \in Q$ is the initial state,
- $\square \in \Gamma \setminus \Sigma$ is the blank symbol,
- $F \subseteq Q$ is the set of accepting states and
- $R \subseteq Q, F \cap R = \varnothing$ is the set of rejecting states.

Example 2.27. *As an example for a DTM let*

$$M = (\{q_0, q_f, q_r\}, \{a, b\}, \{a, b, \square\}, \delta, q_0, \square, \{q_f\}, \{q_r\})$$

where $\delta(q_0, a) = (q_0, \square, R)$, $\delta(q_0, b) = (q_r, b, L)$, $\delta(q_0, \square) = (q_f, \square, N)$. *$M$ is a DTM that accepts all input words that contain no symbol b, i.e.* $L(M) = \{a\}^*$.

Configurations are snapshots of DTMs working on an input word. This includes the working tape, the current state and the current position of the reading head. Formally, $C^M = (q, h, t)$ is called a configuration of a DTM $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ of the computation on some input word, where $q \in Q$ is the current state, $h \in \mathbb{N}$ is the reading head position and $t : \mathbb{N} \to \Gamma$ represents the full tape content. In addition, $t(i)$ represents the content of tape cell i.

Definition 2.28. Let $C_i^M = (q_i, h_i, t_i)$, $C_j^M = (q_j, h_j, t_j)$ be two configurations of a DTM $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F, R)$ with $i \neq j$. C_j^M is the next configuration of C_i^M written as $C_i^M \to_M C_j^M$ iff

- $j = i + 1$,
- $h_j = h_i + d$,
- $t_j(h_i) = a$, $t_j(k) = t_i(k)$, $k \neq h_i$,
- $\delta(q_i, t_i(h_i)) = (q_j, a, D)$

where $D \in \{L, R, N\}$ and d is -1, 1 or 0 if D is L, R or N, respectively. $C_i^M \to_M C_j^M$ is called a **transition** of M.

The start configuration for an input word $w = a_1 \dots a_n$ of a *DTM* M is $(q_0, 0, h_0)$, where q_0 is the initial state of M and $h_0(k) = a_k$ for $1 \leq k \leq n$ and $h_0(l) = \square$ for $l \neq k$. A run of *DTM* M on input word w is a sequence of transitions, $C_0^M \to_M C_1^M \to_M C_2^M \to_M \dots$. A run of *DTM* M on input word w is terminating, if there is a configuration C^M such that the state of C^M is either an accepting or rejecting state of M. A run is accepting if the state of C^M is an accepting state of M.

Example 2.29. *Let M be the DTM from Example 2.27 and $w_1 = aaba$, $w_2 = aaaa$ two input words. For better readability we illustrate tape content functions as infinite words. The run of M on w_1 is*

$$\begin{aligned} C_0^M = &(q_0, 0, aaba\square\cdots) \to_M (q_0, 1, \square aba\square\cdots) \to_M (q_0, 2, \square\square ba\square\cdots) \\ &\to_M (q_r, 1, \square\square ba\square\cdots) = C_3^M \end{aligned}$$

and the run of M on w_2 is

$$\begin{aligned} C_0^M = &(q_0, 0, aaaa\square\cdots) \to_M (q_0, 1, \square aaa\square\cdots) \to_M (q_0, 2, \square\square aa\square\cdots) \\ &\to_M (q_0, 3, \square\square\square a\square\cdots) \to_M (q_0, 4, \square\square\square\square\square\cdots) \\ &\to_M (q_f, 4, \square\square\square\square\square\cdots) = C_5^M \end{aligned}$$

Note that both runs are terminating. The run on input word w_2 is an accepting run whereas the run on input word w_1 is a rejecting one.

Note that it is possible to define *DTM*s that do not accept or reject any input words. For example, let $M = (\{q_0\}, \{a\}, \{a, \square\}, \delta, q_0, \varnothing, \varnothing)$, where $\delta(q_0, x) = (q_0, x, N)$ with $x \in \{a, \square\}$. M is a DTM where any calculation of an input word w looks as follows $q_0 w \to_M q_0 w \to_M \dots$. It never reaches an accepting or rejecting state. In this thesis, we are only interested in *DTM*s that reach an accepting or rejecting state in finite time on any input word. Those *DTM*s we are calling terminating *DTM*s. Any terminating $DTM = \{Q, \Sigma, \Gamma, \delta, q_0, F, R\}$ decides a problem, where a problem is a subset of Σ^*.

Known from computational complexity theory [7], the time and space classes can be defined by functions. These functions take as input a natural number that represents the length of an input word of a terminating *DTM*. In case of time classes the output of the functions depends on the number of configuration steps. In case of space classes the output is based on the longest transition.

Definition 2.30. Let M be a terminating *DTM*. $TIME(n) := max(STEPS(w) \mid |w| = n)$, where $STEPS(w)$ is the number of transitions of M running on input word w. $SPACE(n) := max(STORAGE(w) \mid |w| = n)$,

where $STORAGE(w) := max(h_i \mid i \in \{1, \dots m\}$ and $C_i^M = (q_i, h_i, t_i))$ is the rightmost head position of M while M runs on input word w and $C_0^M \to_M C_1^M \to_M \cdots \to_M C_m^M$ is this run of M on w.

Example 2.31. *Let w_1 and w_2 be the two words from Example 2.29 and M the terminating DTM of Example 2.27. It is easy to see that $STEPS(w_1) = 3$, $STEPS(w_2) = 5$, $STORAGE(w_1) = 2$ and $STORAGE(w_1) = 4$ for M running on w_1 and w_2. Because the maximal number of steps for an input word of length 4 occurs if the input word consists only of symbol a, it holds that $TIME(4) = 5$ or more general that $TIME(n) = n + 1$. $SPACE(n) = n$ holds, because the length of the input word will only shrink and we do not move to the right side of the word.*

It is possible to group the DTMs by functions. These groups are the computational complexity classes. In this thesis we are interested in exponential time classes and exponential space classes.

Definition 2.32. Let $f : \mathbb{N} \to \mathbb{N}$ be a polynomial function, then **exp** : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a function defined inductive as follows:

- $exp(0, f(n)) = f(n)$,
- $exp(i, f(n)) = 2^{exp(i-1, f(n))}$ for $i \geq 1$.

With the help of the function exp, we are able to define the complexity classes k-EXPTIME and k-EXPSPACE for all $k \geq 1$.

Definition 2.33. k-EXPTIME is the set of all those problems P where a DTM M and a polynomial function $f : \mathbb{N} \to \mathbb{N}$ exist such that M can decide P in $TIME(exp(k, f(n)))$. k-EXPSPACE is the set of all those problems Q where a DTM M and a polynomial function $f : \mathbb{N} \to \mathbb{N}$ exist such that M can decide Q in $SPACE(exp(k, f(n)))$.

Keep in mind that $TIME$ is the maximal number of transitions and $SPACE$ the longest configuration of a DTM when running on an input word.

An example for a problem that is in 1-EXPSPACE is the problem to recognize if two regular expressions represent different languages [12]. To check if a DTM holds in at most k steps is a problem that lies in 1-EXPTIME. In [13] it was proven that the model checking problem for HFL_{tail}^{k+1} is in k-EXPSPACE. HFL_{tail}^{k+1} is the class of all tail-recursive 1-adic PHFL formulas with order at most $k + 1$. See Section 2.3 for further information about PHFL.

As mentioned in Section 2.1 the queries defined in Definition 2.6 can be categorized. The first category is defined in Definition 2.7. Here we define a second category that describes which complexity class a query belongs to.

Because the queries are defined over LTS and the DTMs in this definition have to work on LTS, a standardized encoding of these LTS has to be used as input word. This standardized encoding transforms a given LTS with a tupel of states to a readable string.

Definition 2.34. Let $\mathcal{T}$ be an LTS with $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ and $(q_1, \ldots, q_r) \in Q^r$. A query $\mathcal{Q}$ belongs to complexity class $\mathcal{C}$ if there is a DTM (see Definition 2.26) in $\mathcal{C}$ for deciding on the standardized encoding of $(\mathcal{T}, (q_1, \ldots, q_r))$ as input whether $(q_1, \ldots, q_r) \in \mathcal{Q}^{\mathcal{T}}$.

Example 2.35. *[2] Let $\mathcal{Q}$ be the query and $\mathcal{T}$ the LTS of Example 2.8. Because bisimilarity can be decided in P[1], there is an algorithm in P for deciding on input $(\mathcal{T}, (q_1, q_2))$ whether $(q_1, q_2) \in \mathcal{Q}^{\mathcal{T}}$ that means $\mathcal{Q}$ belongs to complexity class P.*

From Definition 2.7, Definition 2.34 and the complexity classes k-EXPTIME and k-EXPSPACE of Definition 2.33 the queries we want to investigate can be desired.

Definition 2.36. k-EXPTIME/$\sim$ are the bisimulation invariant queries that belong to complexity class k-EXPTIME and k-EXPSPACE/$\sim$ are the bisimulation invariant queries that belong to complexity class k-EXSPACE, where $k \geq 0$.

In the introduction of this section, we mentioned that the main aim of descriptive complexity is to describe complexity classes with logics. The next definition defines how a complexity class is captured by a logic.

Definition 2.37. A complexity class $\mathcal{C}$ is captured by a logic L if for all queries $\mathcal{Q}$ that belongs to $\mathcal{C}$ there is a formula Φ of L such that for all LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ there is a variable mapping η and it holds that $(q_1, \ldots, q_r) \in \mathcal{Q}^{\mathcal{T}}$ iff $(q_1, \ldots, q_r) \in [\![\Phi]\!]^{\eta}_{\mathcal{T}}$, where $q_1, \ldots, q_r \in Q$ and $[\![\Phi]\!]^{\eta}_{\mathcal{T}}$ is the semantics of Φ under $\mathcal{T}$ and η.

[1] Bisimilarity between two states can be checked by a special kind of breadth-first search. We assume that all states are bisimilar and removing all those pairs where the states have not the same properties. The resulting set is called R. On each step we check any pair of R. Any pair (q, p) has to satisfy the following. If looking at q all possible actions leading to any state q' can also be duplicated from p with same action leading to a state p' and (q', p') is in R. The same will be checked looking at the right state first. If the pair (q, p) does not satisfy these conditions it will be removed from R. This iteration proceeds as long as at least one pair is removed from R. Note that the state set of an LTS is finite. Those cardinality let be denoted with n. Since on each iteration process at least one tuple will be removed this process terminates after a finite amount of steps. In more detail the algorithm checks in the first step n^2 tuples, in the second $n^2 - 1$ and so on. In worst case that is a total of $(n^4 + n^2)/2$ checks.

2.5 Higher Order Logic

For comparing the complexity classes with PHFL, we use combinations of extensions of FO as intermediate logics. The first well known extension is called Higher Order Logic [14], abbreviated with HO. In HO we increase the expressive power of FO by allowing quantification over relations of any order. Therefore, we have to define the types of higher order variables.

2.5.1 Syntax of HO

Definition 2.38. An **HO type** τ is defined inductively as follows

$$\tau ::= \odot \text{ or } \tau := (\tau', \dots, \tau'),$$

where τ' is also an HO type.

The HO type of individuals is $\tau = \odot$. These objects have *order* 1. The HO type $\tau = (\tau', \dots, \tau')$ is that of relations between objects of HO type τ' and has order $1+order(\tau')$. For each HO type we have a countable infinite set of variables. Furthermore, let σ be a signature over a relational vocabulary i.e. σ only contains relation symbols.

Definition 2.39. Let $\mathcal{V} = \{X_1, X_2, \dots\}$ a countable infinite set of variables, P a set of propositions and Σ a set of actions, then **HO formulas** over P and Σ are defined by the grammar

$$\Phi, \Psi ::= p(X_1) \mid a(Y_1, Y_2) \mid Y(Z_1, \dots, Z_n) \mid \neg\Phi \mid \Phi \vee \Psi \mid \exists(X : \tau).\, \Phi$$

where

- $p \in P$ and $X_1 \in \mathcal{V}$ of HO type $\odot$,
- $a \in \Sigma$ and $Y_1, Y_2 \in \mathcal{V}$ of HO type $\odot$,
- $Y \in \mathcal{V}$ of HO type $(\tau', \dots, \tau')$ and $Z_1, \dots, Z_n \in \mathcal{V}$ of HO type τ' and
- $X \in \mathcal{V}$ of HO type τ.

2.5.2 Semantics of HO

At first, we define the universes of the different HO types.

Definition 2.40. Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS then the universes of the HO types are defined inductively as follows:

- $D_{\odot}(Q) = Q$,
- $D_{(\tau,\dots,\tau)}(Q) = \mathcal{P}(D_{\tau}(Q)^n)$,

where $(\tau, \dots, \tau)$ is a n-tuple of type τ.

Moreover, η is a variable mapping that assigns every variable to an element of the appropriate universe, i.e. if variable X is of HO type τ, then $\eta(X) \in D_\tau(Q)$. With $\eta[X \to \mathcal{X}]$, where $\mathcal{X} \in D_\tau(Q)$ and X of HO type τ, we mean the variable assignment η', where $\eta'(X) = \mathcal{X}$ and $\eta'(Y) = \eta(Y)$ for all $Y \neq X$.

Definition 2.41. Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS and η a variable mapping over universe Q. The semantics of an HO formula is defined inductively as follows:

- $\mathcal{T}, \eta \models p(X_1)$ iff $p \in v(\eta(X_1))$,
- $\mathcal{T}, \eta \models a(Y_1, Y_2)$ iff $(\eta(Y_1), a, \eta(Y_2)) \in \Delta$,
- $\mathcal{T}, \eta \models Y(Z_1, \dots Z_n)$ iff $(\eta(Z_1), \dots \eta(Z_n)) \in \eta(Y)$,
- $\mathcal{T}, \eta \models \neg\Phi$ iff $\mathcal{T}, \eta \not\models \Phi$,
- $\mathcal{T}, \eta \models \Phi \vee \Psi$ iff $\mathcal{T}, \eta \models \Phi$ or $\mathcal{T}, \eta \models \Psi$,
- $\mathcal{T}, \eta \models \exists(X : \tau).\, \Phi$ iff there exists $\mathcal{X} \in D_\tau(Q)$ with $\mathcal{T}, \eta[X \to \mathcal{X}] \models \Phi$

We can categorize the formulas by the order of all occurring variables. With HO^k we mean the set of all those formulas whose variables have order less or equal k. Similar to Definition 2.23 we define r-adic queries that are defined by HO formulas with f free first-order variables.

Definition 2.42. Given a set of propositions P, a set of actions a and a closed HO formula Φ with free first-order variables $X_1, \dots, X_f$ we call $\mathcal{Q}^r_\Phi$ a r-adic query defined by Φ if for all LTS $\mathcal{T}$, all variable mappings η and all $(q_1, \dots, q_r) \in {\mathcal{Q}^r_\Phi}^{\mathcal{T}}$ there is a $\mathcal{T}, \eta \models \Phi$ such that $q_i = \eta(x_i)$ for all $i \in \{1, \dots, min(\{r, f\})\}$.

Finally, we define bisimulation-invariant HO formulas.

Definition 2.43. Given an HO formula Φ with f free first-order variables and the f-adic query $\mathcal{Q}^f_\Phi$ defined by Φ then we call Φ bisimulation-invariant if $\mathcal{Q}^r_\Phi$ is bisimulation-invariant.

2.5.3 HO + LFP

Another possibility to extend FO is to add operators that are not expressible in FO. Here, we are interested in two of them, the least fixpoint and the partial fixpoint operator. Instead of defining the operators for FO we want to define these operators for HO. At first, we regard the least fixpoint operator. Like in [5] we want to define special operators that are working on HO type universes.

Definition 2.44. Let P be an set of propositions, Σ a set of actions, X a relation variable of HO type $\tau = (\tau', \ldots, \tau')$, τ' an arbitrary HO type, $X_1, \ldots X_n$ variables of HO type τ' and $\varphi(X, X_1, \ldots, X_k)$ a formula over P and Σ with free variables $X, X_1, \ldots, X_k$. For each LTS $\mathcal{T}$ with state set Q and each variable mapping η, the formula $\varphi(X, X_1, \ldots, X_k)$ induces the operator

$$F_\Phi^{\mathcal{T},\eta} \colon \mathscr{P}(D_\tau(Q)) \longrightarrow \mathscr{P}(D_\tau(Q))$$
$$A \longmapsto F_\Phi^{\mathcal{T}}(A) \coloneqq \{(A_1, \ldots, A_n) \mid \mathcal{T}, \eta \models \varphi(A, A_1, \ldots, A_n)\}$$

where $A_1 \ldots, A_n \in D_{\tau'}(Q)$.

To make $F_\Phi^{\mathcal{T},\eta}$ monotone we have to restrict $\varphi(X, X_1, \ldots, X_k)$ in a way that variable X occurs under an even number of negations within of Φ [5]. Those formulas are called *positive in* X. Therefore, we are able to define the least fixpoint operator for HO formulas, denoted by HO(LFP).

Definition 2.45. Let P be a set of propositions and Σ a set of actions. The set of **HO(LFP) formulas** enhances the set of HO formulas with the following formation rule:

- $[LFP\ \Phi(X, X_1, \ldots, X_n)](V_1, \ldots, V_n)$ is an HO (LFP) formula over P and Σ with free variables $V_1, \ldots, V_n$ iff $\Phi(X, X_1, \ldots, X_n)$ is an HO(LFP) formula with free variables $X, X_1, \ldots, X_n$, if Φ is positive in X, variable X has HO type $\tau = (\tau', \ldots, \tau')$ and $X_1, V_1, \ldots, X_n, V_n$ have HO type τ'.

Similar to HO^k, the set HO(LFP)^k denotes all those HO(LFP) formulas whose variables have types of order less or equal k with one exception: The variable X occurring in formulas of kind $[LFP\ \varphi(X, x_1, \ldots, x_n)](v_1, \ldots, v_n)$ is the only one that can have a type of an order $k + 1$.

Definition 2.46. Let $\mathcal{T}$ be an LTS and η a variable mapping over universe Q. The semantics of an HO formula are extended to the semantics of HO(LFP) via the following definition:

- $\mathcal{T}, \eta \models [LFP\ \varphi(X, X_1, \ldots, X_n)](V_1, \ldots, V_n)$ iff $(\eta(V_1), \ldots, \eta(V_n)) \in LFP\ (F_\varphi^{\mathcal{T},\eta})$.

