Weak Systems of Explicit Mathematics

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Daria Spescha
von Zug und Pigniu

Leiter der Arbeit:
Prof. Dr. T. Strahm
Institut für Informatik und angewandte Mathematik

Weak Systems of Explicit Mathematics

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Daria Spescha
von Zug und Pigniu

Leiter der Arbeit:
Prof. Dr. T. Strahm
Institut für Informatik und angewandte Mathematik

Acknowledgements

I am indebted to various people who helped and supported me in different ways in the course of writing this thesis. I would like to express my gratitude to them.

First of all, I want to thank Prof. Thomas Strahm for his continuous supervision and patience. I am also grateful to Prof. Gerhard Jäger who made this thesis possible. I would like to thank the current and previous members of the TIL group for advice, interesting coffee breaks and lots of pool games in Münchenwiler. Among them I want to mention especially Peppo Brambilla, Jürg Krähenbühl, Roman Kuznets, Dieter Probst, David Steiner and Thomas Studer. Further, many thanks go to Prof. Andrea Cantini for his thorough reading of and valuable remarks on this thesis.

People from outside the academia also contributed a lot to this thesis. I am very grateful for and to all my great friends who made sure I did not forget about the illogical part of life and kept encouraging me whenever stubborn proofs gave me a hard time.

Last but not least, I want to thank my family. I am much indebted to my parents, Eusebius and Marilis, for their constant support and I owe a lot to my grandmothers, Agnes Spescha and Marie Baeriswyl.

Finally, I thank the Swiss National Science Foundation for its financial support.

Contents

Chapter 1

Prologue

The goal of this thesis is to investigate weak systems of Explicit Mathematics. In other words, the aim is to establish a characterisation of weak complexity classes by means of theories of Explicit Mathematics.

Explicit Mathematics was first introduced in the 1970s by Feferman [16, 17, 18] as a framework for Bishop-style constructivism. Originally, Explicit Mathematics was used to provide a constructive justification for strong (impredicative) systems. For example some inherently impredicative systems of set theory and second-order arithmetic can be reduced to theories of Explicit Mathematics. In recent years, Explicit Mathematics was also employed to compare (sub)systems of Peano Arithmetic. Although the first systems of Explicit Mathematics were intuitionistic theories, later theories based on classical logic were also studied.

Systems of Explicit Mathematics are second-order theories that deal with two kinds of objects: the basic elements are operations and the second order part consists of collections of operations called types. A distinguishing feature is that types are referenced by uniformly generated operations, so-called names. The connection between types and their names is established by the naming relation $\Re$.

The first order part is handled by the so-called Applicative Theories.

Although Applicative Theories have been developed as the basis for Explicit Mathematics, they turned out to be of independent interest. A characteristic of Applicative Theories is that operations can freely be applied to each other and also self-application is allowed, but the result might not be defined. Since undefinedness plays an important role, Explicit Mathematics is based on Beeson's Logic of Partial Terms.

In Applicative Theories, the focus is on the intensional aspect of the operations, whereas types are extensional constructs. While types are defined merely by their elements, the names provide information about the construction of the type they are referring to. This immediately implies that a type has several names. In the most popular theory EET+J, all types are generated from two basic ones, the type of natural numbers and the identity type, by closure under conjunction, complement, domains (existential quantification) and inverse images. Furthermore, there is a special type constructor called join that allows us to generate the disjoint union of a uniformly generated family of types.

The induction scheme of a theory usually has an effect on its strength. The most common induction principles are the so-called set induction (i.e. induction on Boolean functions over the natural numbers), type induction and full formula induction.

Systems of Explicit Mathematics are powerful and can be strengthened even more by potent principles such as inductive generations, some form of the axiom of choice or power types. The systems studied so far are ranging in strength from PRA to highly impredicative systems.

Explicit Mathematics also proved to be a convenient tool for studying logical frameworks for programming languages e.g. by Feferman [20, 21, 22]. Mostly functional programming languages have been considered, however also other programming languages were investigated. Studer [58] used the system EET+J to give formal semantics for Featherweight Java, a limited version of the programming language Java. Hayashi and Nakano [29] investigated the extraction of programs from proofs in the context of Feferman style Explicit Mathematics.

In computer science, however, we are more interested in feasible functions. The most familiar proof-theoretically weaker theories, i.e. theories of strength below PRA, are systems of Bounded Arithmetic of Buss [3]. They are weak fragments of Peano Arithmetic and are formulated in an extension of the language of PA. The weakening is achieved by restricting the use of quantifiers to bounded quantifiers in induction formulas.

A well-known recursion-theoretic approach to weaker complexity classes are function algebras as surveyed e.g. by Clote [11]. They start off from a given set of basic functions and generate functions by the closure under a set of functionals that serve as function combinators.

In this thesis, we follow a different approach by working in the setting of Explicit Mathematics. The advantage of working in this setting is the natural flavour of functions since all individual objects are operations and some basic operations are already an integral part of the theories, while even basic functions need tedious coding and bootstrapping in Bounded Arithmetic. In Explicit Mathematics, functions are represented by terms and the strength is determined by the terms that are provably defined for all inputs whereas in Bounded Arithmetic functions are defined by formulas and the strength is defined via representable functions.

Our goal is to develop theories whose proof-theoretic strength is below PRA while including type induction. In the presence of types, type induction appears to be the most natural induction principle. It entails formula induction restricted to the formula class for which comprehension is available.

So far, weak theories have only been studied for Applicative Theories –among others– by Strahm [53, 54] and Cantini [5, 6]. While the original formulation of Applicative Theories is based on natural numbers, Strahm's theories take binary words as the foundation. Furthermore, induction is constrained to a limited formula class. Strahm [53] proposed theories that correspond to the four complexity classes of functions computable

1. in polynomial time,
2. simultaneously in polynomial time and linear space,
3. in polynomial space and
4. in linear space.

In this thesis, we will use these theories extensively.

Krähenbühl [44] introduced a theory of full Explicit Mathematics of proof-theoretic strength PRA in the presence of type induction. The desired complexity is essentially achieved by omitting the complement type constructor.

We continue this approach and restrict the available type constructors even more. In addition to omitting complements, we replace the type of binary words by initial segments of binary words.

As mentioned above, the principle of disjoint union stands out as a type constructor with great expressive power. Informally, if we have an operation $f : a \to \Re$ where a is the name of a type, then we can construct the type $\mathsf{j}(a, f) := \{(x, y) : x \in a, y \in fx\}$. Due to its expressiveness, this principle is delicate to deal with in weak theories. However, we show that the join principle does not increase the proof-theoretic strength in intuitionistic logic while increasing the expressivity.

We are now going to give an outline of this thesis. In Chapter 2, we give an introduction to Explicit Mathematics and provide a brief overview on previous research in this field. Then, in Chapter 3, we describe the weak theories mentioned above. We give detailed proofs for the theory PET that corresponds to the polynomial time computable functions and adapt those for the other complexity classes. The proof of the lower bounds is done by embedding Strahm's weak Applicative Theories while the upper bounds are established by means of a model theoretic argument. Furthermore, we consider some extensions as studied by Cantini [6], among them we focus on the Uniformity Principle that allows us to introduce an additional type constructor for universal quantification.

Since the treatment of the join principle is more complicated for weak theories, we discuss its addition to PET in Chapter 4. First, we switch to

intuitionistic logic and reformulate the theory in sequent calculus style. Then we give a syntactical proof for the upper bounds by means of a realisability relation for positive formulas. We also sketch how to adapt this for the classical case without join. In Chapter 5 we summarise the results of this thesis.

In the appendix, we summarise some basics that are assumed to be known in this thesis together with the formal introduction of a pairing function extensively used in this thesis. We first recapitulate Beeson's Logic of Partial Terms in Appendix A for completeness. In Appendix B, we give a résumé of the function algebra characterisations of the weak complexity classes since they play an important role in the context of this thesis. Finally, in Appendix C, we formally introduce pairing and projection functions in the class of functions computable simultaneously in polynomial time and linear space. The existence of these functions is well-known, see e.g. Ferreira [26]. However, the detailed construction has not yet been spelled out in the setting of Applicative Theories.

Chapter 2

Introduction to Explicit Mathematics

Explicit Mathematics was first developed in the early 1970s by Solomon Feferman [16, 18] as a basis for Bishop-style constructivism. Feferman [17] intended to provide a unifying framework to relate set-theoretic and recursion-theoretic interpretations. Later, Explicit Mathematics turned out to be a useful tool for the proof-theoretic analysis of (sub)systems of second order arithmetic.

Systems of Explicit Mathematics are second order systems and consist basically of two different kinds of objects, namely individuals (operations) and types. The operational core is formed by the so-called Applicative Theories. They provide the rules how individuals are applied to each other. Full systems of Explicit Mathematics add a type structure to the underlying applicative structure. Types are collections of individuals and are referenced by uniformly constructed operations, also called (explicit) names.

Types are extensional objects, i.e. two types are equal if they contain the same elements. By contrast, the first order part focuses on the intensional aspect of operations. The names of the types are uniformly cre-

ated and indicate how the types they are referring to were constructed.

The most famous original system T_0 introduced by Feferman [16] is formulated in intuitionistic logic, whereas later also classical theories were investigated. The underlying basis is Beeson's Logic of Partial Terms LPT as recapitulated in Appendix A. In this setting, application of individuals to each other is not necessarily total. In other words, there is the possibility of the result of an application being undefined.

The focus of this thesis lies on weak systems of Explicit Mathematics. Among the systems previously studied are theories corresponding to primitive recursive arithmetic PRA and Peano arithmetic PA.

In this chapter, we give a short overview over some relevant aspects of Explicit Mathematics while emphasising comparatively weak systems. Following the structure of Explicit Mathematics, we introduce several Applicative Theories in Section 2.1. Afterwards, we turn to the type structure and recapitulate different axiomatisations of types and names in Section 2.2. Since we use model-theoretic arguments later, we discuss the semantics in Section 2.3. In Section 2.4 we sketch some interesting extensions, some of which also play a role in later chapters. Finally, we briefly mention some related work in Section 2.5.

2.1 Applicative Theories

Applicative Theories form the basis of Explicit Mathematics. They provide the framework for the first order part and define the behaviour of elementary operations. At the beginning, they were merely developed as an underlying system while studying type construction. However, Applicative Theories turned out to offer interesting proof-theoretic results independent of the type structure. For more details refer e.g. to the overview article by Jäger, Kahle and Strahm [37] or to Strahm [52].

In Applicative Theories, all objects are operations which may arbitrarily be applied to each other. Also self-application is allowed, although not necessarily total. These systems focus on the intensional aspect of

functions, in contrast to set-theoretic approaches. The natural numbers are often employed to build the basic mathematical structure.

Several interesting extensions of Applicative Theories where studied, such as a μ-operator in Feferman and Jäger [25, 24] or truth theories based on similar systems among others by Cantini [7] or by Kahle [41]. Also total applicative theories where the Logic of Partial Terms is replaced by ordinary first order logic was investigated e.g. by Jäger and Strahm [35]. For more details on these extensions, the reader is referred to the respective literature.

Applicative Theories were discovered to be useful for characterising weak complexity classes like polynomial time computable functions by Cantini [5, 6] and Strahm [56, 57, 54]. Especially in [53], Strahm studied four systems corresponding to the complexity classes for functions computable in polynomial time, simultaneously in polynomial time and linear space, in linear space, and in polynomial space. Those will be crucial in this thesis. In contrast to the stronger systems, binary words turned out to provide the more convenient basic structure than the natural numbers.

In Section 2.1.1, we first present the Basic Theory of Operations and Numbers as the standard formulation and the foundation for most systems of Explicit Mathematics. Different induction schemes used in this context are then presented in Section 2.1.2 and some important extensions are recapitulated in Section 2.1.3. Finally, the above mentioned weak systems are summarised in Section 2.1.4. For all these systems, the underlying logic is Beeson's Logic of Partial Terms as presented in Beeson [1]. To make this thesis self-contained, the axioms and rules are summarised in Appendix A.

2.1.1 The Basic Theory of Operations and Numbers BON

We recapitulate in this section the most common applicative theory, the **B**asic Theory of **O**perations and **N**umbers BON. Its axioms define the general operations of combinatory algebra and pairing and, in addition,

deal with the basic properties of natural numbers. The theory BON as presented here follows the notation developed e.g. in Feferman and Jäger [25, 24]. We are following the general outline of Jäger [34] where the presented results concerning BON are taken from.

The language $\mathcal{L}_{\mathsf{N}}$ of BON contains countably many (individual) variables $a, b, c, f, g, h, x, y, z, \ldots$ (possibly with subscripts) and the following constants: k, s (combinators), p, p_0, p_1 (pairing and projections), $\mathsf{0}$ (zero), s_{N} (numerical successor), p_{N} (numerical predecessor), d_{N} (definition by cases for numbers), and r_{N} (recursion). Furthermore, the language comprises a binary function symbol $\cdot$ (application), and a unary relation symbol N (natural numbers) is added to the relations $\downarrow$ (defined) and $=$ (equality) as given by LPT (c.f. Appendix A). Application is written in infix notation and $(a \cdot b)$ is usually abbreviated (ab) or just ab. Moreover application is associative to the left, i.e. $a_0 a_1 \ldots a_n$ stands for $(\ldots(a_0 a_1)\ldots a_n)$. Furthermore, we use the following abbreviations:

$$\begin{aligned} t' &:= \mathsf{s}_{\mathsf{N}} t & 1 &:= \mathsf{0}' \\ (s,t) &:= \mathsf{p} s t & (t)_i &:= \mathsf{p}_i t \quad (i = 0, 1) \\ (t_0) &:= t_0 & (t_0, \ldots, t_{n+1}) &:= ((t_0, \ldots, t_n), t_{n+1}) \end{aligned}$$

Concerning the predicate N, the following shortcuts are used:

$$\begin{aligned} t \in \mathsf{N} &:= \mathsf{N}(t) \\ (\exists x \in \mathsf{N}) A[x] &:= \exists x (x \in \mathsf{N} \wedge A[x]) \\ (\forall x \in \mathsf{N}) A[x] &:= \forall x (x \in \mathsf{N} \to A[x]) \\ t : \mathsf{N} \mapsto \mathsf{N} &:= (\forall x \in \mathsf{N})(tx \in \mathsf{N}) \\ t : \mathsf{N}^{n+1} \mapsto \mathsf{N} &:= (\forall x \in \mathsf{N})(tx : \mathsf{N}^n \mapsto \mathsf{N}) \end{aligned}$$

For BON, the strictness axioms of LPT result in the following:

(D1) $t\downarrow$ if t is a variable or a constant.

(D2) $(s \cdot t)\downarrow \to s\downarrow \wedge t\downarrow$.

(D3) $t_1 = t_2 \to t_1\downarrow \wedge t_2\downarrow$.

(D4) $\mathsf{N}(t) \to t\!\downarrow$.

The theory BON contains axioms of common combinatory algebra and consists of axioms defining the behaviour of the basic operators such as pairing, definition by cases and primitive recursion. Furthermore, the natural numbers are specified as the closure of $\mathsf{0}$ under the successor operator $\mathsf{s_N}$. Formally, the axioms of BON are spelled out as follows:

I. Partial Combinatory Algebra

(1) $\mathsf{k}xy = x$

(2) $\mathsf{s}xy\!\downarrow \wedge\ \mathsf{s}xyz \simeq xz(yz)$

II. Pairing and Projection

(3) $\mathsf{p}_0(x,y) = x \wedge \mathsf{p}_1(x,y) = y$

III. Natural Numbers

(4) $\mathsf{0} \in \mathsf{N} \wedge (\forall x \in \mathsf{N})(x' \in \mathsf{N})$

(5) $(\forall x \in \mathsf{N})(x' \neq \mathsf{0} \wedge \mathsf{p_N}(x') = x)$

(6) $(\forall x \in \mathsf{N})(x \neq \mathsf{0} \to \mathsf{p_N}x \in \mathsf{N} \wedge (\mathsf{p_N}x)' = x)$

IV. Definition by Numerical Cases

(7) $u \in \mathsf{N} \wedge v \in \mathsf{N} \wedge u = v \to \mathsf{d_N}xyuv = x$

(8) $u \in \mathsf{N} \wedge v \in \mathsf{N} \wedge u \neq v \to \mathsf{d_N}xyuv = y$

V. Primitive Recursion on N

(9) $(f : \mathsf{N} \mapsto \mathsf{N}) \wedge (g : \mathsf{N}^3 \mapsto \mathsf{N}) \to (\mathsf{r_N}fg : \mathsf{N}^2 \mapsto \mathsf{N})$

(10) $(f : \mathsf{N} \mapsto \mathsf{N}) \wedge (g : \mathsf{N}^3 \mapsto \mathsf{N}) \wedge x \in \mathsf{N} \wedge y \in \mathsf{N} \wedge h = \mathsf{r_N}fg$
$\to hx\mathsf{0} = fx \wedge hxy' = gxy(hxy)$

We now recapitulate some crucial properties of BON. Theorem 2.1 is common in the setting of Explicit Mathematics and follows from the axioms about combinatory algebra, see e.g. Beeson [1], Feferman [16] or Hindley and Seldin [30]. It states that λ abstraction of common λ calculus can be simulated in this setting.