Example 2.47. *Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS. Furthermore, let*

$$\begin{aligned}\Phi(X, X_1, X_2) = &\bigvee_{p \in P} \big(p(X_1) \neq p(X_2)\big) \\ &\vee \bigvee_{a \in \Sigma} \big(\exists(Y_1 \colon \odot).\, a(X_1, Y_1) \wedge \forall(Y_2 \colon \odot).\, a(X_2, Y_2) \Rightarrow X(Y_1, Y_2)\big) \\ &\vee \bigvee_{a \in \Sigma} \big(\exists(Y_2 \colon \odot).\, a(X_2, Y_2) \wedge \forall(Y_1 \colon \odot).\, a(X_1, Y_1) \Rightarrow X(Y_1, Y_2)\big)\end{aligned}$$

be an HO^2 formula with free first-order variables x_1 and x_2 and free second-order variable X. Then $LFP(F_\Phi^{\mathcal{T},\eta})$ describes the same set as in Example 2.10, the set of all those pairs of states (q,p) in $\mathcal{T}$ such that $q \not\sim p$ holds. With $[LFP\ \Phi(X,X_1,X_2)](q,p)$ we can check if $q \not\sim p$, where $q,p \in Q$.

2.5.4 HO + PFP

Next, we define the partial fixpoint operator for HO formulas [15]. While the *LFP* operator restricts formulas to be positive in a variable, the partial fixpoint operator does not have any restriction. Therefore, we can define and add the partial fixpoint operator to HO formulas, denoted as HO(PFP).

Definition 2.48. Let P be a set of propositions and Σ a set of actions. The set of **HO(PFP) formulas** enhances the set of HO formulas with the following formation rule:

- $[PFP\ \Phi(X,X_1,\ldots,X_n)](V_1,\ldots,V_n)$ is an HO (PFP) formula over P and Σ with free variables $V_1,\ldots,V_n$ iff $\Phi(X,X_1,\ldots,X_n)$ is an HO(PFP) formula with free variables $X,X_1,\ldots,X_n$, where X has HO type $\tau = (\tau',\ldots,\tau')$ and $X_1,V_1,\ldots,X_n,V_n$ have HO type τ'.

HO(PFP)^k denotes the set of all those HO(PFP) formulas whose variables have types of order less or equal k with one exception: The variable X occurring in formulas of kind $[PFP\ \varphi(X,X_1,\ldots,X_n)](V_1,\ldots,V_n)$ is the only one that can have a type of an order $k+1$.

Definition 2.49. Let $\mathcal{T}$ be an LTS and η a variable mapping over universe Q. The semantics of an HO formula are extended to the semantics of HO(PFP) via the following definition:

- $\mathcal{T},\eta \models [PFP\ \varphi(X,X_1,\ldots,X_n)](V_1,\ldots,V_n)$ iff $(\eta(V_1),\ldots,\eta(V_n)) \in PFP\ (F_\Phi^{\mathcal{A},\eta})$.

Example 2.50. *Let $\mathcal{T} = (Q,\{a\},P,\Delta,v)$ be an LTS with $Q = \{1,\ldots,n\}$ and $\Delta = \{(x,a,y) \mid x,y \in Q \text{ and } x < y\}$. Furthermore, let*

$$\Phi(X,X_1) = X(X_1) \wedge \exists(X_2 \colon \odot).\, a(X_2,X_1) \wedge X(X_2) \vee \\ \neg X(X_1) \wedge \forall(X_2 \colon \odot).\, a(X_2,X_1) \Rightarrow X(X_2)$$

be an HO^1 formula. Then $F_\Phi^{\mathcal{T},\eta}$ describes the same operator as in Example 2.13 and therefore $PFP(F_\Phi^{\mathcal{A},\eta}) = \varnothing$ holds. Note that $\mathcal{T},\eta \not\models [PFP\ \Phi(X, X_1)](V_1)$ for any variable mapping η.

Chapter 3

Upper Bounds

In this chapter we regard the upper bounds of the expressive power of PHFL^k and PHFL$^k_{tail}$. First, we have to show that the upper bound of the expressive power of PHFL^k is k-EXPTIME/$\sim$.

3.1 Upper Bound of PHFL^k

To show that the upper bound of the expressive power of PHFL^k is k-EXPTIME/$\sim$ we reduce the model checking problem of PHFL^k that to HFL^k. Remember that HFL^k is the set of 1-adic PHFL^k formulas. In combination with Theorem 3.1 we get the upper bound of the expressive power of PHFL^k.

Theorem 3.1. *[16] The model checking problem of HFL^k is solvable in k-EXPTIME, i.e. on input $\mathcal{T}, s$ and Φ it is decidable in k-EXPTIME whether $\mathcal{T}, s$ is a model of Φ, where $\mathcal{T}$ is an LTS, s is a state and Φ in HFL^k.*

In the following, we have to reduce the semantics of PHFL^k to that of HFL^k. Thus, we have to convert the input LTS $\mathcal{T}$ and the input PHFL^k formula Φ of the problem of deciding whether $\mathcal{T}$ with a state tuple $\mathbf{s}$ is a model of Φ. We first define a mapping that transforms an LTS into the d-fold product of itself, where $d \in \mathbb{N}$ and give an example for this transformation process. In the next step we define a function that maps a PHFL^k type to an HFL^k type. We then continue to define one additional function that maps a PHFL^k formulas to HFL^k formulas and give an example for such a mapping. Finally, we show that the semantics of the original formula with the original types and the original LTS in the PHFL^k context coincide with the semantics of the converted formula with the converted types and the converted LTS in the HFL^k context[1].

[1]Because the new LTS will have as underlying set d-tuples of states of the original LTS,

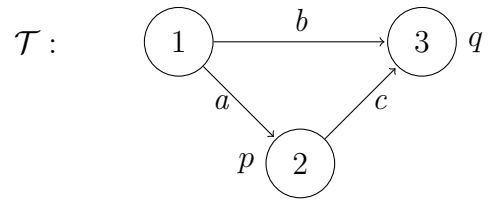
As mentioned above we do start by defining a mapping that transforms a given LTS.

Definition 3.2. Let $d \in \mathbb{N}$ and $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ an LTS, then $\mathcal{T}_d = (Q^d, \Sigma_d \cup S_d, P_d, \Delta_d, v_d)$, where

- $\Sigma_d = \bigcup_{a \in \Sigma} (\bigcup_{i=1}^{d} \{a_i\})$,
- $S_d = \{s_{(e(1),\ldots,e(d))} \mid e : \{1, \ldots d\} \to \{1, \ldots, d\}\}$,
- $P_d = \bigcup_{p \in P} (\bigcup_{i=1}^{d} \{p_i\})$,
- $\Delta_d = \{((q_1, \ldots, q_{i-1}, q_i, q_{i+1}, \ldots, q_d), a_i, (q_1, \ldots, q_{i-1}, {q_i}', q_{i+1}, \ldots, q_d)) \mid (q_i, a, {q_i}') \in \Delta\}$
  $\cup \{((q_{e(1)}, \ldots, q_{e(d)}), s_{(e(1),\ldots,e(d))}, (q_1, \ldots, q_d)) \mid e : \{1, \ldots, d\} \to \{1, \ldots, d\}\}$ and
- $v_d \colon Q^d \to 2^{P_d}$, $v_d((q_1, \ldots, q_d)) = \bigcup_{i=1}^{d} \{p_i \mid p \in v(q_i)\}$.

The following example shows the construction of an LTS as described in Definition 3.2.

Example 3.3. *Let $\mathcal{T}$ be an LTS given by*



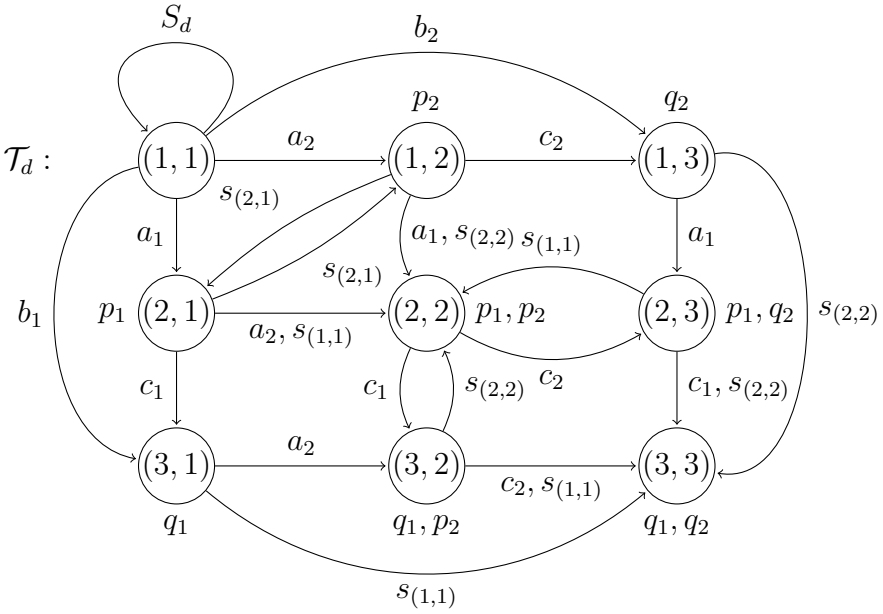
Let $d = 2$ be a natural number then $\mathcal{T}_d$ is the LTS at Figure 3.1. Note that for readability reasons not all edges are drawn in this representation of $\mathcal{T}_d$. The edges that are missing are those that uses action $s_{(1,2)}$ (except $(1,1)$ to $(1,1)$) and the following edges are also not drawn $\{(1,2) \overset{s_{(1,1)}}{\to} (1,1), (1,3) \overset{s_{(1,1)}}{\to} (1,1), (1,3) \overset{s_{(2,1)}}{\to} (3,1), (2,1) \overset{s_{(2,2)}}{\to} (1,1), (2,2) \overset{S_d}{\to} (2,2), (2,3) \overset{s_{(2,1)}}{\to} (3,2), (3,1) \overset{s_{(2,1)}}{\to} (1,3), (3,1) \overset{s_{(2,2)}}{\to} (1,1), (3,2) \overset{s_{(2,1)}}{\to} (2,3), (3,3) \overset{S_d}{\to} (3,3), (1,2) \overset{b_1}{\to} (3,2), (1,3) \overset{b_1}{\to} (3,3), (2,1) \overset{b_2}{\to} (2,3), (3,1) \overset{b_2}{\to} (3,3)\}$, where $q \overset{S_d}{\to} q' = \{q \overset{a}{\to} q' \mid a \in S_d\}$.

The next step is to define a function that maps a PHFL type to an HFL type.

---

the semantics of both logics represents the same mathematically object.

Figure 3.1: Construction of an LTS for $d = 2$ with respect to Definition 3.2.



Definition 3.4. Let $\tau_{PHFL}, \sigma_{PHFL}$ be arbitrary PHFL types, $\bullet_{PHFL}$ the base type of PHFL, $\bullet_{HFL}$ the base type of HFL and v an arbitrary variance, then T is a function that maps a PHFL type to an HFL type defined inductive over the type of PHFL as follows:

$$T(\bullet_{PHFL}) = \bullet_{HFL},$$
$$T(\sigma^v_{PHFL} \to \tau_{PHFL}) = T(\sigma_{PHFL})^v \to T(\tau_{PHFL})$$

The type function T of Definition 3.4 can be adapted on type environments. If $\Gamma = X_1^{v_1} \colon \tau_1, \ldots, X_n^{v_n} \colon \tau_n$ is a type environment, then $T(\Gamma) = X_1^{v_1} \colon T(\tau_1), \ldots, X_n^{v_n} \colon T(\tau_n)$.

Definition 3.5. Let $\mathcal{T}$ be an LTS, $d \in \mathbb{N}$ with $d > 0$ and $\mathcal{T}_d$ the transformed LTS by Definition 3.2 and T the type mapping of Definition 3.4. Furthermore, let η be a variable mapping over $\mathcal{T}$ for PHFL formulas, then $V(\eta)$ is a variable mapping over $\mathcal{T}_d$ for HFL formulas, where $V(\eta)(X) = \eta(X) = \mathcal{X}$ with $\mathcal{X} \in [\![\tau]\!]_{\mathcal{T}}$. Note that for $\mathcal{X}$ it holds that $\mathcal{X} \in [\![\tau]\!]_{\mathcal{T}}$ iff $\mathcal{X} \in [\![T(\tau)]\!]_{\mathcal{T}_d}$ due to the definition of $\mathcal{T}_d$ and $T(\tau)$.

We continue with the definition of the function that maps a PHFL^k formula to an HFL^k formula.

Definition 3.6. Let T be the type function from Definition 3.4 and let P be a set of propositions, Σ a set of actions and $\mathcal{V} = \{X_1, \ldots, X_n\}$ a finite set of variables for a d-adic PHFL^k formula Φ, then F is a function that maps a d-adic PHFL^k formula over P, Σ and $\mathcal{V}$ to an HFL^k formula over proposition set $P_d = \bigcup_{p \in P} (\bigcup_{i=1}^{d} \{p_i\})$, action set $\Sigma_d \cup S_d = \bigcup_{a \in \Sigma} (\bigcup_{i=1}^{d} \{a_i\}) \cup \{s_{(e(1),\ldots,e(d))} \mid e : \{1, \ldots d\} \to \{1, \ldots, d\}\}$ and variable set $\mathcal{V}$ which is defined inductive over the d-adic PHFL^k formula as follows:

$$
\begin{aligned}
F(\top) &= \top \\
F(X) &= X \\
F(p_i) &= p_i \\
F(\langle a \rangle_i \psi) &= \langle a_i \rangle F(\psi) \\
F(\psi \vee \psi') &= F(\psi) \vee F(\psi') \\
F(\neg \psi) &= \neg F(\psi) \\
F(\{\mathbf{j}\}\psi) &= \langle s_{\mathbf{j}} \rangle F(\psi) \\
F(\mu(X \colon \tau).\, \psi) &= \mu(X \colon T(\tau)).\, F(\psi) \\
F(\lambda(X^v \colon \tau).\, \psi) &= \lambda(X^v \colon T(\tau)).\, F(\psi) \\
F(\psi\, \psi') &= F(\psi)\, F(\psi')
\end{aligned}
$$

Example 3.7. *We transform the 2-adic PHFL^2 formula of Example 2.20 with the function of Definition 3.6 to an HFL^2 formula.*

$$
\begin{aligned}
\Phi = &(\mu(F \colon \bullet^0_{PHFL} \to (\bullet^0_{PHFL} \to \bullet_{PHFL})).\, \lambda(X \colon \bullet_{PHFL}).\, \lambda(Y \colon \bullet_{PHFL}).\\
&X \Leftrightarrow Y \wedge \bigwedge_{a \in \Sigma} F\langle a \rangle_1 X \langle a \rangle_2 Y) \top \top
\end{aligned}
$$

will be transformed to

$$
\begin{aligned}
F(\Phi) = &(\mu(F \colon \bullet^0_{HFL} \to (\bullet^0_{HFL} \to \bullet_{HFL})).\, \lambda(X \colon \bullet_{HFL}).\, \lambda(Y \colon \bullet_{HFL}).\\
&X \Leftrightarrow Y \wedge \bigwedge_{a \in \Sigma} F\langle a_1 \rangle X \langle a_2 \rangle Y) \top \top.
\end{aligned}
$$

Remark 3.8. It holds for type environment Γ and PHFL^k formula Φ of PHFL^k type τ that if $\Gamma \vdash \Phi \colon \tau$ is derivable, then $T(\Gamma) \vdash F(\Phi) \colon T(\tau)$ is also derivable. This statement can easily be proven by induction over the structure of formula Φ.

Because the type judgement is always derivable we ignore the type environment in the following proof and write just $[\![\Phi]\!]^\eta_{\mathcal{T}}$ instead of $[\![\Gamma \vdash \Phi \colon \tau]\!]^\eta_{\mathcal{T}}$, where Φ is a PHFL formula, η is a variable mapping, $\mathcal{T}$ is an LTS, Γ is a type environment and τ is a PHFL type.

3.1.1 Correctness Proof

In the final step we show that the semantics of a given PHFL^k formula coincides with the transformed HFL^k formula.

Lemma 3.9. *Let $\mathcal{T}$ be an LTS, η a variable mapping, Φ a well-typed d-adic PHFL^k formula, $\mathcal{T}_d$ the LTS transformed by the process of Definition 3.2, T the type function of Definition 3.4, V the variable mapping function of Definition 3.5 and F the formula function of Definition 3.6 then $[\![\Phi]\!]^\eta_\mathcal{T} = [\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}$*

Although it looks unexpected this is exactly the statement that we have to prove. The semantics of a PHFL formula is a set of tuples but in the HFL formula this is just a set. Because this set is the set of states of $\mathcal{T}_d$ and this is a set of tuples, the both semantics operates on the same kind of sets.

Proof. We show that $[\![\Phi]\!]^\eta_\mathcal{T} = [\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}$ by induction on formula Φ.

- In case of $\Phi = \top$, $[\![\Phi]\!]^\eta_\mathcal{T}$ is the set of d-tuples of state set Q of $\mathcal{T}$ what means that
$$[\![\Phi]\!]^\eta_\mathcal{T} = Q^d.$$
By construction of $\mathcal{T}_d$ the set of states is also the set of d-tuples of state set Q. Moreover, due to formula function F it is true that $F(\top) = \top$. It holds that
$$[\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d} = Q^d$$
because the set of states of $\mathcal{T}_d$ is Q^d.
- In case of $\Phi = p_i$,
$$[\![\Phi]\!]^\eta_\mathcal{T} = \{(q_1, \ldots, q_d) \in Q^d \mid p \in v(q_i)\}.$$
By construction of $\mathcal{T}_d$ it holds that $v_d((q_1, \ldots, q_d)) = \bigcup_{i=1}^{d} \{p_i \mid p \in v(q_i)\}$ and by formula function F it is true that $F(p_i) = p_i$. Because $p_i \in v_d((q_1, \ldots, q_d))$ iff $p \in v(q_i)$ it holds that
$$[\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d} = \{(q_1, \ldots, q_d) \in Q^d \mid p_i \in v_d((q_1, \ldots, q_d))\}$$
and furthermore that
$$\{(q_1, \ldots, q_d) \in Q^d \mid p \in v(q_i)\}$$
is equal to
$$\{(q_1, \ldots, q_d) \in Q^d \mid p_i \in v_d((q_1, \ldots, q_d))\}.$$

- The last base case is $\Phi = X$. It holds that

$$[\![\Phi]\!]^{\eta}_{\mathcal{T}} = \eta(X).$$

  Moreover, it is true that

$$[\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d} = V(\eta)(X).$$

  By definition of the function F it is also true that $F(X) = X$. The combination of construction $\mathcal{T}_d$, type function T and by construction of variable mapping $V(\eta)$ in Definition 3.5 it follows that $V(\eta)(X) = \eta(X)$.