Theorem 2.1 (λ Abstraction) *For every $\mathcal{L}_{\mathsf{N}}$ term t and any variable x, there is a term $(\lambda x.t)$, such that the following holds for any term s:*

1) $FV(\lambda x.t) = FV(t) \setminus \{x\}$

2) $\mathsf{BON} \vdash (\lambda x.t)\downarrow$

3) $\mathsf{BON} \vdash (\lambda x.t)x \simeq t$

4) $\mathsf{BON} \vdash s\downarrow \rightarrow (\lambda x.t)s \simeq t[s/x]$

Despite the above theorem, λ abstraction is not simulated to the full extent. In conventional λ calculus, the term $(\lambda x.t)[s/y]$ is equal to $(\lambda x.t[s/y])$. By contrast, a counter example for this can be constructed in BON. However, the following substitution principle does hold for BON:

Lemma 2.2 *For all $\mathcal{L}_{\mathsf{N}}$ terms s and t and any two distinct variables x and y, we have*

$$\mathsf{BON} \vdash (\lambda x.t)[s/y]x \simeq t[s/y]$$

2.1.2 Induction Schemes over BON

Systems of Explicit Mathematics are usually equipped with an induction scheme over the natural numbers. The choice of the induction generally influences or even determines the proof-theoretic strength of the system. Thus we present two different (common) induction schemes over BON, resulting in theories with different proof-theoretic strength. The first one makes induction for arbitrary formulas available, whereas the second one restricts induction to so-called subsets of the natural numbers and is therefore weaker.

Before stating the induction schemas, the concept of a subset of the natural numbers has to be introduced. A subset a of N, denoted by the

formula Set_{N}, shall be seen as its characteristic function:

$$\begin{aligned} Set_{\mathsf{N}}(a) &:= (\forall x \in \mathsf{N})(ax = 0 \vee ax = 1) \\ a \in Set_{\mathsf{N}} &:= Set_{\mathsf{N}}(a) \\ b \,\varepsilon\, a &:= ab = 0 \end{aligned}$$

Now we are ready to introduce two schemes for complete induction on N, namely Set Induction and Formula Induction:

Set Induction on N For all $\mathcal{L}_{\mathsf{N}}$ terms t:

$$(\mathsf{S\text{-}I_N}) \qquad t \in Set_{\mathsf{N}} \wedge 0 \,\varepsilon\, t \wedge (\forall x \in \mathsf{N})(x \,\varepsilon\, t \to x' \,\varepsilon\, t) \to (\forall x \in \mathsf{N})(x \,\varepsilon\, t)$$

Formula Induction on N For all formulas $A[x]$ of $\mathcal{L}_{\mathsf{N}}$:

$$(\mathsf{F\text{-}I_N}) \qquad A[0] \wedge (\forall x \in \mathsf{N})(A[x] \to A[x']) \to (\forall x \in \mathsf{N})A[x]$$

Remark 2.3 Axioms (9) and (10) are superfluous in the presence of $(\mathsf{F\text{-}I_N})$. To be precise, there is a closed term $\mathrm{rec}_{\mathsf{N}}$ (not containing $\mathsf{r_N}$) such that the theory $\mathsf{BON} \setminus \{(9), (10)\} + (\mathsf{F\text{-}I_N})$ proves that

1) $(f : \mathsf{N} \mapsto \mathsf{N}) \wedge (g : \mathsf{N}^3 \mapsto \mathsf{N}) \to (\mathrm{rec}_{\mathsf{N}}\, fg : \mathsf{N}^2 \mapsto \mathsf{N})$

2) $(f : \mathsf{N} \mapsto \mathsf{N}) \wedge (g : \mathsf{N}^3 \mapsto \mathsf{N}) \wedge x \in \mathsf{N} \wedge y \in \mathsf{N} \wedge h = \mathrm{rec}_{\mathsf{N}}\, fg$
$\to hx0 = fx \wedge hxy' = gxy(hxy)$

As mentioned above, the proof-theoretic strength of BON extended by one of the induction schemes above depends on the choice of the schema:

Theorem 2.4 *The proof-theoretic strength of* BON *plus induction was established as follows:*

$$\begin{aligned} \mathsf{BON} + (\mathsf{S\text{-}I_N}) &\equiv \mathsf{PRA} \\ \mathsf{BON} + (\mathsf{F\text{-}I_N}) &\equiv \mathsf{PA} \end{aligned}$$

2.1.3 Extensions of BON

Several additional axioms can be added to BON resulting in interesting theories. In this section, we briefly mention the most common ones, especially totality and extensionality of operations will be used later in this thesis.

Applicative Theories are based on the Logic of Partial Terms where operations are partial and terms need not be defined. However, sometimes systems with totality of application are better suited. For this purpose, we define the Totality Axiom:

(Tot) $\quad \forall x \forall y (xy\downarrow)$

In the presence of (Tot), every term is provably defined, i.e. for any term t, $\mathsf{BON} + (\mathsf{Tot}) \vdash t\downarrow$. Therefore, the underlying Logic of Partial Terms can be replaced by ordinary first-order predicate logic.

As mentioned above, Applicative Theories concentrate on the intensional aspect of operation. At times however, extensionality for operations is desired. Therefore we can add an axiom stating that two operations are equal if they produce the same result for all arguments:

(Ext) $\quad \forall f \forall g (\forall x (fx \simeq gx) \rightarrow f = g)$

A powerful means for generating functions from the base operations is via definition by cases. Yet, the built-in operator $\mathsf{d_N}$ works on natural numbers only. To rectify this deficiency, we can extend $\mathcal{L}_\mathsf{N}$ by a new constant $\mathsf{d_V}$ for definition by cases on the universe and add the following axiom

($\mathsf{d_V}$) $\quad (u = v \rightarrow \mathsf{d_V} xyuv = x) \wedge (u \neq v \rightarrow \mathsf{d_V} xyuv = y)$

Definition by cases on the universe is quite a strong principle, causing conflicts with several other schemes as indicated below. We quickly recapitulate the following ontological properties for BON:

Theorem 2.5 *We have the following ontological properties:*

1) $\mathsf{BON} + (\mathsf{d_V}) + (\mathsf{Ext})$ *is inconsistent.*

2) $\mathsf{BON} + (\mathsf{d_V}) + (\mathsf{Tot})$ *is inconsistent.*

2.1.4 Weak Applicative Theories

As mentioned before, Strahm –among others– studied different weak Applicative Theories. He presented a first system corresponding to polynomial time computable functions in [57]. Later, in [53], a refined version of this system was introduced together with systems for the complexity classes of functions computable simultaneously in polynomial time and linear space, in polynomial space and in linear space (c.f. Appendix B). He then investigated functionals of higher types and proved in [54] that the provably total type two functionals of PT coincide with the basic feasible functionals as defined by Melhorn [47].

In this section, we present the four weak theories as introduced by Strahm. As we will make heavy use of these systems later, we summarise the crucial results in sufficient detail. Unless stated otherwise, the definitions and theorems are recapitulated from Strahm [53].

When working with systems of strength lower than PRA, it is often more convenient to work with binary words $\mathbb{W}$ instead of natural numbers. In our context, binary words are considered as strings consisting of 0 and 1, and not as binary representations of natural numbers.

The language $\mathcal{L}_{\mathsf{W}}$ is very similar to $\mathcal{L}_{\mathsf{N}}$, it differs mainly in the constants which are intended to deal with binary words. $\mathcal{L}_{\mathsf{W}}$ comprises the following constants: k, s (combinators), p, p_0, p_1 (pairing and projections), ϵ (empty word), s_0 and s_1 (successors), $\mathsf{p_W}$ (predecessor), s_ℓ and p_ℓ (lexicographic successor and predecessor), $\mathsf{d_W}$ (definition by cases on binary words), $\mathsf{c}_\subseteq$ (initial subword), $\mathsf{l_W}$ (tally length), as well as $*$ and $\times$ (concatenation and multiplication). Similar to $\mathcal{L}_{\mathsf{N}}$, $\mathcal{L}_{\mathsf{W}}$ adds the binary function symbol $\cdot$ (application), and a unary relation symbol W (binary

words) to those of LPT.

We use the same abbreviations as introduced for BON and define the additional shortcuts:

$$0 := \mathsf{s}_0\epsilon \qquad 1 := \mathsf{s}_1\epsilon$$
$$s \subseteq t := \mathsf{c}_{\subseteq}st = 0 \qquad s \leq t := \mathsf{l}_{\mathsf{W}}s \subseteq \mathsf{l}_{\mathsf{W}}t$$

Furthermore, the following shorthand notations are used with respect to the predicate W where $\vec{s} = s_1, \ldots, s_n$:

$$\begin{aligned}
\vec{s} \in \mathsf{W} &:= \mathsf{W}(s_1) \wedge \cdots \wedge \mathsf{W}(s_n),\\
\mathsf{W}_a(s) &:= (\mathsf{W}(s) \wedge s \leq a),\\
\vec{s} \in \mathsf{W}_a &:= \mathsf{W}_a(s_1) \wedge \cdots \wedge \mathsf{W}_a(s_n),\\
(\exists x \in \mathsf{W})A &:= (\exists x)(x \in \mathsf{W} \wedge A),\\
(\forall x \in \mathsf{W})A &:= (\forall x)(x \in \mathsf{W} \to A),\\
(\exists x \leq t)A &:= (\exists x \in \mathsf{W})(x \leq t \wedge A),\\
(\forall x \leq t)A &:= (\forall x \in \mathsf{W})(x \leq t \to A),\\
(t : \mathsf{W} \mapsto \mathsf{W}) &:= (\forall x \in \mathsf{W})(tx \in \mathsf{W}),\\
(t : \mathsf{W}^{m+1} \mapsto \mathsf{W}) &:= (\forall x \in \mathsf{W})(tx : \mathsf{W}^m \mapsto \mathsf{W}).
\end{aligned}$$

We first recapitulate the basic theory B which is used as the foundation for all four theories later. Besides the strictness axioms analogous to BON, it consists of the following axiom groups defining the behaviour of the built-in operators and predicates:

I. Partial Combinatory Algebra and Pairing

(1) $\mathsf{k}xy = x$,

(2) $\mathsf{s}xy\downarrow \wedge \mathsf{s}xyz \simeq xz(yz)$,

(3) $\mathsf{p}_0(x, y) = x \wedge \mathsf{p}_1(x, y) = y$.

II. Definition by Cases on W

(4) $a \in \mathsf{W} \wedge b \in \mathsf{W} \wedge a = b \to \mathsf{d}_{\mathsf{W}}xyab = x$,

(5) $a \in \mathsf{W} \wedge b \in \mathsf{W} \wedge a \neq b \rightarrow \mathsf{d}_{\mathsf{W}} xyab = y.$

III. Closure, Binary Successors and Predecessor

(6) $\epsilon \in \mathsf{W} \wedge (\forall x \in \mathsf{W})(\mathsf{s}_0 x \in \mathsf{W} \wedge \mathsf{s}_1 x \in \mathsf{W}),$

(7) $\mathsf{s}_0 x \neq \mathsf{s}_1 y \wedge \mathsf{s}_0 x \neq \epsilon \wedge \mathsf{s}_1 x \neq \epsilon,$

(8) $\mathsf{p}_{\mathsf{W}} : \mathsf{W} \mapsto \mathsf{W} \wedge \mathsf{p}_{\mathsf{W}} \epsilon = \epsilon,$

(9) $x \in \mathsf{W} \rightarrow \mathsf{p}_{\mathsf{W}}(\mathsf{s}_0 x) = x \wedge \mathsf{p}_{\mathsf{W}}(\mathsf{s}_1 x) = x,$

(10) $x \in \mathsf{W} \wedge x \neq \epsilon \rightarrow \mathsf{s}_0(\mathsf{p}_{\mathsf{W}} x) = x \vee \mathsf{s}_1(\mathsf{p}_{\mathsf{W}} x) = x.$

IV. Lexicographic Successor and Predecessor

(11) $\mathsf{s}_\ell : \mathsf{W} \mapsto \mathsf{W} \wedge \mathsf{s}_\ell \epsilon = 0,$

(12) $x \in \mathsf{W} \rightarrow \mathsf{s}_\ell(\mathsf{s}_0 x) = \mathsf{s}_1 x \wedge \mathsf{s}_\ell(\mathsf{s}_1 x) = \mathsf{s}_0(\mathsf{s}_\ell x),$

(13) $\mathsf{p}_\ell : \mathsf{W} \mapsto \mathsf{W} \wedge \mathsf{p}_\ell \epsilon = \epsilon,$

(14) $x \in \mathsf{W} \rightarrow \mathsf{p}_\ell(\mathsf{s}_\ell x) = x,$

(15) $x \in \mathsf{W} \wedge x \neq \epsilon \rightarrow \mathsf{s}_\ell(\mathsf{p}_\ell x) = x.$

V. Initial Subword Relation

(16) $x \in \mathsf{W} \wedge y \in \mathsf{W} \rightarrow \mathsf{c}_{\subseteq} xy = 0 \vee \mathsf{c}_{\subseteq} xy = 1,$

(17) $x \in \mathsf{W} \rightarrow (x \subseteq \epsilon \leftrightarrow x = \epsilon),$

(18) $x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge y \neq \epsilon \rightarrow (x \subseteq y \leftrightarrow x \subseteq \mathsf{p}_{\mathsf{W}} y \vee x = y).$

(19) $x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge z \in \mathsf{W} \wedge x \subseteq y \wedge y \subseteq z \rightarrow x \subseteq z$

VI. Tally Length of Binary Words

(20) $\mathsf{l}_{\mathsf{W}} : \mathsf{W} \mapsto \mathsf{W} \wedge \mathsf{l}_{\mathsf{W}} \epsilon = \epsilon,$

(21) $x \in \mathsf{W} \rightarrow \mathsf{l}_{\mathsf{W}}(\mathsf{s}_0 x) = \mathsf{s}_1(\mathsf{l}_{\mathsf{W}} x) \wedge \mathsf{l}_{\mathsf{W}}(\mathsf{s}_1 x) = \mathsf{s}_1(\mathsf{l}_{\mathsf{W}} x),$

(22) $x \in \mathsf{W} \wedge \mathsf{l}_{\mathsf{W}}(x) = x \rightarrow \mathsf{l}_{\mathsf{W}}(\mathsf{s}_\ell x) = \mathsf{s}_1 x,$

(23) $x \in \mathsf{W} \wedge \mathsf{l}_{\mathsf{W}}(x) \neq x \rightarrow \mathsf{l}_{\mathsf{W}}(\mathsf{s}_\ell x) = \mathsf{l}_{\mathsf{W}}(x),$

(24) $x \in \mathsf{W} \wedge y \in \mathsf{W} \rightarrow x \leq y \vee y \leq x.$

Analogous to BON, we can simulate λ abstraction by means of k and s.

Lemma 2.6 (λ Abstraction) *For each term t and all variables x, there is a term $(\lambda x.t)$ whose free variables are those of t except x and such that* B *proves*

$$(\lambda x.t)\downarrow \wedge (\lambda x.t)x \simeq t$$

We generalise λ abstraction to several arguments as usual by writing $(\lambda x_1 \cdots x_n.t)$ for $(\lambda x_1.\cdots(\lambda x_n.t))$.

As a consequence of the enclosure of combinatory algebra, the existence of a recursion or fixpoint operator is also provable in B.

Lemma 2.7 (Recursion / Fixed point operator) *There exists a closed term* fix *such that* B *proves*

$$\mathsf{fix}f\downarrow \wedge \mathsf{fix}fx \simeq f(\mathsf{fix}f)x$$

Two basic operations on binary words are missing in B, namely word concatenation and multiplication, which are usually written in infix notation. Concatenation $*$ appends the second word to the first one as expected. Multiplication $a \times b$ repeats the first word a length of b many times. Formally, this is specified by the following axioms:

VII. Word Concatenation

(25) $* : \mathsf{W}^2 \mapsto \mathsf{W}$,

(26) $x \in \mathsf{W} \rightarrow x * \epsilon = x$,

(27) $x \in \mathsf{W} \wedge y \in \mathsf{W} \rightarrow x * (\mathsf{s}_i y) = \mathsf{s}_i(x * y) \quad (i = 0, 1)$.

VIII. Word Multiplication

(28) $\times : \mathsf{W}^2 \mapsto \mathsf{W}$,

(29) $x \in \mathsf{W} \rightarrow x \times \epsilon = \epsilon$,

(30) $x \in \mathsf{W} \wedge y \in \mathsf{W} \rightarrow x \times (\mathsf{s}_i y) = (x \times y) * x \quad (i = 0, 1)$.

For convenience, we will write $\mathsf{B}(*)$ for B expanded by axiom group VII. and $\mathsf{B}(*, \times)$ for adding VII. and VIII. to B.

In the presence of the axioms about multiplication $\times$, the tally length can be defined as the shortcut

$$\mathsf{l_W}s := 1 \times s$$

and the axioms of group VI. are derivable with suitable induction. We will heavily work with a theory with multiplication later and are interested in keeping the number of axioms as minimal as possible. Therefore, we let BOW stand for the theory comprising axiom groups I.–III., V. without (19), VII., and VIII.

Before we are able to state the induction principles for the desired theories, we first have to define the two formula classes Σ^b_W and Σ^{b-}_W, where positive formulas are formulas containing neither negation ($\neg$) nor implication ($\rightarrow$).

Definition 2.8 (Formula classes $\boldsymbol{\Sigma^b_\mathsf{W}}$ and $\boldsymbol{\Sigma^{b-}_\mathsf{W}}$)
A formula $A[f, x]$ belongs to the formula class Σ^b_W (Σ^{b-}_W) iff it is of the form $(\exists y \leq fx)B[f, x, y]$ where B is positive and does not contain W (and does not contain the quantifier $\forall$). ⊛

In contrast to ($\mathsf{F\text{-}I_N}$) where induction was allowed for all formulas, the induction schemes are in this context limited to formulas of the class Σ^b_W. For any formula $A[x] \equiv (\exists y \leq fx)B[f, x, y]$ in Σ^b_W we have

$$(\Sigma^b_\mathsf{W}\text{-}\mathsf{I_W}) \quad f : \mathsf{W} \mapsto \mathsf{W} \wedge A[\epsilon] \wedge (\forall x \in \mathsf{W})(A[\mathsf{p_W}x] \rightarrow A[x]) \rightarrow (\forall x \in \mathsf{W})A[x]$$

$$(\Sigma^b_\mathsf{W}\text{-}\mathsf{I}_\ell) \quad f : \mathsf{W} \mapsto \mathsf{W} \wedge A[\epsilon] \wedge (\forall x \in \mathsf{W})(A[\mathsf{p}_\ell x] \rightarrow A[x]) \rightarrow (\forall x \in \mathsf{W})A[x]$$

We are now ready to state Strahm's theories designed to correspond to the complexity classes mentioned before:

$$\mathsf{PT} := \mathsf{B}(*, \times) + (\Sigma^b_\mathsf{W}\text{-}\mathsf{I_W}) \qquad \mathsf{PTLS} := \mathsf{B}(*) + (\Sigma^b_\mathsf{W}\text{-}\mathsf{I_W})$$

$$\mathsf{PS} := \mathsf{B}(*, \times) + (\Sigma^b_\mathsf{W}\text{-}\mathsf{I}_\ell) \qquad \mathsf{LS} := \mathsf{B}(*) + (\Sigma^b_\mathsf{W}\text{-}\mathsf{I}_\ell)$$

For T any one of these four theories, T^- is defined as the corresponding theory with the only difference that the induction scheme is limited to Σ^{b-}_{W} formulas instead of Σ^b_{W} formulas. Furthermore, let us mention that PT is equivalent to $\mathsf{BOW} + (\Sigma^b_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$.

For these weak theories, the proof theoretic strength is defined via the notion of provably total functions. Therefore, we first need to formally define the concept of a provably total function. Before we can do this, we have to observe that for every binary word $w \in \mathbb{W}$, there is a canonical closed term $\overline{w}$ ("worderal") representing w. $\overline{w}$ is constructed from the empty word ϵ by applications of the successor functions s_0 and s_1.

Definition 2.9 (Provably total function)
A function $F : \mathbb{W}^n \to \mathbb{W}$ is called provably total in an $\mathcal{L}_{\mathsf{W}}$ theory T iff there exists a closed term t_F such that

1) $\mathsf{T} \vdash t_F : \mathsf{W}^n \mapsto \mathsf{W}$ and
2) $\mathsf{T} \vdash t_F\overline{w_1}\cdots\overline{w_n} = \overline{F(w_1 \cdots w_n)}$ for all $w_1, \ldots, w_n \in \mathbb{W}$ ⊛

In this definition, the first condition is crucial as the second condition is provable for all recursive functions in PT.

In Strahm [53], the lower bounds for PT and PTLS are established by deriving a natural form of bounded recursion on notation within the theory. In contrast to BON, the recursion operator is not built-in by the axioms. For its formulation, a cut-off operator is required. Informally speaking, $t \mid s$ is t if $t \leq s$ and s otherwise. More formally, we can make use of definition by cases d_{W} on W and the characteristic function $\mathsf{c}_{\subseteq}$, i.e. $t \mid s$ is simply an abbreviation for $\mathsf{d}_{\mathsf{W}}ts(\mathsf{c}_{\subseteq}(\mathsf{l}_{\mathsf{W}}t)(\mathsf{l}_{\mathsf{W}}s))0$.

Lemma 2.10 (Bounded recursion on notation) *There is a closed term* r_{W} *in* $\mathcal{L}_{\mathsf{W}}$ *such that* $\mathsf{B} + (\Sigma^b_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ *proves*

$$\begin{aligned} f : \mathsf{W} \mapsto \mathsf{W} \wedge g : \mathsf{W}^3 \mapsto \mathsf{W} \wedge b : \mathsf{W}^2 \mapsto \mathsf{W} \\ \to (\mathsf{r}_{\mathsf{W}}fgb : \mathsf{W}^2 \mapsto \mathsf{W} \wedge [x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge y \neq \epsilon \wedge h = \mathsf{r}_{\mathsf{W}}fgb] \\ \to hx\epsilon = fx \wedge hxy = gxy(hx(\mathsf{p}_{\mathsf{W}}y)) \mid bxy) \end{aligned}$$

We also need a recursion scheme working along lexicographic ordering analogous to the one used to characterise BRL in Appendix B.

Lemma 2.11 (Bounded lexicographic recursion) *There is a closed term* r_ℓ *in* $\mathcal{L}_\mathsf{W}$ *such that* $\mathsf{B} + (\Sigma^b_\mathsf{W}\text{-}\mathsf{I}_\ell)$ *proves*

$$\begin{aligned} f : \mathsf{W} \mapsto \mathsf{W} \wedge g : \mathsf{W}^3 \mapsto \mathsf{W} \wedge b : \mathsf{W}^2 \mapsto \mathsf{W} \\ \to (\mathsf{r}_\ell f g b : \mathsf{W}^2 \mapsto \mathsf{W} \ \wedge [x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge y \neq \epsilon \wedge h = \mathsf{r}_\ell f g b] \\ \to hx\epsilon = fx \wedge hxy = gxy(hx(\mathsf{p}_\ell y)) \mid bxy) \end{aligned}$$

We can now summarise the main result of Strahm [53] claiming that the four theories indeed have the desired proof-theoretic strength. The previous two lemmas immediately entail the lower bounds for the four theories which results in the following theorem:

Theorem 2.12 *The following proof-theoretic bounds hold:*

1) *The provably total functions of* PT *coincide with* FPTIME.
2) *The provably total functions of* PTLS *coincide with* FPTIMELINSPACE.
3) *The provably total functions of* PS *coincide with* FPSPACE.
4) *The provably total functions of* LS *coincide with* FLINSPACE.

Remark 2.13 When analysing the proof of the above theorem in Strahm [53], it is obvious that the corresponding theory T^- suffices for achieving the desired proof-theoretic strength of each of the theories. The two formula classes differ only in the admittance of the universal quantifier. But the fact that (sub)formulas of the form $\forall x A$ are permitted is not exploited at any point in the proof of the lower bounds. Especially Lemma 2.10 and Lemma 2.11 can be proved with induction over Σ^{b-}_W formulas only.

2.2 Types

As noted above, the focus of Explicit Mathematics usually is on the type structure. After the discussion of the Applicative Theories in the previous section, we are now ready to introduce the second order part.

In Explicit Mathematics, types are extensional and have (explicit) names which are intensional. The names are individuals and generated via uniform operations. The link between names and the types they are referring to is established by the naming relation $\Re$ following the notation of Jäger [31]. Also the element relation $\in$ connects individuals with types, expressing that an individual is a member of a type.

While the first order part deals with the operations, the second order part defines which types exist and how names are generated. Theories of Explicit Mathematics are specified either by a comprehension scheme or by means of a finite axiomatisation. In the first case, the existence of a type X is (informally) assured such that $X = \{x : A[x]\}$ for a formula A of a certain formula class. The names are then constructed uniformly by means of the constants c_e where e is the Gödel number of A. In the latter case, types are generated by means of a finite number of name constructors.

The most prevailing theory, the theory of elementary explicit types EET, is first presented in its original form via a comprehension scheme in Section 2.2.1. As we will later give a finite axiomatisation for our theory PET, we present the equivalent finite version of EET in Section 2.2.2. In this way, the differences will be more apparent. We will also introduce an additional induction principle, namely type induction.

So far, full systems of Explicit Mathematics were rather strong. To be precise, the weakest theories studied are of strength PRA. Feferman [21, 22] introduced a first system of strength PRA to provide a framework for formalising functional programming languages. Krähenbühl [44] introduced a theory called $\Sigma^+\mathsf{ET}$ of the same strength. Since $\Sigma^+\mathsf{ET}$ is closely related to our theories, we present it in Section 2.2.3.

2.2.1 Elementary Explicit Types

In this section, we introduce the theory EET of **E**lementary **E**xplicit **T**ypes following the outline given in Jäger [34]. Unless stated otherwise, results are recapitulated therefrom.

EET is formulated in the language $\mathcal{L}^2_{\mathsf{N}}$ which is obtained by adding to $\mathcal{L}_{\mathsf{N}}$ countably many type variables $X, Y, Z, \ldots$ (possibly with subscripts), two binary relation symbols $\in$ for elementship and $\Re$ for naming, and also new individual constants c_e for all natural numbers e. To improve readability, we will write the relation $\in$ in infix notation. The relation symbol $\in$ is different from the shortcut $t \in \mathsf{N}$, but this will always be clear from the context.

The individual terms of $\mathcal{L}^2_{\mathsf{N}}$ are the same as in $\mathcal{L}_{\mathsf{N}}$, but of course taking the new constants into account. The type terms of $\mathcal{L}^2_{\mathsf{N}}$ are simply the type variables. The atomic formulas of $\mathcal{L}^2_{\mathsf{N}}$ are the ones of $\mathcal{L}_{\mathsf{N}}$ plus expressions of the form $s \in X$, $\Re(s, X)$, and $X = Y$ for s an individual term and X, Y type variables. The formulas of $\mathcal{L}^2_{\mathsf{N}}$ are defined as the closure of the atomic formulas under the usual logical connectives as well as under existential and universal quantification in both sorts, individuals and types.

Before stating the axioms, we need to define two formula classes:

Definition 2.14 (Stratified and elementary formulas)

1) A $\mathcal{L}^2_{\mathsf{N}}$ formula is called a *stratified formula* if the relation symbol $\Re$ does not occur.

2) An *elementary formula* is a formula which is stratified and where no type variables are bound, either. ⊛

As we will mostly refer to types via their name, we introduce the following abbreviations where $\vec{s} = s_1, \ldots, s_n$, $\vec{X} = X_1, \ldots, X_n$:

$$\begin{aligned} \Re(\vec{s}, \vec{X}) &:= \Re(s_1, X_1) \wedge \cdots \wedge \Re(s_n, X_n) \\ \Re(s) &:= \exists X \Re(s, X) \\ \Re(\vec{s}) &:= \Re(s_1) \wedge \cdots \wedge \Re(s_n) \end{aligned}$$

$$s \mathrel{\dot\in} t := \exists X(\Re(t, X) \wedge s \in X)$$
$$(\forall x \in X)A[x] := \forall x(x \in X \to A[x])$$
$$(\exists x \in X)A[x] := \exists x(x \in X \wedge A[x])$$

The theory EET contains the axioms of BON where strictness is extended to respect the new relations:

(D5) $\quad s \in X \to s\downarrow$

(D6) $\quad \Re(s, X) \to s\downarrow$

Furthermore, EET comprises the following two groups of axioms about types. The first group ensures that every type has a name and that types are indeed extensional. The second group formally defines the uniform type generation for elementary formulas.

I. Explicit Representation and Extensionality

(E1) $\quad \exists x \Re(x, X)$

(E2) $\quad \Re(s, X) \wedge \Re(s, Y) \to X = Y$

(E3) $\quad \forall z(z \in X \leftrightarrow z \in Y) \to X = Y$

II. Elementary Comprehension. Let $A[x, \vec{y}, \vec{Z}]$ be an elementary formula with Gödel number e such that $x, \vec{y}, \vec{Z}$ is a conclusive list of the free variables

(ECA1) $\quad \exists X \forall x(x \in X \leftrightarrow A[x, \vec{u}, \vec{Z}])$

(ECA2) $\quad \Re(\vec{v}, \vec{V}) \wedge \forall x(x \in X \leftrightarrow A[x, \vec{u}, \vec{V}]) \to \Re(\mathsf{c}_e(\vec{u}, \vec{v}), X)$

Different induction schemes were already discussed in Section 2.1.2 for Applicative Theories. Also for EET the choice of the induction determines the strength of the system. The two principles introduced over BON, ($\mathsf{S\text{-}I_N}$) and ($\mathsf{F\text{-}I_N}$), can also be added to EET, whereat formula induction is defined for all $\mathcal{L}^2_{\mathsf{N}}$ formulas. In full systems of Explicit Mathematics, there is an additional interesting induction principle, namely type induction:

$$(\mathsf{T\text{-}I_N}) \qquad 0 \in X \wedge (\forall x \in \mathsf{N})(x \in X \to \mathsf{s_N} x \in X) \to (\forall x \in \mathsf{N})(x \in X)$$

When taking comprehension into account, $(\mathsf{T\text{-}I_N})$ is obviously equivalent to restricting $(\mathsf{F\text{-}I_N})$ to elementary formulas.

The following theorem states the proof-theoretic strength of the different induction principles added to EET. In the presence of $(\mathsf{S\text{-}I_N})$, the strength is independent of the type structure. In contrast, formula induction is a stronger principle when additional formulas regarding types are allowed. In fact, $\mathsf{EET{+}(F\text{-}I_N)}$ is proof-theoretically equivalent to the system Π^0_∞-CA of second order arithmetic with arithmetical comprehension. Type induction corresponds to formula induction for elementary formulas and is therefore weaker than full formula induction.

Theorem 2.15 *The following proof-theoretic equivalences were established for* EET *plus induction:*

$$\begin{aligned} \mathsf{EET{+}(S\text{-}I_N)} &\equiv \mathsf{PRA} \\ \mathsf{EET{+}(T\text{-}I_W)} &\equiv \mathsf{PA} \\ \mathsf{EET{+}(F\text{-}I_N)} &\equiv \Pi^0_\infty\text{-CA} \end{aligned}$$

2.2.2 Finite Axiomatisation of EET

Feferman and Jäger [24] introduced a finite axiomatisation of EET. The proof of the equivalence of the two systems is spelled out in detail e.g. in Jansen [39].