The induction hypothesis is that it holds for all subformulas ψ and ψ' of Φ that $[\![\psi]\!]^{\eta}_{\mathcal{T}} = [\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d}$ and $[\![\psi']\!]^{\eta}_{\mathcal{T}} = [\![F(\psi')]\!]^{V(\eta)}_{\mathcal{T}_d}$

- In case of $\Phi = \langle a \rangle_i \psi$,

$$\begin{aligned}[\![\Phi]\!]^{\eta}_{\mathcal{T}} = \{&(q_1, \ldots, q_d) \in Q^d \mid \\ &\text{there exists } (q_1{}', \ldots, q_d{}') \in [\![\Gamma \vdash \psi]\!]^{\eta}_{\mathcal{T}} \text{ such that} \\ &q_i \xrightarrow{a} q_i{}' \text{ and for all } i \neq j \text{ it holds that } q_j = q_j{}'\}.\end{aligned}$$

  By the induction hypothesis $(q_1{}', \ldots, q_d{}') \in [\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d}$ because $(q_1{}', \ldots, q_d{}') \in [\![\psi]\!]^{\eta}_{\mathcal{T}}$. Based on construction of $\mathcal{T}_d$ and $q_i \xrightarrow{a} q_i{}' \in \Delta$ it follows that $(q_1{}', \ldots, q'_{i-1}, q_i, q'_{i+1}, \ldots, q_d{}') \xrightarrow{a_i} (q_1{}', \ldots, q'_{i-1}, q_i{}', q'_{i+1}, \ldots, q_d{}') \in \Delta_d$. That means that

$$\begin{aligned}\{(q_1, \ldots, q_d) \in Q^d \mid &\text{there exists } (q_1{}', \ldots, q_d{}') \in [\![\psi]\!]^{\eta}_{\mathcal{T}} \text{ such that} \\ &q_i \xrightarrow{a} q_i{}' \text{ and for all } i \neq j \text{ it holds that } q_j = q_j{}'\}\end{aligned}$$

  is equal to

$$\begin{aligned}\{(q_1, \ldots, q_d) \in Q^d \mid &\text{there exists } (q_1{}', \ldots, q_d{}') \in [\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d} \text{ such that} \\ &q_i \xrightarrow{a_i} q_i{}' \text{ and for all } i \neq j \text{ it holds that } q_j = q_j{}'\}.\end{aligned}$$

  As a result of of T, V and that $F(\langle a \rangle_i \psi) = \langle a_i \rangle \psi$ holds the second set is exactly the definition of the semantics of

$$[\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}.$$

- In case of $\Phi = \psi \vee \psi'$,

$$[\![\Phi]\!]^{\eta}_{\mathcal{T}} = [\![\psi]\!]^{\eta}_{\mathcal{T}} \sqcup_{\bullet} [\![\psi']\!]^{\eta}_{\mathcal{T}}.$$

By the induction hypothesis it holds that

$$[\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d} = [\![\psi]\!]^{\eta}_{\mathcal{T}}$$

and

$$[\![F(\psi')]\!]^{V(\eta)}_{\mathcal{T}_d} = [\![\psi']\!]^{\eta}_{\mathcal{T}}.$$

By construction of T it is true that

$$[\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d} \sqcup_{T(\bullet)} [\![F(\psi')]\!]^{V(\eta)}_{\mathcal{T}_d}$$

is equal to

$$[\![\psi \colon \tau]\!]^{\eta}_{\mathcal{T}} \sqcup_{\bullet} [\![\psi']\!]^{\eta}_{\mathcal{T}}.$$

Because $F(\psi \vee \psi') = F(\psi) \vee F(\psi')$ one can see that

$$[\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}$$

is equal to

$$[\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d} \sqcup_{T(\bullet)} [\![F(\psi')]\!]^{V(\eta)}_{\mathcal{T}_d}.$$

- In case of $\Phi = \neg\psi$,

$$[\![\Phi]\!]^{\eta}_{\mathcal{T}} = Q^d \setminus [\![\psi]\!]^{\eta}_{\mathcal{T}}.$$

By the induction hypothesis it holds that

$$[\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d} = [\![\psi]\!]^{\eta}_{\mathcal{T}}.$$

Due to the equality of the two sets it follows that

$$Q^d \setminus [\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d} = Q^d \setminus [\![\psi]\!]^{\eta}_{\mathcal{T}}.$$

Since $F(\neg\psi) = \neg F(\psi)$ the first set is exactly the semantics of

$$[\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}.$$

- In case of $\Phi = \{(e(1), \ldots, e(d))\}\,\psi$,

$$[\![\Phi]\!]^{\eta}_{\mathcal{T}} = \{(q_1, \ldots, q_d) \in Q^d \mid (q_{e(1)}, \ldots, q_{e(d)}) \in [\![\psi]\!]^{\eta}_{\mathcal{T}}\}.$$

By the induction hypothesis $(q_{e(1)}, \ldots, q_{e(d)}) \in [\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d}$ because $(q_{e(1)}, \ldots, q_{e(d)}) \in [\![\psi]\!]^{\eta}_{\mathcal{T}}$. Moreover, the state tuple $(q_{e(1)}, \ldots, q_{e(d)})$ that fulfills ψ is reached by 'moving' from state tuple $(q_1, \ldots, q_d)$ to $(q_{e(1)}, \ldots, q_{e(d)})$. This movement is integrated in the construction of $\mathcal{T}_d$. There exists for each endomorphism e a substitution action $s_{(e(1),\ldots,e(d))}$ for each tuple

state. With $F(\{(e(1),\ldots,e(d))\}\psi) = \langle s_{(e(1),\ldots,e(d))}\rangle F(\psi)$ we receive the state tuples that have the action $(q_1,\ldots,q_d) \overset{s_{(e(1),\ldots,e(d))}}{\to} (q_{e(1)},\ldots,q_{e(d)})$ where $(q_1,\ldots,q_d) \in [\![F(\psi)]\!]^{V(\eta)}_{\mathcal{T}_d}$. It follows that

$$[\![\Phi]\!]^{\eta}_{\mathcal{T}} = [\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}.$$

- In case of $\Phi = \mu(X\colon \tau).\,\psi$,

$$[\![\Phi]\!]^{\eta}_{\mathcal{T}} = \bigsqcap_{[\![\tau]\!]_{\mathcal{T}}} \{\mathcal{X} \in [\![\tau]\!]_{\mathcal{T}} \mid [\![\psi]\!]^{\eta[X\mapsto\mathcal{X}]}_{\mathcal{T}} \leq_{[\![\tau]\!]_{\mathcal{T}}} \mathcal{X}\}.$$

  By the induction hypothesis it applies that

$$[\![F(\psi)]\!]^{V(\eta)[X\mapsto\mathcal{X}]}_{\mathcal{T}_d} = [\![\psi]\!]^{\eta[X\mapsto\mathcal{X}]}_{\mathcal{T}}.$$

  Moreover, $F(\mu(X\colon \tau).\psi) = \mu(X\colon T(\tau)).F(\psi)$. In particular, it holds that

$$[\![\Phi]\!]^{\eta}_{\mathcal{T}} = \bigsqcap_{[\![\tau]\!]_{\mathcal{T}_d}} \{\mathcal{X} \in [\![\tau]\!]_{\mathcal{T}_d} \mid [\![F(\psi)]\!]^{V(\eta)[X\mapsto\mathcal{X}]}_{\mathcal{T}_d} \leq_{[\![\tau]\!]_{\mathcal{T}_d}} \mathcal{X}\}.$$

  This is exactly the semantics of

$$[\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}.$$

- In case of $\Phi = \lambda(X^v\colon \sigma).\,\psi$,

$$[\![\Phi\colon \sigma^v \to \tau]\!]^{\eta}_{\mathcal{T}} = F \in [\![\sigma^v \to \tau]\!]_{\mathcal{T}}$$

  such that for all $\mathcal{X} \in [\![\sigma]\!]_{\mathcal{T}}$ it holds that $F(\mathcal{X}) = [\![\psi\colon \tau]\!]^{\eta[X\mapsto\mathcal{X}]}_{\mathcal{T}}$. By the induction hypothesis it is true that

$$[\![F(\psi)\colon T(\tau)]\!]^{V(\eta)[X\mapsto\mathcal{X}]}_{\mathcal{T}_d} = [\![\psi\colon \tau]\!]^{\eta[X\mapsto\mathcal{X}]}_{\mathcal{T}}.$$

  Moreover, $F(\lambda(X^v\colon \sigma).\psi) = \lambda(X^v\colon T(\sigma)).F(\psi)$. Consequently it holds that

$$[\![\Phi\colon \sigma^v \to \tau]\!]^{\eta}_{\mathcal{T}} = F \in [\![T(\sigma^v \to \tau)]\!]_{\mathcal{T}_d}$$

  such that for all $\mathcal{X} \in [\![T(\sigma)]\!]_{\mathcal{T}_d}$ the following is true

$$F(\mathcal{X}) = [\![F(\psi)\colon T(\tau)]\!]^{V(\eta)[X\mapsto\mathcal{X}]}_{\mathcal{T}_d}.$$

  This is exactly the semantics of

$$[\![F(\Phi)\colon T(\sigma^v \to \tau)]\!]^{V(\eta)}_{\mathcal{T}_d}.$$

- In case of $\Phi = \psi\,\psi'$,

$$[\![\Phi\colon \tau]\!]^\eta_{\mathcal{T}} = [\![\psi\colon \sigma^v \to \tau]\!]^\eta_{\mathcal{T}}([\![\psi'\colon \sigma]\!]^\eta_{\mathcal{T}}).$$

  By the induction hypothesis it applies that

$$[\![F(\psi)\colon T(\sigma^v \to \tau)]\!]^{V(\eta)}_{\mathcal{T}_d} = [\![\Gamma \vdash \psi\colon \sigma^v \to \tau]\!]^\eta_{\mathcal{T}}$$

  and

$$[\![F(\psi')\colon T(\sigma)]\!]^{V(\eta)}_{\mathcal{T}_d} = [\![\Gamma \vdash \psi'\colon \sigma]\!]^\eta_{\mathcal{T}}.$$

  It follows that

$$[\![F(\psi)\colon T(\sigma^v \to \tau)]\!]^{V(\eta)}_{\mathcal{T}_d}([\![F(\psi')\colon T(\sigma)]\!]^{V(\eta)}_{\mathcal{T}_d}$$

  is equal to

$$[\![\psi\colon \sigma^v \to \tau]\!]^\eta_{\mathcal{T}}([\![\psi'\colon \sigma]\!]^\eta_{\mathcal{T}}).$$

  Because $F(\psi\,\psi') = F(\psi)F(\psi')$ the first set is equal to the semantics of

$$[\![F(\Phi)\colon T(\tau)]\!]^{V(\eta)}_{\mathcal{T}_d}.$$

This shows the correctness of the construction. □

To make sure that this construction does not exceed the bounds of HFL^k we have to verify that the growth of the defined construction is at most polynomially. Thereafter it is proven that the semantics of PHFL^k is reducible to the semantics of HFL^k.

Lemma 3.10. *Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS, η a variable mapping, Γ a type environment, Φ a well-typed d-adic PHFL^k formula of type τ, $\mathcal{T}_d$ the LTS transformed by the process of Definition 3.2, T the type function of Definition 3.4, V the variable mapping function of Definition 3.5 and F the formula function of Definition 3.6 then*

- *$\mathcal{T}_d$ grows polynomially related to $\mathcal{T}$,*
- *$T(\tau)$ grows polynomially related to τ,*
- *$T(\Gamma)$ grows polynomially related to Γ,*
- *$V(\eta)$ grows polynomially related to η and*
- *$F(\Phi)$ grows polynomially related to Φ.*

Proof. $F(\Phi)$ and $T(\tau)$ grow obviously linearly related to Φ and τ respectively. It follows that also $T(\Gamma)$ grows linearly related to Γ. To show that $\mathcal{T}_d$ grows polynomially related to $\mathcal{T}$, we take a look at the particular components of the LTS. For the set of states it holds that $|Q^d| = |Q|^d$. That means Q^d grows polynomially related to Q. The cardinality of the set of actions of

$\mathcal{T}_d$ is $|\Sigma_d| + |S_d|$. Since $|S_d|$ is a constant it holds also that $|\Sigma_d| = |\Sigma|^d$. By combining these two statements, one can see that the set of actions also grows polynomially. The set of propositions P^d also grows polynomially, because it is constructed like the subset of actions Σ_d. The labelled transition relation Δ_d also grows polynomially because it is constructed like the set of actions of $\mathcal{T}_d$. The first set has cardinality $|\Delta|^d$ and the second set is a constant. Again, by combining the previous statements it is clear that Δ_d grows polynomially related to Δ. At least, the mapping v_d grows polynomially related to v because the set of states and the set of propositions of $\mathcal{T}$ grow polynomially. As a result it follows that $\mathcal{T}_d$ grows polynomially related to $\mathcal{T}$. Because $\mathcal{T}_d$ grows polynomially related to $\mathcal{T}$ and $T(\tau)$ grows polynomially related to τ $V(\eta)$ also grows polynomially related to η. □

The following theorem is a consequence of Lemma 3.9, Lemma 3.10 and Theorem 3.1.

Theorem 3.11. *The model checking problem of PHFL^k for $k > 0$ is solvable in k-EXPTIME, i.e. on input $\mathcal{T}, s$ and Φ it is decidable in k-EXPTIME whether $\mathcal{T}, s$ is a model of Φ, where $\mathcal{T}$ is an LTS, s is a state and Φ in PHFL^k.*

3.2 Upper Bound of PHFL$^{k+1}_{tail}$

To show that the upper bound of the expressive power of PHFL$^{k+1}_{tail}$ is k-EXPSPACE/$\sim$ we can reduce the semantics of PHFL$^k_{tail}$ to the semantics of HFL$^k_{tail}$. Keep in mind that HFL$^k_{tail}$ is the set of tail-recursive 1-adic PHFL^k formulas. In combination with the following theorem we get the upper bound of the expressive power of PHFL$^k_{tail}$.

Theorem 3.12. *[13] Given an LTS $\mathcal{T}$, a state s and an HFL$^{k+1}_{tail}$ formula Φ, the model checking problem i.e., deciding whether $\mathcal{T}, s \models \Phi$ is in k-EXPSPACE, where $k > 0$.*

Lemma 3.13. *Let $\mathcal{T}$ be an LTS, η a variable mapping, Φ a well-typed d-adic PHFL$^k_{tail}$ formula, $\mathcal{T}_d$ the LTS transformed by process of Definition 3.2, V the variable mapping function of Definition 3.5 and F the formula function of Definition 3.6 then $[\![\Phi]\!]^\eta_{\mathcal{T}} = [\![F(\Phi)]\!]^{V(\eta)}_{\mathcal{T}_d}$.*

Proof. This proof is similar to the proof of Lemma 3.9. Because the definition of F obviously retains tail-recursiveness this lemma holds. □

The following theorem is a consequence of Lemma 3.13 and Theorem 3.12.

Theorem 3.14. *The model checking problem of* $PHFL^{k+1}_{tail}$ *for* $k > 0$ *is solvable in k-EXPSPACE, i.e. on input* $\mathcal{T}, s$ *and* Φ *it is decidable in k-EXPSPACE whether* $\mathcal{T}, s$ *is a model of* Φ*, where* $\mathcal{T}$ *is an LTS,* s *is a state and* Φ *in* $PHFL^{k+1}_{tail}$*.*

Chapter 4

Lower Bounds

In this chapter we want to establish that the lower bounds of the expressive power of PHFL^k and PHFL^k_{tail} are k-EXPTIME/$\sim$ and k-EXPSPACE/$\sim$, respectively. The lower bounds of PHFL^k and PHFL^k_{tail} can be proven by encoding the run of a Turing Machine as query. As another possibility one can use intermediate logics HO(LFP)^{k+1} and HO(PFP)^{k+1} and encode the bisimulation-invariant fragments of these as PHFL^k and PHFL^k_{tail}, respectively. Note that PHFL cannot distinguish between bisimilar structures [3]. This means that PHFL formulas can only define bisimulation-invariant graph problems. Moreover, PHFL is sufficient to encode the bisimulation-invariant fragments of HO(LFP)^{k+1} and HO(PFP)^{k+1}. To encode the bisimulation-invariant fragments of HO(LFP)^{k+1} and HO(PFP)^{k+1} as PHFL^k and PHFL^k_{tail} respectively we want to translate HO(LFP)^{k+1} formulas into PHFL^k formulas and HO(PFP)^{k+1} into a PHFL^k_{tail} formula.

Encoding existential quantifiers is the most complex part of the translation. In the first section we consider some preparations that are necessary to model them in PHFL. Thereafter, we show that existential quantifiers that bind a variable of order $k \geq 1$ can be expressed by a PHFL^{k-1} formula. In the subsequent section we use this formula to show that the bisimulation invariant fragment of HO(LFP)^{k+1} can be encoded into PHFL^k and so that the lower bound of the expressive power of PHFL^k is k-EXPTIME/$\sim$. And finally we show that the lower bound of the expressive power of PHFL^k_{tail} is k-EXPSPACE/$\sim$.

4.1 Preparation

Before we can start with the encodings there are some important steps that we have to consider. Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be an LTS, $q_1, \ldots, q_n, p_1, \ldots, p_m \in$

Q some states of $\mathcal{T}$ and let $\mathcal{T}'$ be the reduced LTS of $\mathcal{T}$ with respect to $p_1, \ldots, p_m$. As mentioned in the introduction of this chapter it is known from [2] that PHFL cannot distinguish between bisimilar structures. That means PHFL formulas can only define bisimulation-invariant graph problems. That means that, without loss of generality, we can check if $([q_1]_\sim, \ldots, [q_n]_\sim)$ satisfies a formula Φ with respect to $\mathcal{T}'$ instead of checking if $(q_1, \ldots, q_n)$ satisfies Φ with respect to $\mathcal{T}$. With $[q_i]_\sim$ we denote the equivalence class of q_i with respect to $\sim$.

From this point all LTS are reduced LTS with respect to some states of their state sets. In more detail, let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be a reduced LTS with respect to $(q_1, \ldots, q_r)$, where $q_1, \ldots, q_r \in Q$. On those LTS it is possible to define a total order on their states.

Remark 4.1. In [4] it was shown that it is possible to define a 2-adic PHFL^0 formula $\Phi_<$ over reduced LTS that defines a transitive relation $<$ such that $< \cap > = \emptyset$ and $< \cup > = \not\sim$. This relation $<$ defines a total order on states of a reduced LTS.

4.2 Existential Quantifiers in PHFL

In this section we define PHFL^k formulas that describe existential quantification over HO domains of types of order $k \geq 1$. But before we can define these formulas we have to translate the types.