The finite axiomatisation is formulated in the language $\mathcal{L}^{2f}_{\mathsf{N}}$ which differs from $\mathcal{L}^{2}_{\mathsf{N}}$ only in the constants available. Instead of the constants c_e, $\mathcal{L}^{2f}_{\mathsf{N}}$ contains new constants used as name generators: nat, id, co, int, inv, and dom.

The axioms of EET^f are the same as those of EET except for the comprehension axioms, (ECA1) and (ECA2). These are replaced by the following axioms:

I. Natural Numbers

(1) $\exists X \forall x(x \in X \leftrightarrow \mathsf{N}(x))$

(2) $\forall x(x \in X \leftrightarrow \mathsf{N}(x)) \rightarrow \Re(\mathsf{nat}, X)$

II. Identity

(3) $\exists X \forall x(x \in X \leftrightarrow \exists y(x = (y, y)))$

(4) $\forall x(x \in X \leftrightarrow \exists y(x = (y, y))) \rightarrow \Re(\mathsf{id}, X)$

III. Complements

(5) $\exists X \forall x(x \in X \leftrightarrow x \notin Y)$

(6) $\Re(y, Y) \wedge \forall x(x \in X \leftrightarrow x \notin Y) \rightarrow \Re(\mathsf{co}y, X)$

IV. Intersection

(7) $\exists X \forall x(x \in X \leftrightarrow x \in Y \wedge x \in Z)$

(8) $\Re(y, Y) \wedge \Re(z, Z) \wedge \forall x(x \in X \leftrightarrow x \in Y \wedge x \in Z) \rightarrow \Re(\mathsf{int}(y, z), X)$

V. Domains

(9) $\exists X \forall x(x \in X \leftrightarrow \exists y((x, y) \in Y))$

(10) $\Re(y, Y) \wedge \forall x(x \in X \leftrightarrow \exists y((x, y) \in Y)) \rightarrow \Re(\mathsf{dom}(y), X)$

VI. Inverse Images

(11) $\exists X \forall x(x \in X \leftrightarrow fx \in Y)$

(12) $\Re(y, Y) \wedge \forall x(x \in X \leftrightarrow fx \in Y) \rightarrow \Re(\mathsf{inv}(f, y), X)$

The equivalence of EET and EET^f mainly follows from the following:

Lemma 2.16 *For any elementary formula $A[x, \vec{y}, \vec{Z}]$ of EET^f with Gödel number e, there is a closed term t_e depending on e such that EET^f proves:*

1) $\exists X \forall x(x \in X \leftrightarrow A[x, \vec{y}, \vec{Z}])$

2) $\Re(\vec{z}, Z) \wedge \forall x(x \in X \leftrightarrow A[x, \vec{y}, \vec{Z}]) \rightarrow \Re(t_e(\vec{y}, \vec{z}), X)$

The other direction, i.e. the embedding of EET^f into EET, is immediate as all formulas stating membership conditions in the axioms above are elementary formulas. Therefore, a closed $\mathcal{L}^2_{\mathsf{N}}$ term can be constructed for every additional constant of $\mathcal{L}^{2f}_{\mathsf{N}}$ by means of the constants c_e.

2.2.3 Weaker Theories with Types

In his Master thesis [44], Krähenbühl introduced the theory $\Sigma^+\mathsf{ET}$ of Positive Existential Comprehension. As this is the system of Explicit Mathematics most closely related to our theories, we shortly summarise the results in this section.

The theory $\Sigma^+\mathsf{ET}$ is formulated in the language $\mathcal{L}^{2f}_{\mathsf{N}}$ and is given by means of a finite axiomatisation in the flavour of EET^f. To be precise, the axioms are those of EET^f except for complement. As disjunction (union) has to be added in the absence of complement, we repeat the axioms here in the more compact formulation of Krähenbühl [44]:

(1) $\Re(\mathsf{nat}) \wedge \forall x(x \mathrel{\dot\in} \mathsf{nat} \leftrightarrow x \in \mathsf{N})$

(2) $\Re(\mathsf{id}) \wedge \forall x(x \mathrel{\dot\in} \mathsf{id} \leftrightarrow \exists y(x = (y, y)))$

(3) $\Re(a) \wedge \Re(b) \rightarrow \Re(\mathsf{un}(a, b)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{un}(a, b) \leftrightarrow (x \mathrel{\dot\in} a \vee x \mathrel{\dot\in} b))$

(4) $\Re(a) \wedge \Re(b) \rightarrow \Re(\mathsf{int}(a, b)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{int}(a, b) \leftrightarrow (x \mathrel{\dot\in} a \wedge x \mathrel{\dot\in} b))$

(5) $\Re(a) \rightarrow \Re(\mathsf{dom}(a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{dom}(a) \leftrightarrow \exists y((x, y) \mathrel{\dot\in} a))$

(6) $\Re(a) \rightarrow \Re(\mathsf{inv}(f, a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{inv}(f, a) \leftrightarrow fx \mathrel{\dot\in} a)$

Krähenbühl also identifies the corresponding formula class $\Sigma^+\mathsf{E}$ for the comprehension scheme. Reflecting the type constructors above, this class corresponds to the class of elementary formulas, except for the lack of negation. Formally, it is defined as follows:

Definition 2.17 (Positive Existential Elementary Formulas) The class of $\Sigma^+\mathsf{E}$ formulas is inductively defined by the following:

1) For all terms s, t and any type variable X, the formulas $s = t$, $s{\downarrow}$, $s \in \mathsf{N}$, and $s \in X$ are $\Sigma^+\mathsf{E}$ formulas.

2) If A, B are $\Sigma^+\mathsf{E}$ formulas, then so are $A \vee B$, $A \wedge B$, and $\exists x A$ for any individual variable x. ⊛

A comprehension comprehension scheme analogous to Lemma 2.16 can be proved in $\Sigma^+\mathsf{ET}$ for $\Sigma^+\mathsf{E}$ formulas.

As $\Sigma^+\mathsf{ET}$ is obviously weaker than EET, only two induction principles are investigated, namely $(\mathsf{T}\text{-}\mathsf{I}_\mathsf{N})$ and $(\mathsf{F}\text{-}\mathsf{I}_\mathsf{N})$. It emerges that $\Sigma^+\mathsf{ET}+(\mathsf{T}\text{-}\mathsf{I}_\mathsf{N})$ and $\Sigma^+\mathsf{ET}+(\mathsf{F}\text{-}\mathsf{I}_\mathsf{N})$ are of the same strength as $\mathsf{EET}+(\mathsf{S}\text{-}\mathsf{I}_\mathsf{N})$ and $\mathsf{EET}+(\mathsf{T}\text{-}\mathsf{I}_\mathsf{N})$ respectively. Formally, the result as established by Krähenbühl [44] is spelled out as follows.

Theorem 2.18 *The following proof-theoretic equivalences were established for* $\Sigma^+\mathsf{ET}$ *plus induction:*

$$\begin{aligned} \Sigma^+\mathsf{ET}+(\mathsf{T}\text{-}\mathsf{I}_\mathsf{N}) &\equiv \mathsf{PRA} \\ \Sigma^+\mathsf{ET}+(\mathsf{F}\text{-}\mathsf{I}_\mathsf{N}) &\equiv \mathsf{PA} \end{aligned}$$

2.3 Semantics of Explicit Mathematics

In this section, we give an overview over the most crucial components of the semantics of Explicit Mathematics. Analogous to the setup of systems of Explicit Mathematics, models consist of two layers. The first part models the underlying Applicative Theory. Interpretations for the second-order elements (types, and naming and element relations) are then constructed on top of this first-order model.

Thus, we first mention some common models for the theory BON without going into details. Then we state a recipe for constructing models for EET from models of BON. Finally, we give a short description of the open term model $\mathcal{M}(\lambda\eta)$ that will play a decisive role later in Chapter 4.

Definition 2.19 ($\mathcal{L}_\mathsf{N}$ and $\mathcal{L}_\mathsf{W}$ structure)
A $\mathcal{L}_\mathsf{N}$ ($\mathcal{L}_\mathsf{W}$) structure $\mathcal{M}$ is given by a tuple

$$\mathcal{M} = (M, N, I, \mathrm{App}, \bot)$$

with the following properties:

(i) The universe M is a non-empty set and usually written $|\mathcal{M}|$.
(ii) N is a non-empty subset of M.
(iii) I is a mapping assigning to each constant c of $\mathcal{L}_{\mathsf{N}}$ ($\mathcal{L}_{\mathsf{W}}$) an element $I(\mathsf{c})$ in M that is also abbreviated $\mathsf{c}^{\mathcal{M}}$.
(iv) App is a partial binary function on M and sometimes written $\cdot^{\mathcal{M}}$.
(v) $\perp$ is an object not belonging to M. ⊛

The notions of variable valuations and validity in $\mathcal{M}$ are defined as usual for predicate logic, where the relation N (W) is interpreted by N and undefinedness is represented by $\perp$. Furthermore, the so-called numeral $\overline{n}$ of a natural number n is defined as the closed standard term representing n. It is constructed by repeatedly applying the successor s_{N} to $\mathsf{0}$.

Depending on the purpose, different models for BON are considered, those presented here are taken from Jäger [34]. Given the design of BON, a very natural model is the so-called recursion-theoretic model $\mathcal{PRF}$. The universe $|\mathcal{PRF}|$ consists of the natural numbers $\mathbb{N}$ and the relation N is also interpreted by $\mathbb{N}$. Therefore, $\mathcal{PRF} \models \forall x \mathsf{N}(x)$. Furthermore, for $e \in \mathbb{N}$, $\{e\}$ stands for the partial recursive function over $\mathbb{N}$ with index e. Application is given as the partial function $\mathbb{N} \times \mathbb{N} \rightsquigarrow \mathbb{N}$ by $e \cdot^{\mathcal{PRF}} n :\simeq \{e\}(n)$. The constants are interpreted by indices of suitable functions. It is easy to see that the such constructed $\mathcal{PRF}$ is a model of $\mathsf{BON} + (\mathsf{F\text{-}I}_{\mathsf{N}})$.

There are two common syntactical models, the (closed) term models $\mathcal{CTT}$ and $\mathcal{CNT}$. Both use a convenient notion of term reduction. They mostly differ in their universe, $|\mathcal{CTT}|$ consists of the equivalence classes of the closed $\mathcal{L}$ terms with respect to reduction whereas $|\mathcal{CNT}|$ is composed of the closed $\mathcal{L}$ terms in normal form. Constants are interpreted by themselves and the natural number by the numerals (respectively the equivalence classes thereof). Application in $\mathcal{CTT}$ is given by the equivalence class of the resulting term, whereas in $\mathcal{CNT}$ application is interpreted by taking the strong normal form of the resulting term. Both

structures are models of $\mathsf{BON}+(\mathsf{F}\text{-}\mathsf{I}_\mathsf{N})$. Moreover, $\mathcal{CTT}$ also validates (Tot) in contrast to $\mathcal{CNT}$.

Of course, there are many more models for Applicative Theories, however it would go beyond the scope of this thesis to sketch them.

We now turn to models for full systems of Explicit Mathematics and first define the notion of a structure.

Definition 2.20 ($\mathcal{L}^2_\mathsf{N}$ structure)
A $\mathcal{L}^2_\mathsf{N}$ structure $\mathcal{N}$ is given as a tuple

$$\mathcal{N} = (\mathcal{M}, \mathcal{T}, \mathcal{E}, \mathcal{R}, (m_e : e \in \mathbb{N}))$$

such that the following conditions are satisfied.

(i) $\mathcal{M}$ is a $\mathcal{L}_\mathsf{N}$ structure,
(ii) $\mathcal{T}$ is a non-empty set of subsets of $|\mathcal{M}|$,
(iii) $\mathcal{E}$ and $\mathcal{R}$ are non-empty subsets of $|\mathcal{M}| \times \mathcal{T}$,
(iv) $(m_e : e \in \mathbb{N})$ is a family of elements of $|\mathcal{M}|$. ⊛

Variable valuations and validity are extended as usual to the language $\mathcal{L}^2_\mathsf{N}$. We now give a rough description for constructing a model of EET from a model of BON as specified before.

1. Take any model $\mathcal{M}$ of BON.
2. Define interpretations for constants $\mathsf{c}_e^\mathcal{N} := (\lambda x.(\mathsf{0}, \overline{e}, x))^\mathcal{M}$.
3. $\mathcal{E}$ is the usual $\in$ relation on $|\mathcal{M}| \times \mathcal{T}$.
4. Inductively define sets of names $R_k^\mathcal{N} \subseteq |\mathcal{M}|$ for every $k \in \mathbb{N}$. In parallel, define the extension $ext(m)$ for each $m \in R_k^\mathcal{N}$. Interpretations at level k are defined as $\mathcal{T}_k := \{ext(m) : m \in R_k^\mathcal{N}\}$ and the naming relation is the mapping of the names to their extension $\mathcal{R}_k := \{(m, n) : m \in R_k^\mathcal{N}, n \in ext(m)\}$. Let $\mathcal{N}_k = (\mathcal{M}, \mathcal{T}_k, \mathcal{R}_k, (\mathsf{c}_e^\mathcal{N}))$
   - For the base case, if $A[x, \vec{y}]$ is an elementary formula without

type variables and e its Gödel number, we add $\mathsf{c}_e{}^{\mathcal{N}}(\vec{n})$ to $R_0^{\mathcal{N}}$. $ext(\mathsf{c}_e{}^{\mathcal{N}}(\vec{n})) = \{m \in |\mathcal{M}| : \mathcal{M} \models A[m, \vec{n}]\}$

– For $k > 0$, add $\mathsf{c}_e^{\mathcal{N}}(\vec{n}, \vec{q})$ to $R_k^{\mathcal{N}}$ such that e is the code of an elementary formula $A[x, \vec{y}, \vec{X}]$ and $\vec{q} \in R_{k-1}^{\mathcal{N}}$. $ext(\mathsf{c}_e{}^{\mathcal{N}}(\vec{n}, \vec{q})) = \{m \in |\mathcal{M}| : \mathcal{M} \models A[m, \vec{n}, \vec{q}]\}$

5. The types are now interpreted by $\mathcal{T} := \bigcup_{k\in\mathbb{N}} \mathcal{T}_k$ and the naming relation by $\mathcal{R} := \bigcup_{k\in\mathbb{N}} \mathcal{R}_k$.

Any structure constructed according to this recipe from a model of BON is indeed a model of EET. Furthermore, if $\mathcal{M}$ models (F-I_N) in $\mathcal{L}_\mathsf{N}$, then $\mathcal{N}$ validates (T-I_N) for $\mathcal{L}_\mathsf{N}^2$.

Later in this thesis, we will make use of the open term model $\mathcal{M}(\lambda\eta)$ and thus give a short recapitulation here. In contrast to $\mathcal{CT}$ and $\mathcal{CNT}$, the universe $|\mathcal{M}(\lambda\eta)|$ consists of open terms instead of closed terms. To be precise, it contains the terms of ordinary λ calculus extended by constants according to those of $\mathcal{L}_\mathsf{W}$ or $\mathcal{L}_\mathsf{N}$. Equality is defined by common $\beta\eta$ reduction, in other words, two terms are equal if they have common reduct under $\beta\eta$ reduction. The constants are interpreted by themselves. For more details on the interpretation of the combinatory algebra part refer e.g. to Hindley and Seldin [30]. Additional reduction rules are added to the usual $\beta\eta$ ones to deal with the other basic operations of BON or PT, as was described in a similar setting by Cantini [7, §4A]. Actually, $\mathcal{M}(\lambda\eta)$ models (Tot) and therefore validates the applicative theory based on ordinary first order predicate calculus.

2.4 Extensions

Since the introduction of Explicit Mathematics in the early seventies, several extensions have been discussed. In this section, we recapitulate some of them which are relevant later in this thesis and mention very quickly further extensions.

Two common extensions are totality and extensionality for operations which were already introduced in Section 2.1.3. They mainly affect the first-order part and are therefore independent of the type structure.