Note that most types in HO^{k+1} do not exist in PHFL^k. For example, while HO^{k+1} includes sets of sets, PHFL^k does not support this kind of type. But each set X in HO^{k+1} can be described by the characteristic function of X in PHFL^k.

The following definition translates all HO types of order 2 or greater to types in PHFL. The base type of HO has to be encoded differently, and will be established after this definition.

Definition 4.2. T is a function that maps any type of HO of order 2 or greater to a type of PHFL defined inductive over the type of HO as follows:

$$T((\odot, \ldots, \odot)) = \bullet$$
$$T((\tau', \ldots, \tau')) = T(\tau')^+ \to (T(\tau')^+ \to \cdots \to (T(\tau')^+ \to \bullet) \cdots),$$

where $\tau' \neq \odot$.

Note that the orders of HO types and PHFL types are defined differently. It holds that $ord(T(\tau)) = order(\tau) - 2$ for all HO types τ with order 2 or greater.

Example 4.3. *Let $\tau = (\tau', \tau')$ be a type of HO with*

$$\tau' = ((\odot), (\odot))$$

then by Definition 4.2 of type function T

$$T(\tau) = T(\tau') \to (T(\tau') \to \bullet)$$

where

$$T(\tau') = \bullet \to (\bullet \to \bullet).$$

With this type function T an HO^{k+1} variable X of type τ can be translated to a PHFL^k variable of type $T(\tau)$. Intuitively, the variable X which is a set of $D_\tau(Q)$ in HO^{k+1} is represented in PHFL^k as the characteristic function of X over $D_\tau(Q)$. This function maps x to Q^d if $x \in X$ and x to $\varnothing$ if $x \notin X$, where $d > 0$ is a natural number. Note that the domain of HO types of order 2 is similar to the domain of base type of PHFL.

As mentioned above the base type of HO has to be encoded differently. The reason for that is that the base type in PHFL is a set of tuples of states and a single state cannot be depicted directly by a variable. This thesis uses the idea of [2] to use the polyadicity of PHFL to represent the different first-order variables of an HO formula Ψ. Each first-order variable of Ψ represents one component in the dimension of the corresponding PHFL^k formula Φ that means each variable increases the dimension of Φ. The assignment of a first-order variable X_i in Ψ is then the current state of the i-th component in Φ.

Let Φ be an HO formula. From now on, we assume that for every HO formula Φ, the first-order variables occurring in Φ are $X_1, \ldots, X_q$ for some q depending on Φ.

4.2.1 First-Order and Second-Order Quantification

After we know how to interpret the different HO types and variables we can now take a look at the existential quantification. Before we establish higher-order quantification we start with first-order and second-order quantification.

As mentioned in the introduction of this chapter we encode the bisimulation-invariant fragment of $\mathrm{HO(LFP)}^{k+1}$ and $\mathrm{HO(PFP)}^{k+1}$. In Section 4.1 we explained that we can treat the semantics of them by using only reduced LTS, where any state is reachable by at least one of the states $q_1, \ldots, q_r$. Because of the total order on states of $\mathcal{T}$ explained in Remark 4.1 the first-order quantification can be encoded by going over all states reachable from one $q_1, \ldots, q_r$ and check whether we reached a state tuple where the bound formula holds.

To access the states $q_1, \ldots, q_r$ in the PHFL formulas that we get by encoding HO formulas we use the polyadicity and store $q_1, \ldots, q_r$ in components of a dimension that will be never influenced by the PHFL formulas. The following remark explains that the PHFL formulas has a dimension that is large enough to satisfy all the requirements.

Remark 4.4. The PHFL formula Φ that we get through the encoding of a given HO formula Ψ has dimension d that is always large enough to translate all second-order variables of Ψ to an order 0 variable in Φ. In more detail s is the maximal arity of second-order variables in Ψ and $d > s$. To compare two elements of Q^s, where Q is the set of states of an LTS, the dimension d of Φ is at least twice as big as the maximum of s. Because the higher-order variables can be handled otherwise they do not influence the dimension d. To distinguish all different first-order variables in Ψ, the dimension d of Φ has to be extended by q, where q is the number of different first-order variables. That means $d > 2s + q$. Finally, to access the states $q_1, \ldots, q_r$ described in Section 4.1, we extend d additionally with r components. That means the dimension of Φ is $d = 2s + q + r$.

If we consider $\exists(X_i : \odot).\, \Phi$ then it can be understood as the check whether we reach a state tuple where Φ holds once the i-th component is replaced by one of the last r components. This is where $q_1, \ldots, q_r$ are stored.

Definition 4.5. Let d and r be the constants as described in Remark 4.4 where $r < d$, and $1 \leq i \leq d$ then $\exists_i \Phi$ is a PHFL^k formula of dimension d defined as

$$\exists_i \Phi \coloneqq \bigvee_{j=d-r+1}^{d} \{(1, \ldots, i-1, j, i+1, \ldots, d)\}\mu(X : \bullet).\, \Phi \vee \bigvee_{a \in \Sigma} \langle a \rangle_i X.$$

The formula $\forall_i \Phi$ is also a PHFL^k formula of dimension d and is defined as

$$\forall_i \Phi \coloneqq \neg \exists_i \neg \Phi.$$

Note that the constant i of Definition 4.5 can be between 1 and d. Because we use $\exists_i$ later in this thesis only for such i that are in the range of 1 and $d - r$ the formula works as expected.

Observation 4.6. Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be a reduced LTS with respect to $(q_1, \ldots, q_r)$, where $q_1, \ldots, q_r \in Q$. The formula given in Definition 4.5 defines first-order quantification in PHFL, because it replaces the i-th component by one of the last r components and moves through all reachable states. This is enough because any element of Q is reachable in reduced LTS $\mathcal{T}$ from at least one state of $q_1, \ldots, q_r$ and those are stored in the last r components of

any d tuple. The movement at the i-th component and checking if Φ holds is a consequence of $\mu(X : \bullet).\, \Phi \vee \bigvee_{a\in\Sigma}\langle a\rangle_i X$. To replace the i-th component by one of $q_1, \ldots, q_r$ we use this part of the formula $\bigvee_d^{j=d-r+1}\{(1, \ldots, i-1, j, i+1, \ldots, d)\}$.

Now we consider second-order quantification. Let $\tau = (\odot, \ldots, \odot)$ be an HO type and Q the set of states of an LTS $\mathcal{T}$. Because the transformation for first-order quantification cannot be adapted to the second-order quantification, we use a different encoding. To obtain second-order quantification in PHFL we have to iterate over all possible elements of a given domain $D_\tau(Q)$ and check if the given formula is satisfied. The first thing that we need to iterate over any element of a domain $D_\tau(Q)$ is an order on $D_\tau(Q)$. If we have the order of $D_\tau(Q)$ we can use this order to define a formula that returns the successor of a given element of $D_\tau(Q)$ in the scope of this order. Finally, this formula can be used to iterate over all elements and check if a given formula is satisfied. At first, we need the order of domains of type $(\odot, \ldots, \odot)$.

To get the order of type $\tau = (\odot, \ldots, \odot)$ we define two formulas. The first one describes which is the smaller one of two given sets of type τ. The other one describes the smaller of two tuples of sets of type τ. We say that a tuple x is smaller than y with respect to τ if there is an index i such that the element in x on i is smaller than the element in y on i in respect to $\odot$, and such that there is no position j on the left-hand side of i such that the element in x on j is larger than the element in y on j with respect to $\odot$. We say that a set X is smaller than a set Y with respect to τ if there is an element s_1 in X that is not in Y, and such that all elements s_2 that are smaller than s_1 with respect to $\odot$ are only in X if s_2 is also in Y. This is formalized in PHFL in the following definition.

Definition 4.7. Let d and s be the constants as described in Remark 4.4 where $2 * s < d$, then $<^{\odot}$, $<^{\odot\times\cdots\times\odot}$ and $<^{(\odot,\ldots,\odot)}$ (X, Y) are PHFL^k formulas of dimension d defined as:

$$
\begin{aligned}
<^{\odot} &:= \Phi_{<} \\
<^{\odot\times\cdots\times\odot} &:= \bigvee_{i=1}^{s}\{(i, i+s, 3, \ldots, d)\} <^{\odot} \wedge \\
&\quad \bigwedge_{j=1}^{i-1}\{(j+s, j, 3, \ldots, d)\}\neg <^{\odot}
\end{aligned}
$$

$$
\begin{aligned}
<^{(\odot,\dots,\odot)}(X,Y) \coloneqq\ & \exists_{i_1}.\ \dots \exists_{i_n}.\{(i_1,\dots,i_n,n+1,\dots,d)\}Y \wedge \\
& \neg\{(i_1,\dots,i_n,n+1,\dots,d)\}X \wedge \\
& \forall_{j_1}.\ \dots \forall_{j_n}.\{(j_1,\dots,j_n,n+1,\dots,s, \\
& i_1,\dots,i_n,s+n+1,\dots,d)\} <^{\odot\times\cdots\times\odot} \Rightarrow \\
& (\{(j_1,\dots,j_n,n+1,\dots,d)\}X \Rightarrow \\
& \neg\{(j_1,\dots,j_n,n+1,\dots,d)\}Y)
\end{aligned}
$$

We can see that $<^{\odot\times\cdots\times\odot}$ defines the lexicographic order of HO type $\odot\times\cdots\times\odot$ and $<^{(\odot,\dots,\odot)}(X,Y)$ the orders of HO types $(\odot,\dots,\odot)$.

After we have now orders of the HO types $(\odot,\dots,\odot)$ we can define formulas that return the successor of an input element with respect to the order of $(\odot,\dots,\odot)$. The idea of the following formula is based on binary incrementation. Let $\tau=(\odot,\dots,\odot)$ be an HO type and Q the set of states of an LTS $\mathcal{T}$. Remember that a set $X\in D_\tau(Q)$ can be represented by its characteristic function. This can be transformed to a binary string where each position of this string represents an element of $D_\odot(Q)^n$. Because each position in the binary string represents an element of $D_\odot(Q)^n$ and a position always has to represent the same element in $D_\odot(Q)^n$, the elements in $D_\odot(Q)^n$ have to be ordered. The order of the elements of $D_\odot(Q)^n$ is a consequence of the formula $<^{\odot\times\cdots\times\odot}$ of Definition 4.7. If the position i in the binary string is 1, the element with index i in $D_\odot(Q)^n$ is also in X. And if the position i in the binary string is 0, the element with index i in $D_\odot(Q)^n$ is not in X. This binary representation of X in regard to $D_\odot(Q)^n$ can be thought of as a function $f\colon D_\tau(Q)\to 0,\dots,|D_\tau(Q)|-1$ such that each element X of $D_\tau(Q)$ will be mapped to its binary string in regard to $D_\odot(Q)^n$. Similar to operator F of Example 2.13 or operator $F^{\mathcal{T}}_\Phi$ of Example 2.50 the following formula identifies $Y\in D_\tau(Q)$ as the successor of $X\in D_\tau(Q)$ if $f(Y)=f(X)+1 \mod |D_\tau(Q)|$. In detail that means that the i-th bit is 1 in $f(Y)$ if the i-th bit is either 1 in X and there is a bit on the left of i that is 0 in X or the i-th bit is 1 in $f(Y)$ if it is 0 in X and all bits to the left of i are 1 in X.

Definition 4.8. Let d and s be the constants as described in Remark 4.4 where $2*s<d$, then $next^{(\odot,\dots,\odot)}$ is a PHFL^k formula of dimension d defined as:

$$
\begin{aligned}
next^{(\odot,\dots,\odot)} \coloneqq\ & \lambda(X\colon\bullet).\,(\neg X \wedge \forall_{s+1}\dots\forall_{s+s} <^{\odot\times\cdots\times\odot} \Rightarrow \\
& \{(s+1,\dots,s+s,s+1,\dots,d)\}X) \vee \\
& (X \wedge \exists_{s+1}\dots\exists_{s+s} <^{\odot\times\cdots\times\odot} \wedge \\
& \{(s+1,\dots,s+s,s+1,\dots,d)\}\neg X)
\end{aligned}
$$

Remark 4.9. Iterated application of $next^{(\odot,\dots,\odot)}$ to $\bot$ cycles through all elements in the domain of $(\odot,\dots,\odot)$. The PHFL formula $\bot$ represents the empty set. If we put $\bot$ into $next^{(\odot,\dots,\odot)}$ we can check whether all elements of $D_{(\odot,\dots,\odot)}$ satisfy one of the subformulas of the disjunction. We will see that only the smallest element with respect to $<^{\odot\times\cdots\times\odot}$ satisfies the first subformula. The reason is that the smallest element is not in the empty set and there are no smaller elements than the smallest. So the first subformula is true and it is the only element that satisfies the disjunction. So the formula $next^{(\odot,\dots,\odot)^1}\bot$ returns the set that includes only the smallest element with respect to $\odot\times\cdots\times\odot$. If we take a look at the formula $next^{(\odot,\dots,\odot)^2}\bot$, we put the set that only contains the smallest element into $next^{(\odot,\dots,\odot)}$. If we dive deeper into this formula we will see that the formula returns a set that includes only the second smallest element. The smallest element does not satisfy the disjunction, but the second smallest satisfies the second subformula. As described in the introduction of Definition 4.8, formula $next^{(\odot,\dots,\odot)}$ can be thought of as binary incrementation. In this manner each possible set of type $(\odot,\dots,\odot)$ will be reached. Note that if we put the full set into $next^{(\odot,\dots,\odot)}$ it returns the empty set.

With this definition we are now able to define the second-order quantification in PHFL.

Definition 4.10. Let d be the constant as described in Remark 4.4 and Φ be a PHFL^k formula with order 0 variable X, then $\exists^{(\odot,\dots,\odot)}X.\,\Phi(X)$ is a PHFL^k formula of dimension d defined as:

$$\exists^{(\odot,\dots,\odot)}X.\,\Phi(X) \coloneqq (\mu(F\colon\bullet\to\bullet).\,\lambda(X\colon\bullet).\,\Phi(X)\vee F(next^{(\odot,\dots,\odot)}X))\bot$$

The formula $\forall^{(\odot,\dots,\odot)}X.\,\Phi$ is also a PHL^k formula of dimension d and is defined as

$$\forall^{(\odot,\dots,\odot)}X.\,\Phi(X) \coloneqq \neg\exists^{(\odot,\dots,\odot)}X.\,\neg\Phi(X).$$

In the last step we show that the given formula of Definition 4.10 defines second-order existential quantification in PHFL.

Lemma 4.11. *For all HO types $\tau=(\odot,\dots,\odot)$, all variable mappings η and all LTS $\mathcal{T}$ it holds that*

$$[\![\exists^{\tau}X.\,\Phi(X)]\!]^{\eta}_{\mathcal{T}} \equiv \bigsqcup_{\mathcal{X}\in[\![\tau]\!]_{\mathcal{T}}} [\![\Phi(X)]\!]^{\eta[X\to\mathcal{X}]}_{\mathcal{T}}.$$

Proof. By fixpoint unfolding and β-reduction the formula

$$\exists^{(\odot,\dots,\odot)}X.\,\Phi(X) = (\mu(F\colon\bullet\to\bullet).\,\lambda(X\colon\bullet).\,\Phi(X)\vee F(next^{(\odot,\dots,\odot)}X))\bot$$

is equivalent to

$$\Phi(\bot) \vee \Phi(next^{(\odot,\dots,\odot)}\bot) \vee \Phi(next^{(\odot,\dots,\odot)}(next^{(\odot,\dots,\odot)}\bot)) \vee \cdots .$$

This can be simplified to

$$\bigvee_{i\geq 0} \Phi(next^{(\odot,\dots,\odot)^i}\bot)$$

Because these sets, reached as explained in Remark 4.9, are checked one after another in the scope of Φ and this iteration is finite because of the least fixpoint operator, the lemma holds. □

4.2.2 Higher-Order Quantification

In this subsection we use the encoding of second-order quantification we just defined to lift this encoding to higher-order quantification. The idea to obtain higher-order quantification in PHFL is similar to the second-order quantification. We use the existential quantifier of type $\tau = (\odot, \dots, \odot)$ to define the order of domains of kind $D_{(\tau,\dots,\tau)}(Q)$. This order can then be used to define a formula that returns the successor of a given element of $D_{(\tau,\dots,\tau)}(Q)$ with respect to this order. Finally, we use this formula to define the existential quantifier of type $(\tau, \dots, \tau)$. This procedure will be applied to all possible types of HO. In this way we get higher-order quantification of any type in PHFL.

The first thing we need is the order for every domain. Note that an order for a type $\tau = (\tau', \dots, \tau')$ always depends on the quantifiers of type τ' and the orders of type $\tau' \times \cdots \times \tau'$. The orders of type $\tau' \times \cdots \times \tau'$, however, depends on the order of type τ'. In case of $\tau' = (\odot, \dots, \odot)$ we use the quantifier formulas of Definition 4.10. On the other hand if τ' is a type of an order bigger then 2, the quantifier formulas of Definition 4.16 are used. Like we did in the second-order case we define two formulas. The first one tells us which set of two given sets of type τ is the smaller one. The other formula tells us the same for two tuples in the same sets of type τ.

Definition 4.12. Let d be the constant as described in Remark 4.4 and $\tau \neq \odot$ an HO type, then $<^{(\tau,\dots,\tau)}(X, Y)$ and $<^{\tau\times\cdots\times\tau}(X, Y)$ are PHFL^k

formulas of dimension d defined as:

$$
\begin{aligned}
<^{\tau\times\cdots\times\tau}(X_1,Y_1,\ldots,X_n,Y_n) &\coloneqq \bigvee_{i=1}^{n} <^{\tau}(X_i,Y_i) \wedge \bigwedge_{j=1}^{i-1} \neg <^{\tau}(Y_j,X_j)\\
<^{(\tau,\ldots,\tau)}(X,Y) &\coloneqq \exists^{\tau} X_1.\ \ldots \exists^{\tau} X_n.\, (\cdots(Y\,X_1)\cdots)\,X_n\\
&\quad \wedge \neg(\cdots(X\,X_1)\cdots)\,X_n \wedge\\
&\quad \forall^{\tau} Y_1.\ \ldots \forall^{\tau} Y_n.\ <^{\tau\times\cdots\times\tau}(Y_1,X_1,\ldots,Y_n,X_n)\\
&\quad \Rightarrow \big((\cdots(X\,Y_1)\cdots)\,Y_n \Rightarrow (\cdots(Y\,Y_1)\cdots)\,Y_n\big).
\end{aligned}
$$

The formulas $\exists^{\tau}$ and $\forall^{\tau}$ are defined in Definition 4.10 or Definition 4.16 if $ord(\tau) = 2$ or $ord(\tau) > 2$, respectively. In case of $\tau = (\odot,\ldots,\odot)$ the formula $<^{(\odot,\ldots,\odot)}(X,Y)$ is defined in Definition 4.7.