In the original formulation of Feferman [16], an additional type constructor was present, the so-called join operator which constructs the disjoint union of a family of types. The join principle was studied over several theories and turned out to be an expressively powerful schema. Formally, join is formulated by means of an additional constant j as follows:

$$(\mathsf{J}) \qquad \Re(a,X)\wedge(\forall x\in X)\exists Y\Re(fx,Y)\rightarrow\exists Z(\Re(\mathsf{j}(a,f),Z)\wedge\Sigma[X,f,Z])$$

where $\Sigma[X,f,Z]$ is an abbreviation for the following formula:

$$\forall x(x\in Z\leftrightarrow x=((x)_0,(x)_1)\wedge(x)_0\in X\wedge\exists Y(\Re(f(x)_0,Y)\wedge(x)_1\in Y)$$

The join principle has been studied over several systems of explicit mathematics, among them EET plus induction and $\Sigma^+\mathsf{ET}$ plus induction. It is an interesting principle with nice applications. Indeed, Studer [58] used $\mathsf{EET}+\mathsf{J}$ as the basis for formalising Featherweight Java, a limited version of the widespread programming language Java.

Despite its expressive power, join does in most cases not increase the proof-theoretic strength of the studied system. For example, Krähenbühl [44] proved that the addition of join does not increase the strength of $\Sigma^+\mathsf{ET}$ with neither $(\mathsf{T}\text{-}\mathsf{I}_\mathsf{N})$ nor $(\mathsf{F}\text{-}\mathsf{I}_\mathsf{N})$. However, for stronger systems join makes a difference in strength due to Feferman [16, 18]. While the proof-theoretic strength of $\mathsf{EET}+\mathsf{J}+(\mathsf{T}\text{-}\mathsf{I}_\mathsf{N})$ is still PA, the join principle increases the strength in the presence of $(\mathsf{F}\text{-}\mathsf{I}_\mathsf{N})$, resulting in the system $\Pi^0_\infty\text{-}\mathsf{CA}_{<\varepsilon_0}$.

Further type constructor schemes were studied such as different forms of power types among others by Cantini and Minari [10], Jäger [32] and Krähenbühl [44]. Also several ontological principles as for example "everything is a name" $(\forall\Re)$ were investigated. However, caution is advised

when combining those, as some combinations result in inconsistent systems. E.g. $\mathsf{EET}+\mathsf{J}+\forall\Re$ is inconsistent while both $\mathsf{EET}+\mathsf{J}$ and $\mathsf{EET}+\forall\Re$ are consistent.

Moreover, comprehension for other formula classes was investigated, e.g. for stratified formulas. However, full comprehension (i.e. comprehension for all formulas) leads to the Russell paradox and is therefore again inconsistent. For more details, refer to the corresponding literature, especially to the overviews given in Jäger [34], Jansen [39]

2.5 Related Work

In this section, we shortly discuss related fields of research. For more information about the systems mentioned, we refer to the respective literature.

In the context of Explicit Mathematics, further systems have been studied that are much stronger than the systems described above. We can only mention a few of them. The principle of inductive generations was part of Feferman's original system T_0 [16]. Jäger and Studer [36] discussed it in an extension of T_0 by the Limit axiom and the Mahlo axiom. Also the widespread concept of universes was investigated among others by Feferman [19], Jäger, Kahle and Studer [38], and Strahm [52]. Furthermore, Probst [49] studied pseudo-hierarchies in the setting of Explicit Mathematics.

Feferman [21, 22] also studied weaker systems where he establishes a framework to formalise functional programming languages. The weakest systems presented are of strength PRA.

Worth mentioning are also the systems of Operational Set Theory as suggested by Feferman [15] and further researched by Jäger [33]. Another theory relating sets and operations was presented by Beeson [1].

In thesis, the focus is on weak systems, i.e. systems where the provably total functions are at most the functions computable in polynomial

space. In this context, familiar systems are the systems of bounded arithmetic as discussed by Buss [3] or Hájek and Pudlák [28]. They deal with weak complexity classes by bounding the quantifiers. To the same field also belongs the work of Krajíček [45] and Ferreira [26, 27]

A different approach to weak complexity classes are tiered systems. In this context, the restriction is not by bounding quantifiers but by distinguishing different kinds of variables and controlling their usage employing so-called safe induction. The class of polynomial time computable functions in this setting was studied by Bellantoni and Cook [2] and Leivant [46]. In the setting of Applicative Theories, Cantini studied systems with different types of variables in [9, 5]. A similar concept in a different setting was investigated by Ostrin and Wainer [48].

Systems dealing with truth are an interesting field in logic. Different theories about truth were also studied in the setting of Explicit Mathematics and especially Applicative Theories. Cantini [7] gives a thorough overview of truth theories in this setting. Kahle [42, 40] studied several versions based on the theory BON. Furthermore, Cantini [8] considered weak truth theories (i.e. corresponding to the polynomial time computable functions) in the setting of Applicative Theories.

In connection with weak complexity classes, higher type functionals are also of importance. A reasonable notion of feasibly computable functions is more complex in this context than for functions $\mathbb{N} \to \mathbb{N}$. A prominent proposal to capture this meaning are the Basic Feasible Functionals introduced by Melhorn [47]. Further research in this direction was conducted by Cook and Urquhart [14]. In the setting of Applicative Theories, Strahm [53, 54] embeds the system of Cook and Urquhart in PT and also relates his weak theories to the basic feasible functionals.

Chapter 3

Weak Axiom Systems including Types

In this chapter, we will discuss systems of Explicit Mathematics corresponding to weak complexity classes. Among the weakest full systems studied in the same setting was the system $\Sigma^+\mathsf{ET}$ corresponding to PRA in the presence of type induction as recapitulated in Section 2.2.3. Weaker theories were only studied for the applicative framework as summarised in Section 2.1.4. Our goal is to work out theories corresponding to feasible complexity classes, i.e. complexity classes interesting for computer science.

To be precise, we are interested in theories characterising the complexity classes FPTIME, FPTIMELINSPACE, FPSPACE, and FLINSPACE (see Appendix B). In other words, we seek second order counterparts of Strahm's applicative theories PT, PTLS, PS, and LS as introduced in [53] (cf. Section 2.1.4).

For this purpose, we start off from the weak applicative theories replicated in the last chapter. We proceed by adding type existence axioms and replacing induction for Σ^b_{W} formulas by type induction. Our type existence axioms are very naturally presented by means of a finite

axiomatisation in the spirit of EET^f (cf. Section 2.2.2). They correspond to a natural restriction of elementary comprehension presented in Section 2.2.1. In the process, our main focus will be on the theory PET whose provably total functions are the polynomial time computable ones.

Cantini [6] studies several extensions for weak applicative theories not increasing the proof-theoretic strength. We will make use of those and add some to our theory PET. Thereby, especially the uniformity principle is interesting as it gives rise to a nice type constructor making the type generators more symmetric.

In the course of specifying weak systems of full Explicit Mathematics, we will first introduce the theory PET for FPTIME in Section 3.1 by giving a finite axiomatisation as well as a corresponding comprehension scheme. Thereafter, we establish the desired upper and lower bounds of PET in Section 3.2 and Section 3.3, respectively. The proof of the upper bounds employs a model theoretic argument, but later, in Section 4.5, we will also point out a syntactical proof of the same result. In Section 3.4, we discuss the above mentioned extensions based on Cantini [6]. Finally, Section 3.5 states theories whose respective proof-theoretic strength corresponds to the remaining three complexity classes.

Part of the results presented in this chapter were already published in Spescha and Strahm [51] and we make free use of this paper.

3.1 The Theory PET

In this section, we introduce PET, the theory of **p**olynomial time operations with **e**xplicit **t**ypes, by a finite axiomatisation. Later we present a comprehension scheme corresponding to the given type constructors.

PET is formulated in the second order language $\mathcal{L}^2_{\mathsf{W}}$ which extends the language $\mathcal{L}_{\mathsf{W}}$ of PT by type variables $U, V, W, X, Y, Z, \ldots$, binary relation symbols $\Re$ (naming) and $\in$ (elementhood), as well as (individual) constants w (initial segment of W), id (identity), dom (domain), un (union),

int (intersection), and inv (inverse image). The relation symbol $\in$ differs from the symbol used in the shortcut $a \in \mathsf{W}$ introduced in 2.1.4, but it will always be clear from the context.

The *individual terms* r, s, t of $\mathcal{L}^2_{\mathsf{W}}$ are those of $\mathcal{L}_{\mathsf{W}}$, but taking into account the new constants, whereas the *type terms* consist of the type variables only. The *formulas* $A, B, C, \ldots$ of $\mathcal{L}^2_{\mathsf{W}}$ (possibly with subscripts) are built from the atomic formulas of $\mathcal{L}_{\mathsf{W}}$ (see Appendix A) as well as formulas of the form $(s \in X)$, $\Re(s, X)$ and $(X = Y)$, by closing under negation, disjunction, conjunction, implication, as well as existential and universal quantification over both individuals and types. If A is an $\mathcal{L}^2_{\mathsf{W}}$ formula, we let $\mathrm{FV}_I(A)$ and $\mathrm{FV}_T(A)$ denote the set of its free individual and type variables, respectively. Finally, we write $\mathrm{FV}_I(t)$ for the set of individual variables occurring in the term t.

Definition 3.1 (Free Variables $\mathbf{FV_I(A)}$ and $\mathbf{FV_T(A)}$)
For an individual term t, the set of individual variables $\mathrm{FV}_I(t)$ is defined by induction on the build-up:

$$
\begin{aligned}
t \equiv x \text{ (individual variable)} &\quad\Longrightarrow\quad \mathrm{FV}_I(t) = \{x\} \\
t \equiv \mathsf{c} \text{ (constant)} &\quad\Longrightarrow\quad \mathrm{FV}_I(t) = \emptyset \\
t \equiv t_0 \cdot t_1 &\quad\Longrightarrow\quad \mathrm{FV}_I(t) = \mathrm{FV}_I(t_0) \cup \mathrm{FV}_I(t_1)
\end{aligned}
$$

The sets $\mathrm{FV}_I(A)$ and $\mathrm{FV}_T(A)$ of free individual and type variables respectively are defined by induction on the construction of the formula A:

$$
\begin{aligned}
A \equiv t{\downarrow} \mid \mathsf{W}(t) &\quad\Longrightarrow\quad \mathrm{FV}_I(A) = \mathrm{FV}_I(t) \\
&\qquad\qquad\ \ \mathrm{FV}_T(A) = \emptyset \\
A \equiv t_0 = t_1 &\quad\Longrightarrow\quad \mathrm{FV}_I(A) = \mathrm{FV}_I(t_0) \cup \mathrm{FV}_I(t_1) \\
&\qquad\qquad\ \ \mathrm{FV}_T(A) = \emptyset \\
A \equiv t \in X \mid \Re(t, X) &\quad\Longrightarrow\quad \mathrm{FV}_I(A) = \mathrm{FV}_I(t) \\
&\qquad\qquad\ \ \mathrm{FV}_T(A) = \{X\} \\
A \equiv X = Y &\quad\Longrightarrow\quad \mathrm{FV}_I(A) = \emptyset \\
&\qquad\qquad\ \ \mathrm{FV}_T(A) = \{X, Y\}
\end{aligned}
$$

$$
\begin{array}{lcl}
A \equiv \neg B & \Longrightarrow & \mathrm{FV}_I(A) = \mathrm{FV}_I(B) \\
 & & \mathrm{FV}_T(A) = \mathrm{FV}_T(B) \\
A \equiv B \odot C & \Longrightarrow & \mathrm{FV}_I(A) = \mathrm{FV}_I(B) \cup \mathrm{FV}_I(C) \\
\quad \odot \in \{\wedge, \vee, \rightarrow\} & & \mathrm{FV}_T(A) = \mathrm{FV}_T(B) \cup \mathrm{FV}_T(C) \\
A \equiv \forall x B \mid \exists x B & \Longrightarrow & \mathrm{FV}_I(A) = \mathrm{FV}_I(B) \setminus \{x\} \\
 & & \mathrm{FV}_T(A) = \mathrm{FV}_T(B) \\
A \equiv \forall X B \mid \exists X B & \Longrightarrow & \mathrm{FV}_I(A) = \mathrm{FV}_I(B) \\
 & & \mathrm{FV}_T(A) = \mathrm{FV}_T(B) \setminus \{X\} \qquad \circledast
\end{array}
$$

As before, types are extensional and their names can be seen as intensional. Since we mostly refer to types by using their names, we use the same abbreviations like $\Re(s)$ and $t \dot{\in} s$ as defined in Section 2.2.1. Furthermore, we let $\mathsf{W}_a(x)$ abbreviate $(\mathsf{W}(x) \wedge x \leq a)$. We are now ready to spell out the axioms of PET in detail.

3.1.1 Axioms of PET

The logic of our system is the usual predicate logic with equality for both sorts as before. The logical axioms of PET are analogous to those of EET (see Section 2.2.1).

PET consists of the axioms of BOW as listed in Section 2.1.4 plus the following axiom groups about types. The axioms in group I. are the so-called ontological axioms about the naming relation and extensionality. They are identical to those of EET and stated here for completeness. In group II. we state the axioms about type existence and finally, we include the induction schema in group III.

I. Explicit Representation and Extensionality

(O.1) $\exists x \Re(x, X)$

(O.2) $\Re(a, X) \wedge \Re(a, Y) \rightarrow X = Y$

(O.3) $\forall z(z \in X \leftrightarrow z \in Y) \rightarrow X = Y$

II. Type Existence Axioms

(w_a) $a \in \mathsf{W} \to \Re(\mathsf{w}(a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{w}(a) \leftrightarrow \mathsf{W}_a(x))$

(id) $\Re(\mathsf{id}) \wedge \forall x(x \mathrel{\dot\in} \mathsf{id} \leftrightarrow \exists y(x = (y, y)))$

(un) $\Re(a) \wedge \Re(b) \to \Re(\mathsf{un}(a,b)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{un}(a,b) \leftrightarrow (x \mathrel{\dot\in} a \vee x \mathrel{\dot\in} b))$

(int) $\Re(a) \wedge \Re(b) \to \Re(\mathsf{int}(a,b)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{int}(a,b) \leftrightarrow (x \mathrel{\dot\in} a \wedge x \mathrel{\dot\in} b))$

(dom) $\Re(a) \to \Re(\mathsf{dom}(a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{dom}(a) \leftrightarrow \exists y((x, y) \mathrel{\dot\in} a))$

(inv) $\Re(a) \to \Re(\mathsf{inv}(f, a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{inv}(f, a) \leftrightarrow fx \mathrel{\dot\in} a)$

III. Type Induction on W

(T-I_W) $\epsilon \in X \wedge (\forall x \in \mathsf{W})(\mathsf{p}_\mathsf{W} x \in X \to x \in X) \to (\forall x \in \mathsf{W})(x \in X)$

Please note that the induction step in (T-I_W) requires that both $\mathsf{s}_0 x \in X$ and $\mathsf{s}_1 x \in X$ if $x \in X$.

The type existence axioms as presented here are a natural restriction of the axioms of EET^f (cf. Section 2.2.2). We omit complement and replace the type of natural numbers by initial segments of binary words. Besides, we have to add the existence of the union of two types in PET as this follows from intersection and complement for EET^f. The main difference to $\Sigma^+\mathsf{ET}$ (cf. Section 2.2.3) is the treatment of the base type of binary words. By letting the binary words as a whole be a type, we would immediately result in a theory of strength PRA again.

3.1.2 Restricted Elementary Ccomprehension

Most systems of Explicit Mathematics state type existence in the form of a comprehension scheme for a specified formula class. In this spirit, we show that the finite axiomatisation of PET gives rise to a natural restriction of the schema of elementary comprehension specified in Section 2.2.1. In the context of this much weaker theory, the formula class used to define types is rather intricate.

Comprehension is available for the class of so-called $\Sigma^\mathsf{b}_\mathrm{T}$ formulas. As the name already suggests, they are not only a subset of the elementary

formulas, but also closely related to the class of Σ^b_{W} formulas used for the induction scheme in Section 2.1.4. The relation symbol W is only allowed for bounded terms t in the form of $\mathsf{W}_a(t)$. As we want to be able to generate a formula defining the element condition for every type generated via the type constructors above, we need to allow several nested bounds. But in return, bounds have to be given as a (fixed) term instead of the more liberal function term in $(\exists y \leq fx)B[f, x, y]$ of Σ^b_{W} formulas. Furthermore, we must ensure that bounds of subformulas do not interfere with each other in composed formulas.