We can see that $<^{\tau\times\cdots\times\tau}$ defines the lexicographic order of HO type τ and $<^{(\tau,\ldots,\tau)}(X,Y)$ the orders of HO types $(\tau,\ldots,\tau)$, where $\tau \neq \odot$. Note that $<^{(\tau,\ldots,\tau)}(X,Y)$ is well defined because the formula uses the quantifier formulas of a lower type and the lowest possible type is already defined in Definition 4.10. The same holds for $<^{\tau\times\cdots\times\tau}$, where the lowest used order formula is defined in Definition 4.7.

The following definition defines an abbreviation for the smallest function of an arbitrary PHFL type that have an PHFL order of at least 1.

Definition 4.13. Let $(\tau,\ldots,\tau)$ be an HO type with $\tau \neq \odot$ then $\bot_{T((\tau,\ldots,\tau))}$ is PHFL formula defined as

$$\bot_{T((\tau,\ldots,\tau))} \coloneqq (\lambda(X_1\colon T(\tau)).\ \ldots \lambda(X_n\colon T(\tau)).\,\bot)).$$

This formula represents a function that maps any input for $X_1,\ldots,X_n$ to $\bot$ and $\bot$ represents the empty set. That means this function is the smallest element with respect to $<^{(\tau,\ldots,\tau)}$.

With this definition of order of any HO type, always depending on the lower type existential quantifier, we can define formulas that return the successor of an input element with respect to the order of the HO type. The idea of the following formula is similar to the successor formula of Definition 4.8 and can be thought of as binary incrementation. Note that here the binary string represents functions. That means that one configuration of a function, represents one bit in the binary string. The smallest function, see Definition 4.13, is this where any element is mapped to $\varnothing$ and the greatest function where any element is mapped to Q^d, where Q is a state set of an LTS and d a dimension of PHFL.

Definition 4.14. Let d be the constant as described in Remark 4.4 and $\tau \neq \odot$ an HO type, then $next^{(\tau,\dots,\tau)}$ is a PHFL^k formula of dimension d defined as:

$$\begin{aligned} next^{(\tau,\dots,\tau)} \coloneqq \lambda(X\colon T((\tau,\dots,\tau))).\, \lambda(X_1\colon T(\tau)).\, \cdots \lambda(X_n\colon T(\tau)).\\ \big(\neg(\cdots(X\,X_1)\cdots)\,X_n \wedge \forall^\tau Y_1.\, \dots \forall^\tau Y_n.\\ <^{\tau\times\cdots\times\tau}(Y_1,X_1,\dots,Y_n,X_n) \Rightarrow (\cdots(X\,Y_1)\cdots)\,Y_n\big) \vee\\ \big((\cdots(X\,X_1)\cdots)\,X_n \wedge \exists^\tau Y_1.\, \dots \exists^\tau Y_n.\\ <^{\tau\times\cdots\times\tau}(Y_1,X_1,\dots,Y_n,X_n) \wedge \neg(\cdots(X\,Y_1)\cdots)\,Y_n\big) \end{aligned}$$

Remark 4.15. Iterated application of $next^{(\tau,\dots,\tau)}$ to $\bot_{T(\tau)}$ cycles through all elements in the domain of $(\tau,\dots,\tau)$. Note that this works similar to the iterated application of $next^{(\odot,\dots,\odot)}$ to $\bot$ in Remark 4.9. If we put $\bot_{T(\tau)}$ into $next^\tau$, where $\tau = (\tau',\dots,\tau')$ is an HO type of order $k+1$ and $\tau' \neq \odot$, we can check whether all elements of D_τ satisfy one of the subformulas of the disjunction. Note that the formulas for the quantifiers used in $next^\tau$ and $<^{\tau'\times\cdots\times\tau'}$ respectively are all for types of order k. By induction hypothesis those formulas define quantification of types of order k. Using these quantifiers we will see that only the smallest element with respect to $\tau' \times \cdots \times \tau'$ satisfies the first subformula. The reason is that the smallest element is mapped in the input function to the empty set and there are no smaller elements to the smallest. That means the first subformula is true and it is the only element that satisfies the disjunction. So the formula $next^\tau 1 \bot_{T((\tau,\dots,\tau))}$ returns the function that maps the smallest element with respect to $\tau' \times \cdots \times \tau'$ to Q^d and all other elements to $\emptyset$ where d is a dimension of PHFL formula $[\![\exists^\tau X.\, \Phi(X)]\!]^\eta_{\mathcal{T}}$. If we take a look at the formula $next^\tau 2 \bot_{T((\tau,\dots,\tau))}$, the function that maps only the smallest element to Q^d is set as input of $next^\tau$. If we dive deeper into this formula we will see that now the formula returns a function that maps only the second smallest element to Q^d. The smallest element does not satisfy the disjunction, but the second smallest satisfies the second subformula. As described in the introduction of Definition 4.8 formula $next^{(\odot,\dots,\odot)}$ is some kind of binary incrementation. The same holds for the formula $next^\tau$ of Definition 4.14. In this manner each possible function of type τ will be reached. Note that if we put the function that maps any input to the full set into $next^{(\odot,\dots,\odot)}$ it returns the function that maps any input to the empty set.

With the previous definition we are now able to define the higher-order quantification in PHFL.

Definition 4.16. Let d be the constant as described in Remark 4.4 and let $\tau = (\tau', \ldots, \tau')$ be an HO type of order l where $\tau' \neq \odot$. Furthermore, let Φ be a PHFL^k formula with free variable X of order $l-2$, then $\exists^\tau X.\, \Phi(X)$ is a PHFL^k formula of dimension d defined as

$$\exists^\tau X.\, \Phi(X) \coloneqq \big(\mu(F\colon T(\tau) \to \bullet).\, \lambda(X\colon T(\tau)).\, \Phi(X) \vee F(next^\tau X)\big) \bot_{T(\tau)}.$$

The formula $\forall^\tau X.\, \Phi$ is also a PHFL^k formula of dimension d and is defined as

$$\forall^\tau X.\, \Phi(X) \coloneqq \neg \exists^\tau X.\, \neg \Phi(X).$$

The last step is to show that the given formula of Definition 4.16 defines higher-order existential quantification.

Lemma 4.17. *For all HO types τ of order 3 or greater, all variable mappings η and all reduced LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ holds*

$$[\![\exists^\tau X.\, \Phi(X)]\!]^\eta_{\mathcal{T}} \equiv \bigsqcup_{\mathcal{X} \in [\![\tau]\!]_{\mathcal{T}}} [\![\Phi(X)]\!]^{\eta[X \to \mathcal{X}]}_{\mathcal{T}}.$$

Proof. This lemma is proven by induction over the order of type τ. The induction basis $\tau = (\odot, \ldots, \odot)$ is given by Lemma 4.11. Based on the induction hypothesis, for any HO type τ of order k, all variable mappings η and all reduced LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ it holds that

$$[\![\exists^\tau X.\, \Phi(X)]\!]^\eta_{\mathcal{T}} \equiv \bigsqcup_{\mathcal{X} \in [\![\tau]\!]_{\mathcal{T}}} [\![\Phi(X)]\!]^{\eta[X \to \mathcal{X}]}_{\mathcal{T}}.$$

We have to show that for any HO type $\tau' = (\tau, \ldots, \tau)$ of order $k+1$, all variable mappings η and all reduced LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ the following statement holds

$$[\![\exists^{\tau'} X.\, \Phi(X)]\!]^\eta_{\mathcal{T}} \equiv \bigsqcup_{\mathcal{X} \in [\![\tau']\!]_{\mathcal{T}}} [\![\Phi(X)]\!]^{\eta[X \to \mathcal{X}]}_{\mathcal{T}}.$$

By fixpoint unfolding and β-reduction the formula

$$\exists^{\tau'} X.\, \Phi(X) \coloneqq \big(\mu(F\colon T(\tau') \to \bullet).\, \lambda(X\colon T(\tau')).\, \Phi(X) \vee F(next^{\tau'} X)\big) \bot_{T(\tau')}$$

is equivalent to

$$\Phi(\bot_{T(\tau')}) \vee \Phi(next^{\tau'} \bot_{T(\tau')}) \vee \Phi(next^{\tau'} next^{\tau'} \bot_{T(\tau')}) \vee \cdots.$$

This can be simplified to

$$\bigvee_{i \geq 0} \Phi(next^{\tau i} \bot_{T(\tau')}).$$

Because these functions, reached as explained in Remark 4.15, are checked consecutively in the scope of Φ and this iteration is finite because of the least fixpoint operator, the lemma holds. □

4.3 Lower Bound of PHFL^k

As mentioned in the introduction of this chapter we can show that the lower bound of PHFL^k is k-EXPTIME/$\sim$ by making a detour over HO(LFP)$^{k+1}$. This and following ideas are oriented on [2] where it was shown that PHFL^1 captures 1-EXPTIME/$\sim$. At first we will see that it was proven that k-fold exponential time coincides with HO(LFP)$^{k+1}$ over finite and ordered structures. To use this we have to encode the bisimulation invariant fragment of HO^{k+1} into PHFL^k. Therefore, we define an abbreviation for the HO(LFP) formulas that uses the LFP operator first and then a function that use this abbreviation and those of Section 4.2 to map an HO(LFP)$^{k+1}$ formula to a PHFL^k formula. Finally, we show that for any bisimulation-invariant HO(LFP)$^{k+1}$ formula Φ there is a transformed PHFL^k formula Ψ such that the query $\mathcal{Q}^d_\Psi$ defined by Ψ can be obtained from the query $\mathcal{Q}^f_\Phi$ defined by Φ via projection to the relevant components. Hence, PHFL^k can define all queries defined by the bisimulation-invariant fragment of HO(LFP)$^{k+1}$.

Theorem 4.18. *[5] For all $k \geq 1$, HO(LFP)$^{k+1}$ captures k-EXPTIME over finite and ordered structures.*

The proof follows the idea to encode the run of a k-EXPTIME Turing Machine M by a formula Φ of HO(LFP)$^{k+1}$ in such a way that M accepts the standardized encoding of $\mathcal{A}$ and η iff $\mathcal{A}, \eta \models \Phi$. On the other hand, such a model-checking problem can be solved by a k-EXPTIME Turing Machine M_Φ.

Because Theorem 4.18 holds, it is also possible to prove that the lower bound of PHFL^k is in k-EXPTIME/$\sim$ by encoding the bisimulation invariant fragment of HO(LFP)$^{k+1}$ into PHFL^k. To encode the bisimulation invariant fragment of HO(LFP)$^{k+1}$ into PHFL^k we have to define a function that transforms an HO (LFP)$^{k+1}$ formula into a PHFL^k formula. Note that the types and variables of an HO formula also need to be transformed. See Section 4.2 for further details.

Before we consider the definition of the transforming function, we define a PHFL formula for HO formulas that uses the LFP operator.

Definition 4.19. Let d be the constant as described in Remark 4.4 and let X be an HO variable of HO type $\tau = (\tau', \ldots, \tau')$ where $\tau' \neq \odot$. Furthermore, let Φ be a PHFL^k formula, then $LFP^\tau X.\, \Phi$ is a PHFL^k formula with dimension d, defined as:

$$LFP^\tau X.\, \Phi \coloneqq \mu(X \colon T(\tau)).\, \Phi(X).$$

In case of $\tau = (\odot, \ldots, \odot)$ let

$$LFP^\tau X.\, \Phi \coloneqq \mu(X \colon \bullet).\, \Phi(X).$$

Now we are able to define the function in the following definition by using the abbreviations of Definitions 4.10, 4.16, 4.5 and 4.19. The function translates a bisimulation invariant HO(LFP)$^{k+1}$ formula to a PHFL^k formula.

Definition 4.20. Define F as the function that maps a bisimulation invariant HO(LFP)$^{k+1}$ formula φ to a PHFL^k formula with dimension d, where d and s are the constants as described in Remark 4.4 and $\Phi_\sim$ is the formula of Example 2.19. In detail, F is defined inductive over φ as follows:

$$
\begin{aligned}
F(p(X_i)) &\coloneqq p_{2s+i} \\
F(a(X_i, X_j)) &\coloneqq \langle a \rangle_{2s+i} \{(2s+i, 2s+j, \\
&\qquad 3, \dots, d)\} \Phi_\sim \\
F(\Phi \vee \Psi) &\coloneqq F(\Phi) \vee F(\Psi) \\
F(\neg \Phi) &\coloneqq \neg F(\Phi) \\
F(\exists (X_i \colon \odot).\, \Phi) &\coloneqq \exists_{2s+i} F(\Phi) \\
F(\exists (X \colon \tau).\, \Phi) &\coloneqq \exists^\tau X.\, F(\Phi(X)) \\
F([LFP\ \Phi(X, X_{i_1}, \dots, X_{i_n})](V_{j_1}, \dots, V_{j_n})) &\coloneqq \{(j_1, \dots, j_n, n+1, \dots, d)\} \\
&\qquad LFP^{(\odot, \dots, \odot)} X.\, F(\Phi) \\
F([LFP\ \Phi(X, X_1, \dots, X_n)](V_1, \dots, V_n)) &\coloneqq (\cdots (LFP^\tau X.\, F(\Phi))\, V_1) \cdots)\, V_n \\
F(X(X_{i_1}, \dots, X_{i_n})) &\coloneqq \{(2s+i_1, \dots, 2s+i_n, \\
&\qquad n+1, \dots, d)\} X \\
F(X(X_1, \dots, X_n)) &\coloneqq (\cdots (X\, X_1) \cdots)\, X_n
\end{aligned}
$$

Keep in mind that we are working on LTS that means that the relations in the signatures for HO(LFP) formulas have either arity one or two. Relations with arity one represent the propositions and those with arity two the actions of an LTS.

4.3.1 Variables

After we encoded HO(LFP)$^{k+1}$ syntactically into PHFL^k formulas, the last step is to translate the interpretation of variables. As described in Section 4.2 the variables in HO of types with order 3 or higher are not supported in PHFL. Also first-order variables are not supported. Therefore, we define a function that maps a given variable mapping for an HO formula to the correct variable mapping for PHFL semantics. This function ignores the mapping of first-order variables, maps second-order variables to sets and higher-order variables to the corresponding characteristic function. Note that the sets of order 2 in HO have order 0 in PHFL. The first-order variables of an

HO formula that are marked as undefined in the following function, can be mapped to any arbitrary value because we do not use them in PHFL directly.

Normally, the HO variables of order 2 or higher are syntactically equivalent to the variables used in the translated PHFL formulas. To distinguish them in this definition an HO variable X is denoted by $\hat{X}$ for the usage in PHFL context.

Definition 4.21. Let d be the constant as described in Remark 4.4, let η be a variable mapping for an HO^k formula and let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be a reduced LTS, then η_V is a variable mapping for a PHFL^k formula with dimension d. Then the variable mapping η_V is defined as:

$$\eta_V(\hat{X}) \coloneqq \begin{cases} \text{undefined}, & \text{if } X \text{ is of type } \odot \\ A, & \text{if } X \text{ is of type } (\odot, \ldots, \odot) \\ G, & \text{if } X \text{ is of type } (\tau, \ldots, \tau) \text{ and } \tau \neq \odot, \end{cases}$$

where $A \subseteq Q^d$ such that $(q_1, \ldots, q_n, q_{n+1}, \ldots, q_d) \in A$ iff $(q_1, \ldots, q_n) \in \eta(X)$ and G is a function of type $T((\tau, \ldots, \tau))$ defined as follows:

$$(\cdots \big(G\, \eta_V(\hat{X}_1)\big) \cdots)\, \eta_V(\hat{X}_n) = Q^d \qquad \text{iff } (\eta(X_1), \ldots, \eta(X_n)) \in \eta(X)$$
$$(\cdots \big(G\, \eta_V(\hat{X}_1)\big) \cdots)\, \eta_V(\hat{X}_n) = \emptyset \qquad \text{iff } (\eta(X_1), \ldots, \eta(X_n)) \notin \eta(X)$$

The following example shows how a set of higher type variables will be translated into the characteristic function via the variable mapping η_V of Definition 4.21.

Example 4.22. *Let $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ be a reduced LTS with the set of states $Q = \{1, 2, 3\}$, let X be an HO(LFP)$^{k+1}$ variable of type $((\odot, \odot), (\odot, \odot))$ mapped through variable mapping η to*

$$\eta(X) = \{(\{(1,1)\}, \{(2,2)\}), (\{(1,1), (2,2)\}, \{(3,3)\})\},$$

let d be a dimension of PHFL, then $\eta_V(X)$ is a PHFL^k function of type $\bullet \to (\bullet \to \bullet)$ such that $\eta_V(X)\ (\{(1,1)\}) = f$, $\eta_V(X)(\{(1,1),(2,2)\}) = g$ and $\eta_V(X)(z) = h$ for $z \in Q^d$ where $z \neq \{(1,1)\}$ and $z \neq \{(1,1),(2,2)\}$. Moreover, f, g and h are functions of type $\bullet \to \bullet$ where $f(\{(2,2)\}) = g(\{(3,3)\}) = Q^d$ and $f(z) = g(z') = h(z'') = \emptyset$ for $z, z', z'' \in Q^d$ where $z \neq \{(2,2)\}$ and $z' \neq \{(3,3)\}$.

4.3.2 Correctness Proof

The last step is to show that for any query that is defined by an HO(LFP)$^{k+1}$ formula Φ can be obtained from the query defined by the PHFL^k formula Ψ

that is obtained by transforming Φ via projection to the relevant components. As mentioned in Section 4.1, without loss of generality, the statement can be proven by considering only reduced LTS.

To make the correctness proof clearer there is one last remark.

Remark 4.23. It holds for any HO(LFP)$^{k+1}$ formula Φ that for PHFL^k formula $F(\Phi)$ the type judgment $\varnothing \vdash F(\Phi)\colon \bullet$ is derivable. This statement is easily proved by induction over the structure of formula Φ.

Because the type judgement is always derivable we ignore the type environment in the following proof and write just $[\![\Phi]\!]^\eta_\mathcal{T}$ instead of $[\![\Gamma \vdash F(\Phi)\colon \tau]\!]^\eta_\mathcal{T}$, where Φ is a PHFL formula, η is a variable mapping, $\mathcal{T}$ is an LTS, Γ is a type environment and τ is a PHFL type.

Theorem 4.24. *Let Φ be a bisimulation-invariant formula of HO(LFP)$^{k+1}$, Ψ a formula of PHFL^k and let $d, f \geq 1$ and $k > 0$. For every Φ there is a Ψ such that a projection on the d-adic query $\mathcal{Q}^d_\Phi$ that is defined by Φ is equal to the f-adic query $\mathcal{Q}^f_\Psi$ that is defined by Ψ.*

Proof. This lemma can be proven by showing for all HO(LFP)$^{k+1}$ formulas Φ with first-order variables $X_1, \ldots, X_q$, all reduced LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ with respect to $\mathbf{q}_r = q_1, \ldots, q_r$ and all variable mappings η that it holds that $\mathcal{T}, \eta \models \Phi$ iff $\mathbf{q} = (\mathbf{q}_s, \mathbf{q}_s, \mathbf{q}_q, \mathbf{q}_r)$ and $\mathbf{q} \in [\![F(\Phi)]\!]^{\eta_V}_\mathcal{T}$. Here $\mathbf{q}_s = q'_1, \ldots, q'_s$ is a sequence of s placeholders used for the interaction of second-order variables, $\mathbf{q}_q = \eta(X_1), \ldots, \eta(X_q)$ is a sequence of first-order variables that are mapped by η where $\eta(X_i) = q_0$ if X_i is bound to a quantifier, $q_0, q'_1, \ldots, q'_s \in Q$ are arbitrary states, F is the formula function of Definition 4.20 and η_V the variable mapping of Definition 4.21. This statement can be proven by induction over Φ.

- In case of $\Phi = p(X_i)$ where X_i is a free first-order variable then $\mathcal{T}, \eta \models \Phi$ holds exactly then if $\eta(X_i) \in p^\mathcal{T}$. Translated to the normal LTS definition of $\mathcal{T}$ it is the same as $p \in v(\eta(X_i))$. With $F(\Phi) = p_{2s+i}$ this is exactly

$$(\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \ldots, \eta(X_{i-1}), \eta(X_i), \eta(X_{i+1}), \ldots, \eta(X_q), \mathbf{q}_r) \in [\![F(\Phi)]\!]^{\eta_V}_\mathcal{T}.$$

- In case of $\Phi = a(X_i, X_j)$ where X_i and X_j are free first-order variables such that, without loss of generality, $i < j$. All other cases working similar. Then $\mathcal{T}, \eta \models \Phi$ holds exactly then if $(\eta(X_i), \eta(X_j)) \in a^\mathcal{T}$. Translated to the normal LTS definition of $\mathcal{T}$ it is the same as $\eta(X_i) \xrightarrow{a} \eta(X_j)$. By definition of the semantics of $\langle a \rangle_{2s+i}$ the tuple

$$\begin{aligned}(&\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \ldots, \eta(X_{i-1}), \eta(X_i), \eta(X_{i+1}), \ldots, \eta(X_{j-1}), \eta(X_j),\\ &\eta(X_{j+1}), \ldots, \eta(X_q), \mathbf{q}_r)\end{aligned}$$

is an element of the semantics of $\langle a \rangle_{2s+i} \Phi_\sim$ iff

$$(\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \ldots, \eta(X_{i-1}), q_m, \eta(X_{i+1}), \ldots, \eta(X_{j-1}), q_n, \eta(X_{j+1}), \ldots, \eta(X_q), \mathbf{q}_r)$$

is an element of the semantics of $\Phi_\sim$ where from $\eta(X_i)$ there is an a-action to q_m. Because $\eta(X_j)$ is the state that has to be reachable via an a action from $\eta(X_i)$ we have to check if $q_n = \eta(X_j)$. If two states are equal in a reduced LTS it is the same to check if these two states are bisimilar. $\sim$ is given by formula $\Phi_\sim$ of Example 2.19. Because this formula returns those d-tuples where the first and second component are bisimilar, we have to move the $2s+i$-th and $2s+j$-th component to the first and second component. This is given by $\{(2s+i, 2s+j, 3, \ldots, d)\}\Phi_\sim$. Summarizing all these steps with $F(\Phi) = \langle a \rangle_{2s+i}\{(2s+i, 2s+j, 3, \ldots, d)\}\Phi_\sim$ it follows

$$(\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \ldots, \eta(X_{i-1}), \eta(X_i), \eta(X_{i+1}), \ldots, \eta(X_{j-1}), \eta(X_j), \eta(X_{j+1}), \ldots, \eta(X_q), \mathbf{q}_r) \in [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}}$$

if $(\eta(X_i), \eta(X_j)) \in a^{\mathcal{T}}$ and

$$(\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \ldots, \eta(X_{i-1}), \eta(X_i), \eta(X_{i+1}), \ldots, \eta(X_{j-1}), \eta(X_j), \eta(X_{j+1}), \ldots, \eta(X_q), \mathbf{q}_r) \notin [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}}$$

if $(\eta(X_i), \eta(X_j)) \notin a^{\mathcal{T}}$.

- In case of $\Phi = X(X_{i_1}, \ldots, X_{i_n})$ where X is a free variable of HO type $(\odot, \ldots, \odot)$ and $X_{i_1}, \ldots, X_{i_n}$ are free first-order variables such that, without loss of generality, $i_1 < i_2, \ldots, i_{n-1} < i_n$. All other cases working similar. Then $\mathcal{T}, \eta \models \Phi$ holds exactly then if $(\eta(X_{i_1}), \ldots \eta(X_{i_n})) \in \eta(X)$. Because of definition of η_V the tuple $(\eta(X_{i_1}), \ldots \eta(X_{i_n}), q'_{n+1}, \ldots, q'_s, \mathbf{q}_s, \mathbf{q}_q, \mathbf{q}_r)$ is in $\eta_V(X)$ if $(\eta(X_{i_1}), \ldots \eta(X_{i_n})) \in \eta(X)$ and is not in $\eta_V(X)$ otherwise. Because components $1, \ldots, n$ are not set to the mappings of first-order variables $X_{i_1}, \ldots, X_{i_n}$, we first move the components $2s+i_1, \ldots, 2s+i_n$ to components $1, \ldots, n$ respectively and check then if
$$(\eta(X_{i_1}), \ldots \eta(X_{i_n}), q'_{n+1}, \ldots, q'_s, \mathbf{q}_s, \mathbf{q}_q, \mathbf{q}_r) \in \eta_V(X).$$
So it holds with $F(\Phi) = (2s+i_1, \ldots, 2s+i_n, n+1, \ldots, d)X$ that
$$(\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \ldots, \eta(X_{i_1-1}), \eta(X_{i_1}), \eta(X_{i_1+1}), \ldots, \eta(X_{i_n-1}), \eta(X_{i_n}), \eta(X_{i_n+1}), \ldots, \eta(X_q), \mathbf{q}_r) \in [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}}$$

if $(\eta(X_{i_1}), \dots \eta(X_{i_n})) \in \eta(X)$ and

$$(\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \dots, \eta(X_{i_1-1}), \eta(X_{i_1}), \eta(X_{i_1+1}), \dots, \eta(X_{i_n-1}), \eta(X_{i_n}), \eta(X_{i_n+1}), \dots, \eta(X_q), \mathbf{q}_r) \notin [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}}$$

if $(\eta(X_{i_1}), \dots \eta(X_{i_n})) \notin \eta(X)$.

- In case of $\Phi = X(X_1, \dots, X_n)$ where X is a free variable of HO type $(\tau, \dots, \tau)$ and $X_1, \dots, X_n$ are free variables of HO type τ then $\mathcal{T}, \eta \models \Phi$ holds exactly then if $(\eta(X_1), \dots \eta(X_n)) \in \eta(X)$. Because of definition of η_V it follows

$$(\cdots (\eta_V(X)\, \eta_V(X_1)) \cdots)\, \eta_V(X_n) = Q^d$$

if $(\eta(X_1), \dots \eta(X_n)) \in \eta(X)$ and

$$(\cdots (\eta_V(X)\, \eta_V(X_1)) \cdots)\, \eta_V(X_n) = \emptyset$$

if $(\eta(X_1), \dots \eta(X_n)) \notin \eta(X)$. With $F(\Phi) = (\cdots (X\, X_1) \cdots)\, X_n$ it follows

$$\mathbf{q} \in [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}} = Q^d.$$

if $(\eta(X_1), \dots \eta(X_n)) \in \eta(X)$ and

$$\mathbf{q} \notin [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}} = \emptyset.$$

if $(\eta(X_1), \dots \eta(X_n)) \notin \eta(X)$.

By induction hypothesis it holds for HO(LFP)$^{k+1}$ formulas Ψ and Ψ' with first-order variables $X_1, \dots, X_q$, all reduced LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ with respect to $\mathbf{q}_r$ and all variable mappings η that $\mathcal{T}, \eta \models \Psi$ iff $\mathbf{q} \in [\![F(\Psi)]\!]^{\eta_V}_{\mathcal{T}}$ and $\mathcal{T}, \eta \models \Psi'$ iff $\mathbf{q} \in [\![F(\Psi')]\!]^{\eta_V}_{\mathcal{T}}$, where $\mathbf{q} = (\mathbf{q}_s, \mathbf{q}_s, \mathbf{q}_q, \mathbf{q}_r)$.

- In case of $\Phi = \neg\Psi$ it follows that $\mathcal{T}, \eta \models \Phi$ exactly then if $\mathcal{T}, \eta \not\models \Psi$. By induction hypothesis that is exactly then the case when

$$\mathbf{q} \notin [\![F(\Psi)]\!]^{\eta_V}_{\mathcal{T}}.$$

This is exactly the case if

$$\mathbf{q} \in Q^d \setminus [\![F(\Psi)]\!]^{\eta_V}_{\mathcal{T}}.$$

And this is exactly the semantics of $F(\Phi) = \neg F(\Psi)$.

- In case of $\Phi = \Psi \vee \Psi'$ it follows that $\mathcal{T}, \eta \models \Phi$ exactly then if $\mathcal{T}, \eta \models \Psi$ or $\mathcal{T}, \eta \models \Psi'$. By induction hypothesis that is exactly then the case when

$$\mathbf{q} \in [\![F(\Psi)]\!]^{\eta_V}_{\mathcal{T}}$$

or

$$\mathbf{q} \in [\![F(\Psi')]\!]_{\mathcal{T}}^{\eta_V}.$$

Because $\sqcup_\bullet = \cup$ this can be combined to

$$\mathbf{q} \in [\![F(\Psi)]\!]_{\mathcal{T}}^{\eta_V} \sqcup_\bullet [\![F(\Psi')]\!]_{\mathcal{T}}^{\eta_V},$$

which is as desired.

- In case of $\Phi = \exists(X_i \colon \odot).\, \Psi$ it follows that $\mathcal{T}, \eta \models \Phi$ iff there exists $\mathcal{X} \in Q$ with $\mathcal{T}, \eta' \models \Psi$, where η' is a variable mapping with $\eta'(x) = \eta(x)$ for all variables $x \neq X_i$ and $\eta'(X_i) = \mathcal{X}$. By induction hypothesis it holds that $\mathcal{T}, \eta' \models \Psi$ is exactly the case when

  $$(\mathbf{q}_s, \mathbf{q}_s, \eta'(X_1), \ldots, \eta'(X_{i-1}), \eta'(X_i), \eta'(X_{i+1}), \ldots, \eta(X_q), \mathbf{q}_r) \in [\![F(\Psi)]\!]_{\mathcal{T}}^{\eta'_V}.$$

  To reach the value of $\eta'(X_i)$ we have to replace the $2s+i$-th component by one of the last r components and move through all reachable states. By Observation 4.6 the formula defined in Definition 4.5 fulfils this behaviour. Because the first-order variable X_i is represented by the $2s+i$-th component and $F(\Phi) = \exists_{2s+i}\Psi$, we replace in $F(\Phi)$ the $2s+i$-th component by one of the last r components, move through all reachable states and checking if $F(\Psi)$ holds. That means it holds that

  $$\mathbf{q} \in [\![F(\Phi)]\!]_{\mathcal{T}}^{\eta_V}$$

  iff $\mathcal{T}, \eta \models \Phi$.
- In case of $\Phi = \exists(X \colon \tau).\, \Psi$ it follows that $\mathcal{T}, \eta \models \Phi$ iff there exists $\mathcal{X} \in D_\tau(Q)$ with $\mathcal{T}, \eta' \models \Psi$, where η' is a variable mapping with $\eta'(x) = \eta(x)$ for all variables $x \neq X$ and $\eta'(X) = \mathcal{X}$. By induction hypothesis it follows that $\mathbf{q} \in [\![F(\Psi)]\!]_{\mathcal{T}}^{\eta'_V}$ iff $\mathcal{T}, \eta' \models \Psi$. By Lemma 4.17 the formula $\exists^\tau X.\, \Psi(X)$ is semantically equivalent to $\bigsqcup_{\mathcal{X} \in [\![\tau]\!]_{\mathcal{T}}} [\![\Psi(X)]\!]_{\mathcal{T}}^{\eta'}$. It follows with $F(\Phi) = \exists^\tau X.\, F(\Psi)(X)$ that it holds

  $$\mathbf{q} \in [\![F(\Phi)]\!]_{\mathcal{T}}^{\eta_V}$$

  iff $\mathcal{T}, \eta \models \Phi$.
- In case of $\Phi = [LFP\, \Psi(X, X_{i_1}, \ldots, X_{i_n})](V_{j_1}, \ldots, V_{j_n})$, where X is a free variable in Ψ of HO type $(\odot, \ldots, \odot)$ and $X_{i_1}, \ldots, X_{i_n}$ are free first-order variables of Ψ and $V_{j_1}, \ldots, V_{j_n}$ are first-order variables of Φ such that, without loss of generality, $i_1 < i_2, j_1 < j_2, \ldots, i_{n-1} < i_n, j_{n-1} < j_n$. All other cases working similar. Then it follows that $\mathcal{T}, \eta \models \Phi$ exactly if $(\eta(V_{j_1}), \ldots, \eta(V_{j_n})) \in LFP(F_\Psi^{\mathcal{T}})$. By definition of LFP the tuple

$(\eta(V_{j_1}),\ldots,\eta(V_{j_n}))$ is in $LFP(F^{\mathcal{T}}_{\Psi})$ iff X is the smallest X such that $X = F^{\mathcal{T}}_{\Psi}(X)$ and $(\eta(V_{j_1}),\ldots,\eta(V_{j_n})) \in F^{\mathcal{T}}_{\Psi}(X)$. By definition of $F^{\mathcal{T}}_{\Psi}(X)$ it holds that $(\eta(V_{j_1}),\ldots,\eta(V_{j_n})) \in F^{\mathcal{T}}_{\Psi}(X)$ exactly then if $\mathcal{T},\eta' \models \Psi$, where η' is a variable mapping with $\eta'(x) = \eta(x)$ for all variables $x \neq X$ and $\eta'(X) = F^{\mathcal{T}}_{\Psi}(X)$. By induction hypothesis this is exactly the case if

$$(\mathbf{q}_s,\mathbf{q}_s,\eta'(X_1),\ldots,\eta'(X_{i_1-1}),\eta'(X_{i_1}),\eta'(X_{i_1+1}),\ldots,\eta'(X_{i_n-1}),\\ \eta'(X_{i_n}),\eta'(X_{i_n+1}),\ldots,\eta'(X_q),\mathbf{q}_r) \in [\![F(\Psi)]\!]^{\eta'_V}_{\mathcal{T}}.$$

The next step is to show that it holds that $\eta'(X) = LFP(F^{\mathcal{T}}_{\Psi})$ exactly then if $[\![F(\Psi)]\!]^{\eta'_V}_{\mathcal{T}} = [\![\mu(X\colon\bullet).F(\Psi)]\!]^{\eta_V}_{\mathcal{T}}$. By Theorem 2.11 the least fixpoint of $F^{\mathcal{T}}_{\Psi}$ can be calculated by a sequence $X_0,\ldots,X_m$ where here $X_0 = \emptyset$ and $X_{i+1} = F^{\mathcal{T}}_{\Psi}(X_i)$ and $\eta'(X) = X_m$. On the other hand $[\![\mu(X\colon\bullet).F(\Psi)]\!]^{\eta_V}_{\mathcal{T}}$ can be calculated by a sequence $Y_0,\ldots,Y_{m'}$ where here $Y_0 = \varnothing$ and $Y_{i+1} = [\![F(\Psi)]\!]^{\eta_V[X\mapsto Y_i]}_{\mathcal{T}}$ and $Y_{m'} = [\![F(\Psi)]\!]^{\eta'_V}_{\mathcal{T}}$. Furthermore, let $\eta'^0,\ldots,\eta'^m$ be a sequence of variable mappings where $\eta'^i(x) = \eta(x)$ for all $x \neq X$ and $\eta'^i(X) = X_i$. It is easy to verify that $\eta'^m = \eta'$. For any η'^i we get by Definition 4.21 the variable mapping η'^i_V and so a sequence of variable mappings $\eta'^0_V,\ldots,\eta'^m_V$. Moreover, η^{j+1}_V is $[\![F(\Psi)]\!]^{\eta'^j_V}_{\mathcal{T}}$. Because $\eta'^0_V(X) = \varnothing$, $\eta'^1_V(X) = [\![F(\Psi)]\!]^{\eta'^0_V}_{\mathcal{T}}$ and so on, it follows that $\eta'^m_V(X) = [\![F(\Psi)]\!]^{\eta'_V}_{\mathcal{T}}$ if and only if $\eta'(X)$ is the least fixpoint of $F^{\mathcal{T}}_{\Psi}$.
Because of the construction of variable mapping η'_V and $(\eta'(V_{j_1}),\ldots,\eta'(V_{j_n})) \in \eta'(X)$ the tuple $(\eta'(V_{j_1}),\ldots,\eta'(V_{j_n}),q'_{n+1},\ldots,q'_s,\mathbf{q}_s,\mathbf{q}_q,\mathbf{q}_r)$ is also in $\eta'_V(X)$.
Because components $1,\ldots,n$ are not set to the mappings of first-order variables $V_{j_1},\ldots,V_{j_n}$, we first move the components $2s+j_1,\ldots,2s+j_n$ to components $1,\ldots,n$ respectively and check then the least fixpoint operator. So it holds with $F(\Phi) = \{2s+j_1,\ldots,2s+j_n,n+1,\ldots,d\}\mu(X).F(\Psi)$ that

$$(\mathbf{q}_s,\mathbf{q}_s,\eta(X_1),\ldots,\eta(V_{j_1-1}),\eta(V_{j_1}),\eta(V_{j_1+1}),\ldots\eta(V_{j_n-1}),\\ \eta(V_{j_n}),\eta(V_{j_n+1}),\ldots,\eta(X_q),\mathbf{q}_r) \in [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}}$$

exactly then if $\mathcal{T},\eta \models \Phi$.