To ensure this condition, in addition to the notion of $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formulas, we have to define a set of designated free individual variables $\mathrm{FV}_{\mathsf{W}}(A)$ which shall be thought of as the binary words bounding existential quantifiers in a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula A. These variables act as parameters in the comprehension schema below.

Definition 3.2 ($\Sigma^{\mathsf{b}}_{\mathrm{T}}$ Formulas)
The class of $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formulas of $\mathcal{L}^2_{\mathsf{W}}$ and the set of variables $\mathrm{FV}_{\mathsf{W}}(A)$ are inductively defined as follows:

1) If A is an $\mathcal{L}^2_{\mathsf{W}}$ formula of the form $(s = t)$, $s{\downarrow}$ or $(s \in X)$, then A is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula and $\mathrm{FV}_{\mathsf{W}}(A) := \emptyset$.

2) If A is the formula $\mathsf{W}_a(t)$ with $a \notin \mathrm{FV}_I(t)$, then A is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula and $\mathrm{FV}_{\mathsf{W}}(A) := \{a\}$.

3) If A is the formula $(B \wedge C)$ or $(B \vee C)$ with B and C in $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ and, in addition,

$$(\mathrm{FV}_I(B) \setminus \mathrm{FV}_{\mathsf{W}}(B)) \cap \mathrm{FV}_{\mathsf{W}}(C) = \emptyset,$$
$$(\mathrm{FV}_I(C) \setminus \mathrm{FV}_{\mathsf{W}}(C)) \cap \mathrm{FV}_{\mathsf{W}}(B) = \emptyset,$$

then A is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula and $\mathrm{FV}_{\mathsf{W}}(A) := \mathrm{FV}_{\mathsf{W}}(B) \cup \mathrm{FV}_{\mathsf{W}}(C)$.

4) If A is the formula $\exists x B$ with $B \in \Sigma^{\mathsf{b}}_{\mathrm{T}}$ and $x \notin \mathrm{FV}_{\mathsf{W}}(B)$, then A is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula and $\mathrm{FV}_{\mathsf{W}}(A) := \mathrm{FV}_{\mathsf{W}}(B)$. ⊛

Remark 3.3 We observe that the above definition also captures formulas starting with a bounded (with respect to W) existential quantifier; namely, if $B[x]$ is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula, then $(\exists x \leq a)B[x]$ can be expressed by the $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula $\exists x(\mathsf{W}_a(x) \wedge B[x])$.

In the sequel we assume that for each $\mathcal{L}^2_{\mathsf{W}}$ formula A, we have a mapping μ_A which assigns to each free type variable X in A a fresh individual variable $\mu_A(X)$ that does not occur in A. We assume that μ_A is injective. The elegant notation in the following definition is adapted from Krähenbühl [44].

Definition 3.4 (Naming Term $\rho_A x.B$)
Assume that A is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula. Then we define a term $\rho_A x.B$ by induction on the complexity of the formula B in $\Sigma^{\mathsf{b}}_{\mathrm{T}}$, where we assume that $x \notin \mathrm{FV}_{\mathsf{W}}(B)$ and x not bound in B:

$$\begin{aligned}
\rho_A x.(s = t) &:= \mathsf{inv}(\lambda x.(s,t), \mathsf{id}),\\
\rho_A x.(s{\downarrow}) &:= \mathsf{inv}(\lambda x.(s,s), \mathsf{id}),\\
\rho_A x.(s \in \mathsf{W}_a) &:= \mathsf{inv}(\lambda x.s, \mathsf{w}(a)),\\
\rho_A x.(s \in X) &:= \mathsf{inv}(\lambda x.s, \mu_A(X)),\\
\rho_A x.(C \wedge D) &:= \mathsf{int}(\rho_A x.C, \rho_A x.D)\\
\rho_A x.(C \vee D) &:= \mathsf{un}(\rho_A x.C, \rho_A x.D)\\
\rho_A x.(\exists y C) &:= \mathsf{dom}(\rho_A x.(C[(x)_0/x, (x)_1/y])).
\end{aligned}$$ ⊛

We write $\rho x.A$ instead of $\rho_A x.A$. The following theorem states that we can derive uniform comprehension for $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formulas in PET.

Theorem 3.5 (Restricted elementary comprehension in PET)
Assume that A is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula with $\mathrm{FV}_T(A) = \{X_1, \ldots, X_n\}$ and $\mathrm{FV}_{\mathsf{W}}(A) = \{w_1, \ldots, w_m\}$. If we let $z_i := \mu_A(X_i)$ for $1 \leq i \leq n$, then we have:

1) $\mathrm{FV}_I(\rho x.A) = (\mathrm{FV}_I(A) \setminus \{x\}) \cup \{z_1, \ldots, z_n\}$,

2) $\mathsf{PET} \vdash \mathsf{W}(\vec{w}) \wedge \Re(\vec{z}, \vec{X}) \rightarrow \Re(\rho x.A)$,

3) $\mathsf{PET} \vdash \mathsf{W}(\vec{w}) \wedge \Re(\vec{z}, \vec{X}) \rightarrow (\forall x)(x \mathrel{\dot\in} \rho x.A \leftrightarrow A)$.

Proof The theorem is immediately derivable from the definition of the name term $\rho x.A$ for $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formulas. □

Using λ abstraction and projections, we obtain the following immediate consequence of the above theorem.

Corollary 3.6 *Assume that* $A[x, \vec{v}, \vec{w}, \vec{X}]$ *is a* $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ *formula with the following free variables:*

$$\begin{aligned} \mathrm{FV}_T(A) &= \{X_1, \ldots, X_n\}, \\ \mathrm{FV}_{\mathsf{W}}(A) &= \{w_1, \ldots, w_m\}, \\ \mathrm{FV}_I(A) \setminus \mathrm{FV}_{\mathsf{W}}(A) &= \{x, v_1, \ldots, v_k\}. \end{aligned}$$

Then we can find a closed $\mathcal{L}^2_{\mathsf{W}}$ *term* c_A *such that* PET *proves:*

1) $\mathsf{W}(\vec{w}) \wedge \Re(\vec{z}, \vec{X}) \rightarrow \Re(c_A(\vec{v}, \vec{w}, \vec{z}))$,

2) $\mathsf{W}(\vec{w}) \wedge \Re(\vec{z}, \vec{X}) \rightarrow (\forall x)(x \mathrel{\dot\in} c_A(\vec{v}, \vec{w}, \vec{z}) \leftrightarrow A[x, \vec{v}, \vec{w}, \vec{X}])$.

It is easy to see that the opposite direction of the above theorem also holds. More precisely, the schema of uniform $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ comprehension clearly entails the type existence axioms as given in the finite axiomatisation of PET. We only sketch the proof of this claim here. Let PET^C be the axiomatisation of PET where type existence is stated in the form of the comprehension axioms analogous to (ECA1) and (ECA2) for EET, but for $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formulas. Then we can prove the existence of closed terms $t_{\mathsf{w}}, t_{\mathsf{id}}, t_{\mathsf{dom}}, t_{\mathsf{inv}}, t_{\mathsf{un}}$, and t_{int} in PET^C such that these terms, informally speaking, fulfil the type constructors given in the finite axiomatisation. For example, take the formula $A \equiv (\exists y \leq a)(x = y)$ and let e be its Gödel number. Then we can choose $t_{\mathsf{w}} := (\lambda a.\mathsf{c}_e(a))$. The remaining terms are analogous and equivalent to the proof for EET and EET^f spelled out in detail e.g. in Jansen [39].

3.2 Lower Bounds

To establish the lower bounds of our theory, we embed PT^- as introduced in Section 2.1.4 into PET in this section. As mentioned before, PT^- is expressively weaker than PT, but the provably total functions are still the polynomial time computable functions (cf. Remark 2.13). Recall that PET is based on the theory BOW where some (derivable) axioms are omitted for the sake of minimality, among them Axiom (19) (on page 17). Before we can specify the embedding, we have to prove some auxiliary lemmas. Mainly, we reformulate type induction as bounded induction scheme and we need to prove that every function on the binary words can be bounded by a monotone function.

Theorem 3.5 establishes that any $\Sigma^{\mathsf{b}}_{\mathrm{T}}$-formula defines a type in PET. With Axiom (T-I_{W}) we have induction on any type which is equivalent to induction for $\Sigma^{\mathsf{b}}_{\mathrm{T}}$-formulas. In the following, we will heavily use this fact. For convenience we will also exploit the obvious equivalence

$$(\forall x \in \mathsf{W})(x \in X \to \mathsf{s}_0 x \in X \wedge \mathsf{s}_1 x \in X) \leftrightarrow (\forall x \in \mathsf{W})(\mathsf{p}_{\mathsf{W}} x \in X \to x \in X)$$

First, we prove the properties of the subword relation that are not stated as an axiom in our present setting:

Lemma 3.7 *The following statements are provable in* PET*:*

1) $x \in \mathsf{W} \wedge z \in \mathsf{W} \wedge x \subseteq \mathsf{p}_{\mathsf{W}} z \to x \subseteq z$,
2) $x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge z \in \mathsf{W} \wedge x \subseteq y \wedge y \subseteq z \to x \subseteq z$ *(Transitivity)*,
3) $x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x \subseteq y \to x \leq y$,
4) $x \in \mathsf{W} \to \epsilon \subseteq x$,
5) $x \in \mathsf{W} \wedge y \in \mathsf{W} \to x \leq y \vee y \leq x$.

Proof In the following, we work informally in PET and assume that $x, y, z \in \mathsf{W}$.

1) Immediate with Axiom (18).

2) The Σ^b_T-formula $\mathsf{c}_\subseteq xy = 1 \vee \mathsf{c}_\subseteq yz = 1 \vee \mathsf{c}_\subseteq xz = 0$ is a reformulation of transitivity. We will prove it by induction on z and assume $\mathsf{c}_\subseteq xy = 0$ and $\mathsf{c}_\subseteq yz = 0$.

   $z = \epsilon$: With Axiom (17) we know that also $y = \epsilon$ and thus $x = \epsilon$. With the same axiom we immediately get $x \subseteq z$.

   We assume that transitivity holds for $\mathsf{p}_\mathsf{W} z$, now we will prove that it also holds for z. With Axiom (18) we know that $v \subseteq w$ iff either $v = w$ or $v \subseteq \mathsf{p}_\mathsf{W} w$ for any words v, w. If $x = y$ or $y = z$, $x \subseteq z$ is immediate from the equality axioms. Otherwise we have $x \subseteq \mathsf{p}_\mathsf{W} y\,(i)$ and $y \subseteq \mathsf{p}_\mathsf{W} z\,(ii)$. From (i) and part 1) of this lemma we get $x \subseteq y$. Therefore, by induction hypothesis and (ii), also $x \subseteq \mathsf{p}_\mathsf{W} z$. We get $x \subseteq z$ again with part 1).

3) We can write this property as $\mathsf{c}_\subseteq xy = 1 \vee \mathsf{c}_\subseteq (1 \times x)(1 \times y) = 0$ which obviously is a Σ^b_T formula. Again we will only look at the case $x \subseteq y$ in the induction on y.

   $y = \epsilon$: As above, this implies that also $x = \epsilon$. Therefore obviously $1 \times x = 1 \times y$.

   Assume the assertion holds for $\mathsf{p}_\mathsf{W} y$. To prove that $x \leq y$ if $x \subseteq y$ we make the same case distinction as above: if $x = y$ then $x \leq y$ is again obvious. Otherwise $x \subseteq \mathsf{p}_\mathsf{W} y$. By induction hypothesis we have $1 \times x \subseteq 1 \times (\mathsf{p}_\mathsf{W} y)\,(i)$. Further, $1 \times y = (1 \times \mathsf{p}_\mathsf{W} y) * 1 = \mathsf{s}_1(1 \times \mathsf{p}_\mathsf{W} y)\,(ii)$. With part 1) we get $1 \times x \subseteq 1 \times y$ from (i) and (ii).

4) We prove this by induction on x. If $x = \epsilon$, then $\epsilon \subseteq \epsilon$ immediate with axiom (17). Assume $\epsilon \subseteq x$. With axiom (18) and induction hypothesis, $\epsilon \subseteq \mathsf{s}_i x$.

5) We prove $A[y] \equiv \mathsf{l}_\mathsf{W} x \subseteq \mathsf{l}_\mathsf{W} y \vee \mathsf{l}_\mathsf{W} y \subseteq \mathsf{l}_\mathsf{W} x$ by induction on y.

   $y = \epsilon$: $\mathsf{l}_\mathsf{W} \epsilon = 1 \times \epsilon = \epsilon$. Since $x \in \mathsf{W}$ also $\mathsf{l}_\mathsf{W} x \in \mathsf{W}$ by totality of multiplication. Thus $\epsilon \leq x$ with part 4).

Before we prove the induction step, we need an auxiliary property, namely $y \leq x \to (\exists z \subseteq x)(\mathsf{l_W} z = \mathsf{l_W} y)$(i). This is equivalent to $B[y] \equiv \mathsf{c}_{\subseteq}(\mathsf{l_W} y)(\mathsf{l_W} x) = 1 \vee (\exists z \subseteq x)(\mathsf{l_W} z = \mathsf{l_W} y)$, which is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula for any fixed $x \in \mathsf{W}$. We prove this by induction on y. If $y = \epsilon$. Then choose $z = \epsilon$ and we are done with part 4). Assume $B[y]$ holds. If $\mathsf{s}_i y \not\leq x$, $B[\mathsf{s}_i y]$ holds. Otherwise (i.e. $\mathsf{s}_i y \leq x$), we have either $\mathsf{l_W}(\mathsf{s}_i y) = \mathsf{l_W} x$ and we choose $z = x$. Otherwise, we have $(\exists z' \leq x)(\mathsf{l_W} z' = \mathsf{l_W} y)$ by transitivity and induction hypothesis. Thus $\mathsf{l_W} z' = \mathsf{l_W} y \subseteq \mathsf{p_W l_W} x$. Therefore, either we choose $z = \mathsf{s}_0 z' \subseteq x$ or $z = \mathsf{s}_1 z' \subseteq x$. Then $\mathsf{l_W} z = \mathsf{s}_1(\mathsf{l_W} z') = \mathsf{s}_1(\mathsf{l_W} y) = \mathsf{l_W}(\mathsf{s}_i y)$.

Assume $A[y]$ holds. Since $\mathsf{l_W} x, \mathsf{l_W} y \in \mathsf{W}$, $\mathsf{l_W} x \subseteq \mathsf{l_W} y$ and $\mathsf{l_W} y \subseteq \mathsf{l_W} x$ are decidable with axiom (16). We have $\mathsf{l_W} y \subseteq \mathsf{s}_1(\mathsf{l_W} y) = \mathsf{l_W}(\mathsf{s}_i y)$. If $\mathsf{l_W} x \subseteq \mathsf{l_W} y$, we immediately get $x \leq \mathsf{s}_i y$ with transitivity. Otherwise, i.e. $\mathsf{l_W} y \subseteq \mathsf{l_W} x$, we have either $\mathsf{l_W} x = \mathsf{l_W} y$ and thus $\mathsf{l_W} x \subseteq \mathsf{l_W}(\mathsf{s}_i y)$, or $\mathsf{l_W} y \subseteq \mathsf{p_W}(\mathsf{l_W} x)$. With (i) we get $(\exists z \subseteq \mathsf{p_W} x)(\mathsf{l_W} z = \mathsf{l_W} y)$ and thus $\mathsf{l_W}(\mathsf{s}_i y) = \mathsf{s}_1(\mathsf{l_W} y) = \mathsf{s}_1(\mathsf{l_W} z) = \mathsf{l_W}(\mathsf{s}_i z)$. Since $x \subseteq \mathsf{p_W} x$, we have $\mathsf{s}_0 z \subseteq x \vee \mathsf{s}_1 z \subseteq x$ and with part 3) we get $\mathsf{l_W}(\mathsf{s}_i z) \subseteq \mathsf{l_W} x$. Thus, $A[\mathsf{s}_i y]$ for $i = 0, 1$ and we can apply induction to get $(\forall y \in \mathsf{W})A[y]$.□

Remark 3.8 The above lemma, part 2) and the definition of $\leq$ immediately imply that also $\leq$ is transitive, provably in PET.