- In case of $\Phi = [LFP\,\Psi(X,X_1,\ldots,X_n)](V_1,\ldots,V_n)$, where X is a free variable in Ψ of HO type $(\tau,\ldots,\tau)$ and $X_1,\ldots,X_n$ are free variables of Ψ of type τ and $V_1,\ldots,V_n$ are free variables of Φ also of type τ, then it follows that $\mathcal{T},\eta \models \Phi$ exactly then if $(\eta(V_1),\ldots,\eta(V_n) \in LFP(F^{\mathcal{T}}_{\Psi})$. By

definition of LFP the tuple $(\eta(V_1), \ldots, \eta(V_n))$ is in $LFP(F_\Psi^\mathcal{T})$ iff X is the smallest X such that $X = F_\Psi^\mathcal{T}(X)$ and $(\eta(V_1), \ldots, \eta(V_n)) \in F_\Psi^\mathcal{T}(X)$. By definition of $F_\Psi^\mathcal{T}(X)$ it holds that $(\eta(V_1), \ldots, \eta(V_n)) \in F_\Psi^\mathcal{T}(X)$ exactly then if $\mathcal{T}, \eta' \models \Psi$, where η' is a variable mapping with $\eta'(x) = \eta(x)$ for all variables $x \neq X$ and $\eta'(X) = F_\Psi^\mathcal{T}(X)$. By induction hypothesis this is exactly the case if

$$\mathbf{q} \in [\![F(\Psi)]\!]_\mathcal{T}^{\eta_V}.$$

The next step is to show that it holds that $\eta'(X) = LFP(F_\Psi^\mathcal{T})$ exactly then if $[\![F(\Psi)]\!]_\mathcal{T}^{\eta'_V} = [\![\mu(X\colon T((\tau, \ldots, \tau))).\, F(\Psi)]\!]_\mathcal{T}^{\eta_V}$. By Theorem 2.11 the least fixpoint of $F_\Psi^\mathcal{T}$ can be calculated by a sequence $X_0, \ldots, X_m$ where here $X_0 = \emptyset$ and $X_{i+1} = F_\Psi^\mathcal{T}(X_i)$ and $\eta'(X) = X_m$. On the other hand $[\![\mu(X\colon T((\tau, \ldots, \tau)).\ F(\Psi)]\!]_\mathcal{T}^{\eta_V}$ can be calculated by a sequence $Y_0, \ldots, Y_{m'}$ where here $Y_0 = \bot_{T((\tau,\ldots,\tau))}$ and $Y_{i+1} = [\![F(\Psi)]\!]_\mathcal{T}^{\eta_V[X \mapsto Y_i]}$ and $Y_{m'} = [\![F(\Psi)]\!]_\mathcal{T}^{\eta'_V}$. Furthermore, let $\eta'^0, \ldots, \eta'^m$ be a sequence of variable mappings where $\eta'^i(x) = \eta(x)$ for all $x \neq X$ and $\eta'^i(X) = X_i$. It is easy to verify that $\eta'^m = \eta'$. For any η'^i we get by Definition 4.21 the variable mapping η'^i_V and so a sequence of variable mappings $\eta'^0_V, \ldots, \eta'^m_V$. Moreover, η_V^{j+1} is $[\![F(\Psi)]\!]_\mathcal{T}^{\eta'^j_V}$. Because $\eta'^0_V(X) = \bot_{T((\tau,\ldots,\tau))}$, $\eta'^1_V(X) = [\![F(\Psi)]\!]_\mathcal{T}^{\eta'^0_V}$ and so on, it follows that $\eta'^m_V(X) = [\![F(\Psi)]\!]_\mathcal{T}^{\eta'_V}$ if and only if $\eta'(X)$ is the least fixpoint of $F_\Psi^\mathcal{T}$.
Because of the construction of variable mapping η'_V and $(\eta'(V_1), \ldots, \eta'(V_n)) \in \eta'(X)$ it holds that $(\cdots (\eta'_V(X)\, \eta'_V(V_1)) \cdots)\, \eta'_V(V_n) = Q^d$ and so

$$\mathbf{q} \in (\cdots (\eta'_V(X)\, \eta'_V(V_1)) \cdots)\, \eta'_V(V_n).$$

With $F(\Phi) = (\cdots (\mu(X\colon T((\tau, \ldots, \tau))).\, \Phi(X))\, V_1) \cdots)\, V_n$ it follows

$$\mathbf{q} \in [\![F(\Phi)]\!]_\mathcal{T}^{\eta_V}.$$

exactly then if $\mathcal{T}, \eta \models \Phi$.

□

Remark 4.25. In the proof of Theorem 4.24 the dimension d of $F(\Phi)$ is $2s+q+r$ but only the components where the free variables are represented are filled with input parameters. That means by only projecting these components in the by $F(\Phi)$ defined query $\mathcal{Q}^d_{F(\Phi)}$ we get the resulting query $\mathcal{Q}^f_\Phi$ that is defined by Φ.

The combination of Theorem 4.24, Theorem 4.18 and Theorem 3.11 proves the following theorem for $k > 0$. For $k = 0$ and $k = 1$ this statement was proven by M. Otto in [4] and by M. Lange and E. Lozes in [2].

Theorem 4.26. *Let $k \geq 0$. PHFL^k captures k-EXPTIME/$\sim$ over finite labelled transition systems.*

4.4 Lower Bound of PHFL$^{k+1}_{tail}$

The lower bound of PHFL$^{k+1}_{tail}$ can be proven similar to the lower bound of PHFL^k. The main idea is not to show directly that the lower bound of PHFL$^{k+1}_{tail}$ is k-EXPSPACE/$\sim$ but rather by detour over the bisimulation-invariant fragment of HO(PFP)$^{k+1}$. In the first subsection we show that a Turing Machine that is in k-EXPSPACE can be encoded by an HO(PFP)$^{k+1}$ formula. The next subsection uses this statement to show that the lower bound of PHFL$^{k+1}_{tail}$ is k-EXPSPACE/$\sim$ by encoding formulas of HO(PFP)$^{k+1}$ in PHFL$^{k+1}_{tail}$.

4.4.1 k-EXPSPACE and HO(PFP)$^{k+1}$

In this subsection we want to show that a run of an $exp(k, f(n))$ space bounded DTM can be encoded by some HO^{k+1} formula. The main idea of this statement is an extension of the result of Abiteboul and Vianu [17] into higher-order. They have shown that HO(PFP)^1 coincides with 0-EXPSPACE.

Lemma 4.27. *Given a k-EXPSPACE-bounded DTM $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F, R)$ there exists a formula Ψ in HO(PFP)$^{k+1}$ over signature σ such that for all suitable variable mappings η and all LTS $\mathcal{T}$ it holds that $\mathcal{T}, \eta \models \Psi$ exactly then if the run of M on the standard coding of $(\mathcal{T}, \eta)$ is accepting.*

Proof. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, F, R)$ be a $exp(k, f(n))$ space bounded DTM, $\mathcal{T}$ an LTS and Q' the set of states of $\mathcal{T}$. Furthermore, we can advise an linear ordering $\exists(<\colon (\odot, \odot)).\, \varphi$ on Q' of $\mathcal{T}$, where φ describes that $<$ is an order [1]. Finally, let τ be an HO type of order $k+1$ and η a variable mapping. To prove this lemma we want to define a relational representation, of the final configuration of M that has the standardized encoding of $(\mathcal{T}, \eta)$[1], abbreviated with w, as input word, as a partial fixpoint of some HO(PFP)$^{k+1}$ formula. In order to do this, we construct a partial fixpoint of order $k+2$ such that for all i, the i-th approximation of this fixpoint encodes the i-th configuration in the run of M with input word w.

[1]The standardized encoding of structures is a non-trivial problem. Because the description of the encoding goes beyond the scope of this thesis, we only refer to [17] for further information about the standardized encoding of structures.

Before we can define the configurations of M in HO we have to make some preparations. Remember that a configuration of M comprises the current state, the current reading head position and the current tape content represented by a function. These configurations will be combined in one relation X. Because the size of the formula we build have to be polynomial and the reading head of M can be on one of $exp(k, f(n))$ cells for example, we have to encode the number in sets of order $k+1$. Furthermore, in order to not exceed the bound of order $k+1$, we have to split the tape content function in such a way that in one tuple x of X is just the current state, the current head position and one position of the tape with its content. By syntax of HO types each component of $x \in X$ has to be of the same type, so to access the different states and tape symbols they have to be numerated. $\{0, \ldots, |Q|-1\}$ for states and $\{0, \ldots, |\Gamma|-1\}$ for tape symbols.

The next step is to define some abbreviations that we want to use in the definitions of the configurations. The first and most important abbreviation is the definition of orders of any HO type. These orders are defined similarly to the defined formulas of Definition 4.12. A tuple x is smaller then a tuple y if there is a position i where $X_i < y_i$ and there is no position $j < i$ where $X_j > y_j$. A set X is smaller then a tuple Y if there is a $x \in Y$ such that $x \notin X$ and there is no $y < x$ such that $y \in X$ but $y \notin Y$.

$$
\begin{aligned}
<^{\odot}(x, y) &\coloneqq <(x, y) \\
<^{\tau' \times \cdots \times \tau'}(X_1, y_1, \ldots, X_n, y_n) &\coloneqq \bigvee_{i=1}^{n} <^{\tau'}(X_i, y_i) \wedge \bigwedge_{j=1}^{i-1} \neg <^{\tau'}(y_j, X_j) \\
<^{(\tau', \ldots, \tau')}(X, Y) &\coloneqq \exists (X_1 \colon \tau').\ \ldots \exists (X_n \colon \tau').\, Y(X_1, \ldots, X_n) \\
&\quad \wedge \neg X(X_1, \ldots, X_n) \wedge \forall (y_1 \colon \tau').\ \ldots \forall (y_1 \colon \tau'). \\
&\quad <^{\tau' \times \cdots \times \tau'}(y_1, X_1, \ldots, y_n, X_n) \\
&\quad \Rightarrow (X(y_1, \ldots, y_n) \Rightarrow Y(y_1, \ldots, y_n))
\end{aligned}
$$

With these formulas it is possible to define another two important abbreviations. On the one hand equality of two variables of arbitrary type and on the other hand the successor of a given element. If X and Y are two variables of type τ then the equality of X and Y is given by the formula

$$X = Y \coloneqq \neg <^{\tau}(X, Y) \wedge \neg <^{\tau}(Y, X).$$

Finally, if X and Y are two variables of type τ then the prove that Y is the successor of X is given by the formula

$$next^{\tau}(X, Y) \coloneqq <^{\tau}(X, Y) \wedge \forall (Z \colon \tau).\ <^{\tau}(Z, Y) \Rightarrow <^{\tau}(Z, X).$$

Now we are able to define the configurations in HO(PFP)^k. The first configuration is the initial configuration. For input word w this is given by the formula

$$\begin{aligned}\varphi_0(q,h,i,b) \coloneqq q = q_0 \land h = 0 \land (\neg <^\tau (|w|, i) \Rightarrow b = w_i) \lor \\ (\neg <^\tau (i, |w|) \land \neg i = |w| \Rightarrow b = \square))\end{aligned}$$

where q is the current state, h the current head position, i a tape index and b the symbol on i. q_0, 0, $|w|$, w_i and $\square$ are the numerical representations as sets of the same elements in Q and Γ.

To iterate through all configurations of M on input w, we need a variable X of type $\tau = (\tau', \tau', \tau', \tau')$ where τ' has order $k+1$, so X has order $k+2$. On an iteration $(F_\varphi^{\mathcal{T},\eta})^{i+1}(\varnothing)$ for the following formula φ the variable X includes all configurations that will be reached within i transitions.

$$\begin{aligned}\varphi(X,q,h,j,b) \coloneqq (\forall(q_{old}\colon \tau').\, \forall(h_{old}\colon \tau').\, \forall(j_{old}\colon \tau').\, \forall(b_{old}\colon \tau').\, \\ \neg X(q_{old}, h_{old}, i_{old}, b_{old}) \lor \neg\varphi_0(q,h,i,b)) \lor \xi(X,q,h,i,b)\end{aligned}$$

Note that φ_0 is invoked only in the first iteration and thus provides the correct initialisation. The formula ξ collects the transitions of those tuples in X according to the transition function δ of M. In each iteration exactly one configuration will be added to X because M is deterministic. The formula ξ is given by

$$\begin{aligned}\xi(X,q,h,i,b) \coloneqq \exists(q_{old}\colon \tau').\, \exists(h_{old}\colon \tau').\, \exists(i_{old}\colon \tau').\, \exists(b_{old}\colon \tau').\, \\ \exists(q_{new}\colon \tau').\, \exists(b_{new}\colon \tau').\, X(q_{old}, h_{old}, i_{old}, b_{old}) \land \\ \Big(\bigvee_{\delta(q_{old},b_{old})=(q_{new},b_{new},d)} q = q_{new} \land h = h_{old} + d \land i = i_{old} \land \\ \big((\neg i = h_{old} \land b = b_{old}) \lor (i = h_{old} \land b = b_{new})\big)\Big)\end{aligned}$$

where $h = h_{old} + d$ depends on d and is given by

$$h = h_{old} + d \coloneqq \begin{cases} next^\tau(h_{old}, h), & \text{if } d = L \\ next^\tau(h, h_{old}), & \text{if } d = R \\ h = hold, & \text{if } d = N \end{cases}$$

Because M terminates, the formula

$$\psi_0(q', h', i', b') \coloneqq [PFP\ \varphi(X,q,h,i,b)](q', h', i', b')$$

is guaranteed to define the relational description of the final configuration of M on input word w. Finally, the formula

$$\psi \coloneqq \exists(h' \colon \tau').\, \exists(i' \colon \tau').\, \exists(b' \colon \tau').\, \bigvee_{q' \in F} \psi_0(q', h', i', b')$$

defines the acceptance of M on input w. □

4.4.2 Encoding of Bisimulation Invariant HO(PFP)$^{k+1}$ in PHFL$^{k+1}_{tail}$

As mentioned in the introduction of this section the main idea is not to show directly that the lower bound of PHFL$^{k+1}_{tail}$ is k-EXPSPACE/$\sim$ but rather by detour over the bisimulation-invariant fragment of HO(PFP)$^{k+1}$. In the previous subsection we have seen that a k-EXPSPACE-bounded DTM can be performed by an HO(PFP)$^{k+1}$ formula. Encoding the bisimulation-invariant fragment of HO(PFP)$^{k+1}$ into PHFL$^{k+1}_{tail}$ combined with the knowledge that k-EXPSPACE is captured by HO(PFP)$^{k+1}$ leads to the lower bound of PHFL$^{k+1}_{tail}$. In Section 4.2 and Section 4.3 we have shown that the HO^{k+1} part can be encoded in PHFL^k. It is easy to prove that the encoded formulas are all tail-recursive[2]. It follows that the HO^{k+1} part can also be encoded in PHFL$^{k+1}_{tail}$. The PFP operator is the only kind of HO(PFP)$^{k+1}$ formula that we have to encode in this subsection to get the lower bound of PHFL$^{k+1}_{tail}$.

Before we give the definition of the transforming function, we define a PHFL formula for HO formulas that uses the PFP operator.

Definition 4.28. Let d be the constant as described in Remark 4.4 and X an HO variable of HO type $\tau = (\tau', \ldots, \tau')$ where $\tau' \neq \odot$. Furthermore, let Φ be a PHFL$^k_{tail}$ formula, then $PFP^\tau X.\, \Phi$ is a PHFL$^k_{tail}$ formula with dimension d defined as:

$$\begin{aligned} PFP^\tau X.\, \Phi \coloneqq \Big(\mu(F \colon T(\tau) \to \bullet).\, \lambda(X \colon T(\tau)).\, \big(X \wedge \forall^{\tau'} X_1.\, \cdots \forall^{\tau'} X_n. \\ ((\cdots(X X_1)\cdots) X_n \Leftrightarrow \Phi(X, X_1, \ldots, X_n)) \vee F(\Phi(X))\big)\Big) \bot_{T(\tau)} \end{aligned}$$

In case of $\tau = (\odot, \ldots, \odot)$ let $PFP^{(\odot,\ldots,\odot)} X.\, \Phi$ defined as:

$$\begin{aligned} PFP^{(\odot,\ldots,\odot)} X.\, \Phi \coloneqq \Big(\mu(F \colon \bullet \to \bullet).\, \lambda(X \colon \bullet).\, \big(X \wedge \forall_1 \cdots \forall_n \\ (X \Leftrightarrow \Phi(X)) \vee F(\Phi(X))\big)\Big) \bot \end{aligned}$$

[2]The PHFL^0 formula $\Phi_\sim$ (Example 2.19) is, indeed, not tail-recursive, but over finite LTS it is equivalent to a tail-recursive PHFL^1 formula [18].

Now we are able to define the function that translates a bisimulation invariant $\mathrm{HO(PFP)}^{k+1}$ formula to a $\mathrm{PHFL}^{k+1}_{tail}$ formula. Note that the encoding function defined in the following definition differs only in the fixpoint operators from the encoding function from Definition 4.20.

Definition 4.29. Define F as the function that maps a bisimulation invariant $\mathrm{HO(PFP)}^{k+1}$ formula φ to a $\mathrm{PHFL}^{k+1}_{tail}$ formula with dimension d, where d and s are the constants as described in Remark 4.4 and $\Phi_\sim$ is the formula of Example 2.19, then F is defined inductive on φ as follows:

$$
\begin{aligned}
F(p(X_i)) &\coloneqq p_{2s+i} \\
F(a(X_i, X_j)) &\coloneqq \langle a \rangle_{2s+i} \{(2s+i, 2s+j, \\
&\qquad 3, \ldots, d)\} \Phi_\sim \\
F(\Phi \vee \Psi) &\coloneqq F(\Phi) \vee F(\Psi) \\
F(\neg \Phi) &\coloneqq \neg F(\Phi) \\
F(\exists (X_i \colon \odot).\, \Phi) &\coloneqq \exists_{2s+i} F(\Phi) \\
F(\exists (X \colon \tau).\, \Phi) &\coloneqq \exists^\tau X.\, F(\Phi(X)) \\
F([PFP\ \Phi(X, X_{i_1}, \ldots, X_{i_n})](V_{j_1}, \ldots, V_{j_n})) &\coloneqq \{(j_1, \ldots, j_n, n+1, \ldots, d)\} \\
&\qquad PFP^{(\odot, \ldots, \odot)} X.\, F(\Phi) \\
F([PFP\ \Phi(X, X_1, \ldots, X_n)](V_1, \ldots, V_n)) &\coloneqq (\cdots (PFP^\tau X.\, F(\Phi))\, V_1) \cdots)\, V_n \\
F(X(X_{i_1}, \ldots, X_{i_n})) &\coloneqq \{(2s+i_1, \ldots, 2s+i_n, \\
&\qquad n+1, \ldots, d)\} X \\
F(X(X_1, \ldots, X_n)) &\coloneqq (\cdots (X\, X_1) \cdots)\, X_n
\end{aligned}
$$

The last step is to show that the semantics of a given $\mathrm{HO(PFP)}^{k+1}$ formula coincides with the semantics of the translated $\mathrm{PHFL}^{k+1}_{tail}$ formula. As mentioned in Section 4.1 without loss of generality the statement can be proven by consider only reduced LTS.