In PET, type induction can also be stated differently, in the form of bounded type induction ($\mathsf{T\text{-}I}^{\mathsf{b}}_{\mathsf{W}}$). As this notion will be more convenient in the following proofs, we will prove that both formulations are equivalent:

($\mathsf{T\text{-}I}^{\mathsf{b}}_{\mathsf{W}}$) $\quad a \in \mathsf{W} \wedge \epsilon \in X \wedge (\forall x \subseteq a)(\mathsf{p_W} x \in X \to x \in X) \to a \in X$

Lemma 3.9 *We have that* ($\mathsf{T\text{-}I_W}$) *and* ($\mathsf{T\text{-}I}^{\mathsf{b}}_{\mathsf{W}}$) *are provably equivalent in* PET *without* ($\mathsf{T\text{-}I_W}$).

Proof The fact that ($\mathsf{T\text{-}I}^{\mathsf{b}}_{\mathsf{W}}$) entails ($\mathsf{T\text{-}I_W}$) is trivial: assume the conditions for ($\mathsf{T\text{-}I_W}$) hold. Obviously, also the requirements for ($\mathsf{T\text{-}I}^{\mathsf{b}}_{\mathsf{W}}$) are fulfilled for any $a \in \mathsf{W}$ and $a \in X$ is now immediate.

For the converse implication, we take any type X and some $a \in \mathsf{W}$. Then we can build the type

$$Y := \{x : \mathsf{c}_{\subseteq} x a = 1 \vee x \in X\}$$

Its membership condition is equivalent to $x \subseteq a \to x \in X$ for $x \in \mathsf{W}$.

Now we assume that the conditions for $(\mathsf{T\text{-}I}^{\mathsf{b}}_{\mathsf{W}})$ hold, i.e. we assume that $\epsilon \in X$ and

$$(\forall x \subseteq a)(\mathsf{p}_{\mathsf{W}} x \in X \to x \in X) \tag{3.1}$$

Obviously, $\epsilon \in Y$ from the definition of Y. Now we have to show that if $x \in Y$ then also $\mathsf{s}_0 x \in Y \wedge \mathsf{s}_1 x \in Y$. Hence, assume $x \in Y$. If $\mathsf{c}_{\subseteq}(\mathsf{s}_i x)a = 1$, then obviously $\mathsf{s}_i x \in Y$. Otherwise, i.e. $\mathsf{s}_i x \subseteq a$, transitivity of the subword relation readily entails $x \subseteq a$, which implies $x \in X$. We make use of 3.1 to derive $\mathsf{s}_i x \in X$ and thus $\mathsf{s}_i x \in Y$.

Now we proved the conditions for applying (common) type induction $(\mathsf{T\text{-}I}_{\mathsf{W}})$ and get $(\forall x \in \mathsf{W})(x \in Y)$. Therefore also $a \in Y$. Since $a \subseteq a \equiv \mathsf{c}_{\subseteq} a a = 0$, a must be in X. □

We now want to prove that every function f of type $\mathsf{W} \mapsto \mathsf{W}$ can be bounded by a monotone function in the sense of the following lemma. We will construct this function f^* as the function taking the maximum of f applied to all subwords with respect to the tally length. Informally, $f^* x = \max_{y \subseteq x} f y$ where the result is maximised only with respect to the tally length. The functional mapping f to f^* is a well-known basic feasible functional, cf. e.g. Cook and Kapron [13, 43].

Lemma 3.10 *There is a closed term* max *such that* PET *proves:*

1) $f : \mathsf{W} \mapsto \mathsf{W} \to \mathsf{max} f : \mathsf{W} \mapsto \mathsf{W}$,
2) $f : \mathsf{W} \mapsto \mathsf{W} \wedge f^* = \mathsf{max} f \wedge x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x \subseteq y \to f^* x \leq f^* y$,
3) $f : \mathsf{W} \mapsto \mathsf{W} \wedge f^* = \mathsf{max} f \wedge x \in \mathsf{W} \to f x \leq f^* x$,
4) $f : \mathsf{W} \mapsto \mathsf{W} \wedge f^* = \mathsf{max} f \wedge x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x \subseteq y \to f x \leq f^* y$.

Proof We first define an auxiliary function $\mathsf{max}^{\mathrm{arg}}$ locating the subword where f is maximised and write $\widetilde{f} = \mathsf{max}^{\mathrm{arg}} f$:

$$\begin{array}{rcl} \mathsf{max}^{\mathrm{arg}} f\epsilon & \simeq & \epsilon \\ \mathsf{max}^{\mathrm{arg}} f(\mathsf{s}_i x) & \simeq & \begin{cases} \mathsf{max}^{\mathrm{arg}} fx & \text{if } f(\mathsf{s}_i x) \leq f(\mathsf{max}^{\mathrm{arg}} fx) \\ \mathsf{s}_i x & \text{otherwise} \end{cases} \end{array}$$

We can construct this term with Lemma 2.7, λ-abstraction and definition by cases. Now we have to prove that $\widetilde{f} : \mathsf{W} \mapsto \mathsf{W}$, provided $f : \mathsf{W} \mapsto \mathsf{W}$.

We fix some $a \in \mathsf{W}$ and define the $\Sigma^{\mathsf{b}}_{\mathrm{T}}$-formula $A[x] \equiv (\exists y \leq a)(\widetilde{f}x = y)$. Now we will make use of $(\mathsf{T}\text{-}\mathsf{I}^{\mathsf{b}}_{\mathsf{W}})$ in order to show $A[a]$ for an arbitrary $a \in \mathsf{W}$:

$x = \epsilon$: $\widetilde{f}\epsilon = \epsilon \leq a$.

Assume $x \subseteq a$ and $A[\mathsf{p}_{\mathsf{W}} x]$: Since we know that $z = \widetilde{f}(\mathsf{p}_{\mathsf{W}} x) \in \mathsf{W}$ we can decide whether $fx \leq fz$ or not. In the first case, by induction hypothesis we have $\widetilde{f}x = \widetilde{f}(\mathsf{p}_{\mathsf{W}} x) \leq a$. In the latter case we have $\widetilde{f}x = x$ which gives $\widetilde{f}x \leq a$ with Lemma 3.7.3).

Now we define $\mathsf{max} := (\lambda f.(\lambda x.f(\mathsf{max}^{\mathrm{arg}} fx)))$.

1) $\mathsf{max} f = (\lambda x.f(\mathsf{max}^{\mathrm{arg}} fx))$ by definition. For $x \in \mathsf{W}$, $\mathsf{max}^{\mathrm{arg}} fx \in \mathsf{W}$ if $f : \mathsf{W} \mapsto \mathsf{W}$ as proved above. $\mathsf{max} fx \in \mathsf{W}$ is now immediate from the totality of f.

2) This follows from the construction of f^* by induction on y. We work informally in PET and assume that $f : \mathsf{W} \mapsto \mathsf{W}$ and $x \in \mathsf{W}$. Let

$$A \equiv \mathsf{c}_{\subseteq} xy = 1 \vee \mathsf{c}_{\subseteq}(\mathsf{l}_{\mathsf{W}}(f^* x))(\mathsf{l}_{\mathsf{W}}(f^* y)) = 0$$

A is obviously a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula. For $x, y \in \mathsf{W}$, we have $A[x, y]$ iff $x \subseteq y \rightarrow f^* x \leq f^* y$. We will now prove this by induction on y.

If $y = \epsilon$, we have either $\mathsf{c}_{\subseteq} x\epsilon = 1$ and thus $A[x, \epsilon]$ or $x = \epsilon$. In this case, $f^* x = f^* y$.

Assume $A[x,y]$ holds. Now we have to prove $A[x,\mathsf{s}_i y]$ $(i=0,1)$. Again, if $\mathsf{c}_{\subseteq} x(\mathsf{s}_i y)=1$, we are obviously done. Otherwise we have either $x=\mathsf{s}_i y$ and clearly $A[x,\mathsf{s}_i y]$ or $x\subseteq y$. In the latter case, we have $\mathsf{l}_{\mathsf{W}}(f^*x)\subseteq \mathsf{l}_{\mathsf{W}}(f^*y)$ by induction hypothesis. By definition $f^*(\mathsf{s}_i y) = f(\mathsf{max}^{\mathrm{arg}} f(\mathsf{s}_i y))$. Then $\mathsf{max}^{\mathrm{arg}} f(\mathsf{s}_i y)=\mathsf{max}^{\mathrm{arg}} fy$ if $f(\mathsf{s}_i y)\leq f(\mathsf{max}^{\mathrm{arg}} fy)$ and therefore $f^*(\mathsf{s}_i y)=f^*y$. Otherwise, we have $f(\mathsf{max}^{\mathrm{arg}} fy)<f(\mathsf{s}_i y)$ and therefore $f^*y\leq f^*(\mathsf{s}_i y)$. With Lemma 3.7.2), we get $f^*x\leq f^*(\mathsf{s}_i y)$ as desired.

3) We prove this by induction on the $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula $A\equiv \mathsf{c}_{\subseteq}(fx)(f^*x)=0$.

For $x=\epsilon$, $f^*\epsilon = f\epsilon$ by definition. We assume $A[x]$ and aim at showing $A[\mathsf{s}_i x]$ $(i=0,1)$. $f^*(\mathsf{s}_i x)=f(\mathsf{max}^{\mathrm{arg}} f(\mathsf{s}_i x))$. As above, if $\mathsf{max}^{\mathrm{arg}} f(\mathsf{s}_i x)=\mathsf{max}^{\mathrm{arg}} fx$, then $f(\mathsf{s}_i x)\leq f(\mathsf{max}^{\mathrm{arg}} fx)=f^*(\mathsf{s}_i x)$. Otherwise, $\mathsf{max}^{\mathrm{arg}} f(\mathsf{s}_i x)=\mathsf{s}_i x$ and obviously $fx=f^*x$.

4) Immediate with parts 2) and 3) and Lemma 3.7. □

With the help of these auxiliary lemmas, we are now in the position to state the main theorem of this section, the embedding of PT^- into PET. The desired lower bounds immediately follow from this inclusion.

Theorem 3.11 PT^- *is contained in* PET.

Proof Clearly, we only need to prove that $\mathsf{PET}\vdash(\Sigma^{b-}_{\mathsf{W}}\text{-}\mathsf{l}_{\mathsf{W}})$, i.e. that induction holds for any Σ^{b-}_{W}-formula $A[x]\equiv(\exists y\leq fx)B[f,x,y]$. Let us work informally in PET and assume, in addition,

$$f:\mathsf{W}\to\mathsf{W}\wedge A[\epsilon]\wedge(\forall x\in\mathsf{W})(A[x]\to A[\mathsf{s}_0 x]\wedge A[\mathsf{s}_1 x]). \tag{3.2}$$

Now we fix $c\in\mathsf{W}$ and aim at showing $A[c]$. First, we need Lemma 3.10 in order to show that for $c\in\mathsf{W}$ and $x\subseteq c$ we get

$$(\exists y\leq fx)B[x,y]\leftrightarrow(\exists y\leq f^*c)(y\leq fx\wedge B[x,y]) \tag{3.3}$$

With Lemma 3.10 we immediately get $fx\leq f^*c$ and then the equivalence is obvious by Lemma 3.7. As $f^*c\in\mathsf{W}$ for $c\in\mathsf{W}$ and B an

$\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula by definition of Σ^{b-}_{W}, we can invoke Theorem 3.5 and construct a type X with the defining property

$$(\forall x \subseteq c)(x \in X \leftrightarrow (\exists y \leq f^* c)(y \leq fx \wedge B[x, y])) \tag{3.4}$$

By (3.2), (3.3) and (3.4) we immediately obtain

$$\epsilon \in X \wedge (\forall x \subseteq c)(\mathsf{p}_{\mathsf{W}} x \in X \to x \in X) \tag{3.5}$$

Now we can apply (T-$\mathsf{I}^{\mathsf{b}}_{\mathsf{W}}$) and derive $c \in X$ and hence $A[c]$ as desired.□

Together with Lemma 2.10 and Remark 2.13, this implies that bounded recursion on notation can be represented as a type two functional in PET. Hence, the following corollary is immediate.

Corollary 3.12 *The polynomial time computable functions are provably total in* PET.

In Section 3.4 we will propose a natural extension of PET allowing us to embed the full theory PT.

3.3 Upper Bounds

In this section we show the conservativity of PET over PT^- for $\mathcal{L}_{\mathsf{W}}$ sentences. As the formula expressing the totality of a function on W is an $\mathcal{L}_{\mathsf{W}}$ sentence, the desired upper bounds are then immediate such that the provably total functions of PET coincide with the polynomial time computable functions.

We use a model-theoretic argument to establish the desired upper bounds for PET. To be precise, our proof strategy is to give a recipe for extending any model of PT^- by an interpretation for the second order part and then to prove that the resulting structure indeed is a model of PET. However, we will indicate a syntactic proof of the same result later in Section 4.5.

We first give a formal definition of a structure for the language $\mathcal{L}^2_{\mathsf{W}}$. It depends on models for PT^- as introduced in Section 2.3 and extends the definition of an $\mathcal{L}_{\mathsf{W}}$ structure given there.

Definition 3.13 ($\mathcal{L}^2_{\mathsf{W}}$ Structure)
A $\mathcal{L}^2_{\mathsf{W}}$-structure $\mathcal{M}^\star$ is a tuple

$$(\mathcal{M}, \mathcal{T}, \mathcal{E}, \mathcal{R}, \underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{dom}}, \underline{\mathsf{inv}})$$

such that the following conditions hold:

(i) $\mathcal{M}$ is a $\mathcal{L}_{\mathsf{W}}$-structure according to Definition 2.19,
(ii) $\mathcal{T}$ is a non-empty set of subsets of $|\mathcal{M}|$,
(iii) $\mathcal{E}$ and $\mathcal{R}$ are a non-empty subsets of $|\mathcal{M}| \times \mathcal{T}$, and
(iv) $\underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{dom}}, \underline{\mathsf{inv}}$ are elements of $|\mathcal{M}|$.

$\mathcal{M}^\star$ is called a **standard** structure if $\mathcal{E}$ is the usual element relation $\in$ on $|\mathcal{M}| \times \mathcal{T}$ and is written $(\mathcal{M}, \mathcal{T}, \mathcal{R}, \underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{dom}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{inv}})$ ⊛

Model Construction

Now we give a scheme for creating a model $\mathcal{M}^\star$ of PET: We take any model $\mathcal{M}$ of PT^- e.g. one as described in Section 2.3. First, we need to choose selected elements $\underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{dom}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{inv}}$ of $|\mathcal{M}|$ as interpretations for the corresponding constants of $\mathcal{L}^2_{\mathsf{W}}$. This can easily be done in such a way that $\mathrm{c}x \neq \mathrm{c}y$ and $\mathrm{c}u \neq \mathrm{d}v$ for all $\mathrm{c} \neq \mathrm{d} \in \{\underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{dom}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{inv}}\}$ and all $x \neq y, u, v \in |\mathcal{M}|$.

We build our model in stages corresponding to the type constructors of PET. In other words, we first define interpretations for the basic types (identity and initial segments of the binary words). Then we proceed by creating the composed types such as union and domain from the types already constructed at the previous level.

Formally, for the construction of $\mathcal{T}$ and $\mathcal{R}$ we introduce sets $R_k \subseteq |\mathcal{M}|$ by induction on the natural number k and simultaneously we establish

a set $ext(m) \subseteq |\mathcal{M}|$ for each $m \in R_k$. Then we set

$$\begin{aligned} \mathcal{T}_k &:= \{ext(m) : m \in R_k\}, \\ \mathcal{R}_k &:= \{(m, ext(m)) : m \in R_k\}, \\ \mathcal{M}_k^\star &:= (\mathcal{M}, \mathcal{T}_k, \mathcal{R}_k, \underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{dom}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{inv}}). \end{aligned}$$

$k = 0$: R_0 contains names of the base types as well as their obvious extensions, i.e.