Lemma 4.30. *Let $f \geq 1$ and $k \geq 0$. For every bisimulation-invariant formula Φ of HO(PFP)$^{k+1}$ there is a PHFL$^{k+1}_{tail}$ formula Ψ such that the f-adic query $\mathcal{Q}^f_\Phi$ defined by Φ is equal to the f-adic query $\mathcal{Q}^f_\Psi$ defined by Ψ.*

Proof. This lemma can be proven by showing for all $\mathrm{HO(PFP)}^{k+1}$ formulas Φ with first-order variables $X_1, \ldots, X_q$, all reduced LTS $\mathcal{T} = (Q, \Sigma, P, \Delta, v)$ with respect to $\mathbf{q}_r = q_1, \ldots, q_r$ and all variable mappings η that it holds that $\mathcal{T}, \eta \models \Phi$ iff $\mathbf{q} = (\mathbf{q}_s, \mathbf{q}_s, \mathbf{q}_q, \mathbf{q}_r)$ and $\mathbf{q} \in [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}}$. Here $\mathbf{q}_s = q'_1, \ldots, q'_s$ is a sequence of s placeholders used for the interaction of second-order variables, $\mathbf{q}_q = \eta(X_1), \ldots, \eta(X_q)$ is a sequence of first-order variables that are mapped by η where $\eta(X_i) = q_0$ if X_i is bound to a quantifier, $q_0, q'_1, \ldots, q'_s \in Q$ are

arbitrary states, F is the formula function of Definition 4.20 and η_V the variable mapping of Definition 4.21. This statement can be proven by induction over formula Φ. Because the correctness proof of the non-fixpoint formulas is very similar to the correctness proof of Theorem 4.24 we concentrate us on showing correctness of the PFP operators.

- In case of $\Phi = [PFP\,\Psi(X, X_{i_1}, \ldots, X_{i_n})](V_{j_1}, \ldots, V_{j_n})$, where X is a free variable in Ψ of HO type $(\odot, \ldots, \odot)$ and $X_{i_1}, \ldots, X_{i_n}$ are free first-order variables of Ψ and $V_{j_1}, \ldots, V_{j_n}$ are first-order variables of Φ such that, without loss of generality, $i_1 < i_2, j_1 < j_2, \ldots, i_{n-1} < i_n, j_{n-1} < j_n$. All other cases working similar. Then it follows that $\mathcal{T}, \eta \models \Phi$ exactly then if $(\eta(V_{j_1}), \ldots, \eta(V_{j_n})) \in PFP(F_\Psi^{\mathcal{T},\eta})$. By Definition 2.12 this is the case when there is an m such that $F_\Psi^{\mathcal{T},\eta^m}(\varnothing) = F_\Psi^{\mathcal{T},\eta^{m+1}}(\varnothing)$ and $(\eta(V_{j_1}), \ldots, \eta(V_{j_n})) \in F_\Psi^{\mathcal{T},\eta^m}(\varnothing)$. By Definition 2.44 $(\eta(V_{j_1}), \ldots, \eta(V_{j_n})) \in F_\Psi^{\mathcal{T},\eta^m}(\varnothing)$ iff $\mathcal{T}, \eta \models \Psi(F_\Psi^{\mathcal{T},\eta^m}(\varnothing), \eta(V_{j_1}), \ldots, \eta(V_{j_n}))$. By induction hypothesis this is exactly the case when

$$(\mathbf{q}_s, \mathbf{q}_s, \eta(X_1), \ldots, \eta(X_{i_1-1}), \eta(X_{i_1}), \eta(X_{i_1+1}), \ldots, \eta(X_{i_n-1}), \eta(X_{i_n}), \eta(X_{i_n+1}), \ldots, \eta(X_q), \mathbf{q}_r) \in [\![F(\Psi)]\!]_{\mathcal{T}}^{\eta_V}.$$

  Note that X is set to $F_\Psi^{\mathcal{T},\eta^m}(\varnothing)$.
  If we use λ-approximation and β-reduction on $\mathrm{PFP}^{(\odot,\ldots,\odot)}X.\,F(\Psi)$ we can see that it can be summarized to

$$\varphi \coloneqq \bigvee_{i=0}^{m'} \Big(F(\Psi)^i \bot \wedge \forall_1 \cdots \forall_n \big(F(\Psi)^i \bot \Leftrightarrow F(\Psi)^{i+1} \bot \big) \Big)$$

  Note that

$$F_\Psi^{\mathcal{T},\eta^m}(\varnothing) = F(\Psi)^{m'} \bot.$$

  This holds because by induction over i obviously it holds that $\varnothing = \bot$ and $F_\Psi^{\mathcal{T},\eta}(\varnothing) = F(\Psi)\bot$. That means $F_\Psi^{\mathcal{T},\eta^{i+1}}(\varnothing) = F(\Psi^{i+1})\bot$ holds because by induction hypothesis it holds that $F_\Psi^{\mathcal{T},\eta^i}(\varnothing) = F(\Psi^i)\bot$ and $F_\Psi^{\mathcal{T},\eta}(F_\Psi^{\mathcal{T},\eta^i}(\varnothing)) = F(\Psi)\big(F(\Psi)^i \bot\big)$.
  If there existst an i such that $F_\Psi^{\mathcal{T},\eta^i}(\varnothing) = F_\Psi^{\mathcal{T},\eta^{i+1}}(\varnothing)$ this is exactly the case when the right conjunct of φ

$$\forall_1 \cdots \forall_n \big(\Psi^i(\bot) \Leftrightarrow \Psi^{i+1}(\bot) \big)$$

  holds. The left conjunct of φ returns then the set that we get through the i-th application of $F(\Psi)$ on the empty set.
  Because of the construction of variable mapping η_V and $(\eta(V_{j_1}), \ldots, \eta(V_{j_n})) \in \eta(X)$ the tuple $(\eta(V_{j_1}), \ldots, \eta(V_{j_n}), q'_{n+1}, \ldots, q'_s, \mathbf{q}_s, \mathbf{q}_q, \mathbf{q}_r)$ is

also in $\eta_V(X)$. That means it holds that $(\eta(V_{j_1}),\ldots,\eta(V_{j_n})) \in F^{\mathcal{T},\eta^i}_{\Psi}(\varnothing)$ with

$$F(\Phi) = \{(i_1,\ldots,i_n)\}(PFP^{(\odot,\ldots,\odot)}X.\,F(\Psi))$$

exactly then if

$$(\mathbf{q}_s,\mathbf{q}_s,\eta(X_1),\ldots,\eta(V_{j_1-1}),\eta(V_{j_1}),\eta(V_{j_1+1}),\ldots\eta(V_{j_n-1}),\\ \eta(V_{j_n}),\eta(V_{j_n+1}),\ldots,\eta(X_q),\mathbf{q}_r) \in [\![F(\Phi)]\!]^{\eta_V}_{\mathcal{T}}.$$

In the case that $PFP(F^{\mathcal{T},\eta}_{\Psi})$ returns the empty set because there is no m such that $F^{\mathcal{T},\eta^m}_{\Psi}(\varnothing) = F^{\mathcal{T},\eta^{m+1}}_{\Psi}(\varnothing)$, the right conjunct of φ is always false. Because of the least fixpoint operator in $PFP^{(\odot,\ldots,\odot)}X.\,F(\Psi)$ the iteration is finite and it also will return the empty set.

- In case of $\Phi = [PFP\,\Psi(X,X_1,\ldots,X_n)](V_1,\ldots,V_n)$, where X is a free variable in Ψ of HO type $(\tau,\ldots,\tau)$ and $X_1,\ldots,X_n$ are free variables of Ψ of type τ and $V_1,\ldots,V_n$ are free variables of Φ also of type τ, then it follows that $\mathcal{T},\eta \models \Phi$ exactly then if $(\eta(V_1),\ldots,\eta(V_n)) \in PFP(F^{\mathcal{T},\eta}_{\Psi})$. By Definition 2.12 this is exactly the case when there is an m such that $F^{\mathcal{T},\eta^m}_{\Psi}(\varnothing) = F^{\mathcal{T}^{m+1}}_{\Psi}(\varnothing)$ and $(\eta(V_1),\ldots,\eta(V_n)) \in F^{\mathcal{T},\eta^m}_{\Psi}(\varnothing)$. By Definition 2.44 $(\eta(V_1),\ldots,\eta(V_n)) \in F^{\mathcal{T},\eta^m}_{\Psi}(\varnothing)$ iff $\mathcal{T},\eta \models \Psi(F^{\mathcal{T}^m}_{\Psi}(\varnothing),\eta(V_1),\ldots,\eta(V_n))$. By induction hypothesis this is exactly the case when $\mathbf{q} \in [\![F(\Psi)]\!]^{\eta_V}_{\mathcal{T}}$. Note that X is set to $F^{\mathcal{T},\eta^m}_{\Psi}(\varnothing)$.
  If we use λ-approximation and β-reduction on $\mathrm{PFP}^{\tau}X.\,F(\Psi)$ we can see that it can be summarized to

$$\varphi \coloneqq \bigvee_{i=0}^{m'}\Big(F(\Psi)^i\bot_{T((\tau,\ldots,\tau))} \wedge \forall^{\tau'}X_1.\cdots\forall^{\tau'}X_n.\\ \big((\cdots(F(\Psi)^i\bot_{T((\tau,\ldots,\tau))}X_1)\cdots)X_n \Leftrightarrow\\ (\cdots(F(\Psi)^{i+1}(\bot_{T((\tau,\ldots,\tau))})X_1)\cdots)X_n\big)\Big)$$

  Note that

$$F^{\mathcal{T},\eta^m}_{\Psi}(\varnothing) = F(\Psi)^{m'}\bot_{T((\tau,\ldots,\tau))}.$$

  This holds because by induction over i obviously it holds that

$$\varnothing = \bot_{T((\tau,\ldots,\tau))}$$

  and

$$F^{\mathcal{T},\eta}_{\Psi}(\varnothing) = F(\Psi)\bot_{T((\tau,\ldots,\tau))}.$$

  That means

$$F^{\mathcal{T},\eta^{i+1}}_{\Psi}(\varnothing) = F(\Psi)^{i+1}\bot_{T((\tau,\ldots,\tau))}$$

holds because by induction hypothesis it holds that

$$F_\Psi^{\mathcal{T},\eta^i}(\varnothing) = F(\Psi)^i \bot_{T((\tau,\ldots,\tau))}$$

and

$$F_\Psi^{\mathcal{T},\eta}(F_\Psi^{\mathcal{T},\eta^i}(\varnothing)) = F(\Psi)\big(F(\Psi)^i \bot_{T((\tau,\ldots,\tau))}\big)$$

If there existst an i such that $F_\Psi^{\mathcal{T},\eta^i}(\varnothing) = F_\Psi^{\mathcal{T}^{i+1}}(\varnothing)$ this is exactly the case when the right conjunct of φ

$$\begin{aligned}&\forall^{\tau'} X_1.\ \cdots \forall^{\tau'} X_n.\\ &\big((\cdots(\Psi^i(\bot_{T((\tau,\ldots,\tau))})X_1)\cdots)X_n \Leftrightarrow\\ &(\cdots(\Psi^{i+1}(\bot_{T((\tau,\ldots,\tau))})X_1)\cdots)X_n\big)\end{aligned}$$

holds. The left conjunct of φ returns then the function that we get through the i-th application of Ψ on $\bot_{T((\tau,\ldots,\tau))}$.
Because of the construction of variable mapping η_V and $(\eta(V_1),\ldots,\eta(V_n)) \in \eta(X)$ it holds that $(\cdots\big(\eta_V(X)\,\eta_V(V_1)\big)\cdots)\,\eta_V(V_n) = Q^d$ and so

$$\mathbf{q} \in (\cdots\big(\eta_V(X)\,\eta_V(V_1)\big)\cdots)\,\eta_V(V_n).$$

That means it holds that $(\eta(V_1),\ldots,\eta(V_n)) \in F_\Psi^{\mathcal{T},\eta^i}(\varnothing)$ with

$$F(\Phi) = (\cdots((PFP^T((\tau,\ldots,\tau))X.\,F(\Phi))\,V_1)\cdots)\,V_n$$

exactly then if

$$\mathbf{q} \in [\![F(\Phi)]\!]_{\mathcal{T}}^{\eta_V}.$$

In the case that $PFP(F_\Psi^{\mathcal{T},\eta})$ returns the empty set because there is no m such that $F_\Psi^{\mathcal{T},\eta^m}(\varnothing) = F_\Psi^{\mathcal{T},\eta^{m+1}}(\varnothing)$, the right conjunct of φ is always false. Because of the least fixpoint operator in $PFP^{T((\tau,\ldots,\tau))}X.\,F(\Psi)$ the iteration is finite and it will return $\bot_{T((\tau,\ldots,\tau))}$.

□

The combination of Lemma 4.30, Lemma 4.27 and Theorem 3.14 proves the following theorem for $k > 0$. For $k = 0$ and $k = 1$ this statement was proven by M. Otto in [4] and by M. Lange and E. Lozes in [2].

Theorem 4.31. *Let $k \geq 0$. $PHFL_{tail}^{k+1}$ captures k-EXPSPACE/$\sim$ over labelled transition systems.*

Chapter 5

Conclusion

In this thesis, we contributed to descriptive complexity theory by relating any order of PHFL to the corresponding complexity class. In detail, we showed that PHFL^k captures the complexity class k-EXPTIME/$\sim$ for any $k > 1$ over finite labelled transition systems. Due to the fact that the statement above is also true for $k = 0$ [4] and $k = 1$ [2] we were able to verify that PHFL^k captures k-EXPTIME/$\sim$ for any $k \geq 0$ on finite labelled transition systems. Furthermore, it was showed that the logic PHFL^{k+1}_{tail} captures the complexity class k-EXPSPACE/$\sim$ for any $k > 1$. In analogy to the exponential time classes, it was also proven that PHFL^{k+1}_{tail} captures k-EXPSPACE/$\sim$ for any $k \geq 0$ on finite labelled transition systems [4] [2].

Since PHFL does not have existential and universal quantification over arbitrary higher-order relations a lot of effort had to be spent into the development of the encoding of the existential quantifiers of any order. To obtain higher-order quantification in PHFL we used the existential quantifiers of type $\tau = (\odot, \ldots, \odot)$ to define the order of domains of kind $D_{(\tau,\ldots,\tau)}(Q)$. This order was then be used to define a formula that returns the successor of a given element of $D_{(\tau,\ldots,\tau)}(Q)$ in respect to this order. Finally, we used this formula to define the existential quantifier of type $(\tau, \ldots, \tau)$. This procedure was applied to all possible types of HO. In this way we got higher-order quantification of any type in PHFL.

The presented results contribute to the understanding on these complexity classes, which opens the possibilities for additional research, especially for further characterization of k-EXPTIME and k-EXPSPACE. That could lead to a further research on the characterization of classes k-NEXPTIME/$\sim$. Those characterizations cannot be mapped to the encodings of k-EXPTIME/$\sim$ presented in this thesis. Another possibility may be the characterization of the polynomial hierarchy [19].

Acknowledgements

First of all I would like to thank my thesis advisor Prof. Dr. Martin Lange for transferring the topic to me. I am very thankful for the help of Florian Bruse. In a twentyfour-seven service he gave me a patient guidance and lot of proof reading. I also like to thank Andreas and Michael for their English proof readings. Furthermore, I want to express my gratitude to my wife Lisa, who supported me in this exhausting time with much patience, understanding and the care of our daughter. A special thanks goes to Richard who not only did a good English proof reading, but furthermore backed me up morally.

Bibliography

[1] R. Fagin. Generalized first-order spectra, and polynomial. time recognizable sets. *Complexity and Computation*, 7:43–73, 1974.

[2] M. Lange and E. Lozes. Capturing bisimulation-invariant complexity classes with higher-order modal fixpoint logic. In Josep Díaz, Ivan Lanese, and Davide Sangiorgi, editors, *IFIP TCS*, volume 8705 of *Lecture Notes in Computer Science*, pages 90–103. Springer, 2014.

[3] M. Viswanathan and R. Viswanathan. A higher order modal fixed point logic. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 512–528. Springer, 2004.

[4] M. Otto. Bisimulation-invariant ptime and higher-dimensional µ-calculus. *Theoretical Computer Science*, 224(1-2):237–265, 1999.

[5] C. M. Freire and A. T. C. Martins. The descriptive complexity of the deterministic exponential time hierarchy. *Electronic Notes in Theoretical Computer Science*, 269:71–82, 2011.

[6] N. Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[7] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[8] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

[9] V. Stoltenberg-Hansen, I. Lindström, and E. R. Griffor. *Mathematical Theory of Domains*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1994.

[10] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[11] J. E. Hopcroft and J. Ullman. *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*. Addison-Wesley, 3. edition, 1994. deutsche Ausgabe.

[12] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT (FOCS)*, pages 125–129. IEEE Computer Society, 1972.

[13] F. Bruse, M. Lange, and E. Lozes. Space-efficient fragments of higher-order fixpoint logic. In Matthew Hague and Igor Potapov, editors, *RP*, volume 10506 of *Lecture Notes in Computer Science*, pages 26–41. Springer, 2017.

[14] J. Van Benthem and K. Doets. *Higher-order logic*, pages 189–243. Springer Netherlands, Dordrecht, 2001.

[15] K.-D. Schewe and J. M. Turull Torres. Fixed-point quantifiers in higher order logics. In *Proceedings of the 2006 Conference on Information Modelling and Knowledge Bases XVII*, pages 237–244, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.

[16] R. Axelsson, M. Lange, and R. Somla. The complexity of model checking higher-order fixpoint logic. *Computing Research Repository*, abs/0704.3931, 2007.

[17] S. Abiteboul and V. Vianu. Computing with first-order logic. *Journal of Computer and System Sciences*, 50(2):309–335, 1995.

[18] M. Lange and E. Lozes. Capturing bisimulation-invariant complexity classes with higher-order modal fixpoint logic. available at http://carrick.fmv.informatik.uni-kassel.de/∼lozes/tcs14-long.pdf.

[19] Stockmeyer L. J. The polynomial-time hierarchy. *TCS*, 3(1):1–22, 1976.