- $\underline{\mathsf{id}} \in R_0$; $ext(\underline{\mathsf{id}}) := \{(m, m) : m \in |\mathcal{M}|\}$
- $\underline{\mathsf{w}}a \in R_0$ if $a \in \mathsf{W}^{\mathcal{M}}$; $ext(\underline{\mathsf{w}}a) := \{m \in |\mathcal{M}| : \mathcal{M} \models m \in \mathsf{W} \wedge m \leq a\}$

$k > 0$: R_k contains R_{k-1}. In addition, if $a, b \in R_{k-1}$ then

- $\underline{\mathsf{un}}(a, b) \in R_k$; $ext(\underline{\mathsf{un}}(a, b)) := \{m \in |\mathcal{M}| : \mathcal{M}_{k-1}^\star \models m \mathrel{\dot\in} a \vee m \mathrel{\dot\in} b\}$
- $\underline{\mathsf{int}}(a, b) \in R_k$; $ext(\underline{\mathsf{int}}(a, b)) := \{m \in |\mathcal{M}| : \mathcal{M}_{k-1}^\star \models m \mathrel{\dot\in} a \wedge m \mathrel{\dot\in} b\}$
- $\underline{\mathsf{dom}}(a) \in R_k$; $ext(\underline{\mathsf{dom}}(a)) := \{m \in |\mathcal{M}| : \mathcal{M}_{k-1}^\star \models \exists y((m, y) \mathrel{\dot\in} a)\}$
- $\underline{\mathsf{inv}}(f, a) \in R_k$; $ext(\underline{\mathsf{inv}}(f, a)) := \{m \in |\mathcal{M}| : \mathcal{M}_{k-1}^\star \models fm \mathrel{\dot\in} a\}$

Finally, we define $\mathcal{T} := \bigcup_{k\in\mathbb{N}} \mathcal{T}_k$ and $\mathcal{R} := \bigcup_{k\in\mathbb{N}} \mathcal{R}_k$. Then our desired $\mathcal{L}^2_{\mathsf{W}}$ structure is given by

$$\mathcal{M}^\star := (\mathcal{M}, \mathcal{T}, \mathcal{R}, \underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{dom}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{inv}}).$$

Now we can state the crucial theorem of this section:

Theorem 3.14 (Model Extension) *Any model $\mathcal{M}^\star$ constructed as described from a model $\mathcal{M}$ of PT^- satisfies the following conditions:*

1) $\mathcal{M} \models A \iff \mathcal{M}^\star \models A$ for any $\mathcal{L}_{\mathsf{W}}$ sentence A,

2) $\mathcal{M}^\star \models (\mathsf{T}\text{-}\mathsf{I}_{\mathsf{W}})$,

3) $\mathcal{M}^\star \models \mathsf{PET}$.

Proof

1) As the first-order part of the model $\mathcal{M}$ remains untouched, the same $\mathcal{L}_{\mathsf{W}}$ formulas are satisfied in the extended model.

2) In order to prove that $\mathcal{M}^\star$ satisfies type induction, we will show that every type X of $\mathcal{T}$ is *weakly* Σ^{b-}_{W} *definable*. This means that membership for this type can be expressed by a Σ^{b-}_{W} formula of $\mathcal{L}_{\mathsf{W}}$ with a fixed bound, namely

$$X = \{m \in |\mathcal{M}| : \mathcal{M} \models (\exists y \leq \mathsf{k}bm)B[m, y]\}$$

for some $b \in \mathsf{W}^{\mathcal{M}}$ and some formula B which is W-free, positive and not containing $\forall$. B may possibly contain parameters in $|\mathcal{M}|$. We now use the fact that types are added to $\mathcal{T}$ as the extension of a name and that every name is added at a certain level. Therefore, we will make induction on R_k to show that every name a can be weakly Σ^{b-}_{W} defined by a formula A. In the process, we make use of the pairing function $\langle \cdot, \cdot \rangle$ in PTLS (c.f. Appendix C).

$a \in \mathbf{R_0}$: Then $a = \underline{\mathsf{id}}$ or $a = \underline{\mathsf{w}}b$ for some $b \in \mathsf{W}^{\mathcal{M}}$. In the former case, we choose A to be $\exists z(x = (z, z))$ which is a Σ^{b-}_{W} formula without bound. In the latter case, A is set to $(\exists y \leq b)(x = y)$.

$a \in \mathbf{R_{n+1}} \setminus \mathbf{R_n}$: Suppose $b, c \in R_n$. By induction hypothesis, there are corresponding defining formulas B and C,

$$\begin{aligned} B[x] &\equiv (\exists y \leq u)B'[x, y], \\ C[x] &\equiv (\exists z \leq v)C'[x, z], \end{aligned}$$

with $u, v \in \mathsf{W}^{\mathcal{M}}$. We distinguish the following cases:

– $a = \underline{\mathsf{un}}(b, c)$: Then we set $d = \langle u, v \rangle$ and $A[x]$ to be

$$(\exists y \leq d)[(\langle y \rangle_0 \leq u \wedge B'[x, \langle y \rangle_0]) \vee (\langle y \rangle_1 \leq v \wedge C'[x, \langle y \rangle_1])],$$

where we use the properties of the polynomial time pairing function described in Appendix C.

– $a = \underline{\mathsf{int}}(b, c)$: Again we take $d = \langle u, v\rangle$ and let $A[x]$

$$(\exists y \leq d)(\langle y\rangle_0 \leq u \wedge B'[x, \langle y\rangle_0] \wedge \langle y\rangle_1 \leq v \wedge C'[x, \langle y\rangle_1]),$$

– $a = \underline{\mathsf{dom}}(b)$: We can choose the formula $A[x]$ to be

$$(\exists y \leq u)\exists z B'[(x, z), y].$$

– $a = \underline{\mathsf{inv}}(f, b)$: In this case we can choose $A[x]$ to be $B[fx]$.

In all these cases, we obviously have that $x \in ext(a)$ iff $A[x]$.

This concludes our argument that each type X in $\mathcal{T}$ can be weakly Σ^{b-}_{W} defined in $\mathcal{M}$. It is now immediate that $\mathcal{M}^\star$ satisfies $(\mathsf{T}\text{-}\mathsf{I}_{\mathsf{W}})$ since $\mathcal{M}$ validates $(\Sigma^{b-}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$.

3) The remaining axioms are obvious by construction. □

The following corollaries are now immediate by Gödel completeness and Theorem 3.11.

Corollary 3.15 PET *is a conservative extension of* PT^- *with respect to* $\mathcal{L}_{\mathsf{W}}$ *formulas.*

Corollary 3.16 *The provably total functions of* PET *coincide with the functions computable in polynomial time.*

Strahm [54] showed that the provably total type two functionals of PT coincide with the basic feasible functionals of type two. An analysis of this arguments yields that already $(\Sigma^{b-}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ suffices for the desired result. Therefore, the provably total type two functionals of PET are also the basic feasible functionals.

3.4 Extensions for PET

In this section we discuss some natural extensions of PET which mostly rely on Cantini [6]. He studies –among other things– various extensions

of the first-order theory PT by means of axioms for self-referential truth, a uniformity principle and an axiom of positive choice, see Section 2.5. We make use of those principles to study interesting extensions of PET without increasing the proof-theoretic strength.

3.4.1 Uniformity and Universal Quantification

Cantini [6] adds a uniformity principle for positive $\mathcal{L}_{\mathsf{W}}$ formulas to PT and proves that this yields a theory whose provably total functions are still those computable in polynomial time. Cantini formulates the uniformity principle for a truth predicate defined for positive formulas. In our context, we can state Cantini's principle as follows. For each *positive* $\mathcal{L}_{\mathsf{W}}$ formula $A[x, y]$:

$$(\mathsf{UP}) \qquad \forall x(\exists y \in \mathsf{W})A[x, y] \to (\exists y \in \mathsf{W})(\forall x)A[x, y]$$

We exploit the fact that $(\exists y \leq t)A[x] \equiv (\exists y \in \mathsf{W})(y \leq t \wedge A[x])$ which is obviously a positive formula. Therefore, we can specify the following form of *bounded uniformity* for positive $\mathcal{L}_{\mathsf{W}}$ formulas $A[x, y]$ which is readily entailed by (UP):

$$(\mathsf{UP'}) \qquad \forall x(\exists y \leq t)A[x, y] \to (\exists y \leq t)(\forall x)A[x, y]$$

The principle (UP') leads to a very natural extension of PET by adding a type existence axiom for universal quantification. This axiom is the natural dual analogue of the domain type present in PET. We first add an additional constant all to $\mathcal{L}^2_{\mathsf{W}}$ and spell out the axiom:

$$(\mathsf{all}) \qquad \Re(a) \to \Re(\mathsf{all}(a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{all}(a) \leftrightarrow \forall y((x, y) \mathrel{\dot\in} a))$$

The presence of the axiom (all) makes the type existence axioms more symmetric, i.e. the types are generated from base types (initial segments of W and the identity type) by closing under inverse images, unions, intersections, existential quantification (domain) and universal quantification.

In order to see that (all) does not increase the proof-theoretic strength of PET, we extend the arguments for the upper bounds given in the previous section. Instead of a model of PT, we now take any model $\mathcal{M}$ of PT+(UP'). We can extend $\mathcal{M}$ to a model $\mathcal{M}^\star$ of PET+(all) in the obvious way by adding a clause for universal quantification in the construction of Section 3.3. To be precise, we add the following case for $k > 0$:

– $\underline{\mathsf{all}}(a) \in R_k$ and $ext(\underline{\mathsf{all}}(a)) := \{m \in |\mathcal{M}| : \mathcal{M}^\star_{k-1} \models \forall y((m, y) \mathrel{\dot\in} a)\}$

The crucial step in Theorem 3.14 is to show that $\mathcal{M}^\star$ satisfies type induction. Towards this aim, we follow the strategy in the proof of the above mentioned theorem and show that each type in $\mathcal{M}^\star$ is weakly Σ^b_{W} definable in $\mathcal{M}$. Because of (UP'), we can drop the restriction to Σ^{b-}_{W} formulas and allow the appearance of $\forall$. The only new case occurs for $a = \underline{\mathsf{all}}(b)$ where we already know by induction hypothesis that b has a weak Σ^b_{W} definition, say $B[u] = (\exists v \leq s)B'[u, v]$. Then we have that $\mathcal{M}^\star$ satisfies

$$\forall u(u \mathrel{\dot\in} b \leftrightarrow (\exists v \leq s)B'[u, v])$$

Thus, by construction of our model $\mathcal{M}^\star$ we obtain that

$$\forall x[x \mathrel{\dot\in} a \leftrightarrow \forall y B[(x, y)] \leftrightarrow \forall y(\exists v \leq s)B'[(x, y), v]]$$

Now we are in a position to invoke (UP') in order to get the equivalence

$$\forall y(\exists v \leq s)B'[(x, y), v] \leftrightarrow (\exists v \leq s)\forall y B'[(x, y), v].$$

The formula $(\exists v \leq s)\forall y B'[\langle x, y\rangle, v]$ is clearly a weak Σ^b_{W} formula and, hence, $a = \underline{\mathsf{all}}(b)$ is weakly Σ^b_{W} definable. This shows that type induction holds in our model $\mathcal{M}^\star$.

To summarise, we proved that PET+(all) is conservative over PT+(UP) for $\mathcal{L}_{\mathsf{W}}$ formulas. As Cantini [6] established the upper bounds for the latter theory, we have that the provably total functions of PET+(all) coincide with the functions computable in polynomial time.

Furthermore, the full theory PT is contained in PET + (all): In the presence of the additional principles, we can establish the comprehension scheme for an enlarged formula class which is also closed under universal quantification. Therefore, we have to extend Definition 3.2 by the following clause:

5) If A is the formula $\forall x B$ with $B \in \Sigma^{\mathsf{b}}_{\mathsf{T}}$ and $x \notin \mathrm{FV}_{\mathsf{W}}(B)$, then A is a $\Sigma^{\mathsf{b}}_{\mathsf{T}}$ formula and $\mathrm{FV}_{\mathsf{W}}(A) := \mathrm{FV}_{\mathsf{W}}(B)$.

We have to add a further case to Definition 3.4:

$$\rho_A x.(\forall x C) \quad := \quad \mathsf{all}(\rho_A x.(C[(x)_0/x, (x)_1/y]))$$

Theorem 3.5 holds for this extended class of formulas in the presence of (all). With these preparations, we can obviously expand the proof of Theorem 3.11 to induction for Σ^b_{W}-formulas as the auxiliary lemmas do not depend on the absence of universal quantification.

3.4.2 Axiom of Choice

In addition to the uniformity principle discussed above, Cantini [6] also considers a form of positive choice in the context of PT with a partial truth predicate and shows that this principle does not increase the proof-theoretic strength. Positive choice in the language $\mathcal{L}_{\mathsf{W}}$ includes for each positive $\mathcal{L}_{\mathsf{W}}$ formula $A[x, y]$ the statement

$$(\mathsf{AC}) \qquad (\forall x \in \mathsf{W})(\exists y \in \mathsf{W})A[x, y] \to (\exists f : \mathsf{W} \mapsto \mathsf{W})(\forall x \in \mathsf{W})A[x, fx]$$

If we extend this schema to the language $\mathcal{L}^2_{\mathsf{W}}$ then, in addition, A is allowed to contain positive occurrences of subformulas of the form $t \in X$. In other words, choice is available for positive elementary formulas A of $\mathcal{L}^2_{\mathsf{W}}$. We will call this principle (AC) as well as it will always be clear from the context whether we refer to the first-order or second order form of the choice axiom.

If we start from a model of $\mathsf{PT} + (\mathsf{AC})$ in the model construction of Section 3.3, we can obviously extend this to a model of PET which satisfies choice (AC) in the extended language $\mathcal{L}^2_{\mathsf{W}}$ as described above. With the same argument, the provably total functions of $\mathsf{PET}+(\mathsf{AC})$ are still the polynomial time computable ones.

3.4.3 Totality and Extensionality

The upper bound computations in Strahm [53] for PT and Cantini [6] for its various extensions actually validate stronger applicative axioms as those spelled out in PT. In particular, our theories can be augmented by totality of application and extensionality of operations without increasing the proof-theoretic strength.

As already mentioned in Section 2.1.3, the addition of the totality axiom (Tot) reduces partial combinatory logic to total combinatory logic. Therefore the logic of partial terms can be replaced by ordinary predicate logic with equality. If, in addition, extensionality of operations is assumed, then the applicative basis is equivalent to ordinary untyped extensional lambda calculus $\lambda\eta$. We are now in a position to summarise the results of this section in the following theorem.

Theorem 3.17 *The provably total functions of* PET *augmented by any combination of the principles* (all)*,* (AC)*,* (Tot)*, and* (Ext) *coincide with the polynomial time computable functions.*

3.5 Further Complexity Classes

Beside PT, Strahm [53] studies three more weak complexity classes, namely simultaneously polynomial time and linear space, polynomial space, and linear space. As recapitulated in Section 2.1.4, they mainly differ in the presence of the axioms about multiplication of binary words and in the induction scheme. We employ the same concept to generate second order theories corresponding to those classes.

Contrary to previous sections, we use the full axiomatisation B as given in Section 2.1.4. We abandon minimality of the single theories for the sake of uniformity in axiomatisation, which saves us the trouble of proving e.g. Lemma 3.7.

First, we introduce the scheme of lexicographic type induction and define the theories. Then we establish their proof-theoretic strength in the respective subsections. The proofs are mostly similar to the one carried out in detail in the previous sections and are thus mainly sketched.

The axiom of lexicographic induction on types, $(\mathsf{T\text{-}I}_\ell)$, is spelled out as follows:

$$(\mathsf{T\text{-}I}_\ell) \qquad \epsilon \in X \wedge (\forall x \in \mathsf{W})(x \in X \to \mathsf{s}_\ell x \in X) \to (\forall x \in \mathsf{W})(x \in X)$$

To improve readability, we define base theories including types:

Definition 3.18 (Base Theories)
Theories BT, $\mathsf{BT}(*)$ and $\mathsf{BT}(*,\times)$ are defined as the corresponding applicative theory B (see Section 2.1.4) augmented by the ontological axioms (group I., p. 38) and the axioms about type construction (group II., p. 39) ⊛

We are now ready to introduce theories with types corresponding to the theories PT, PTLS, LS, and PS, respectively. PET is already discussed in detail above, but is repeated for completeness. The axiomatisation here is based on the full theory $\mathsf{B}(*,\times)$ instead of BOW as before. Those formulations are equivalent as the (relevant) omitted axioms are derivable in this setting.

Definition 3.19
We define the following for theories:

$$\mathsf{PET} \equiv \mathsf{BT}(*,\times) + (\mathsf{T\text{-}I}_\mathsf{W}) \qquad \mathsf{PLSET} \equiv \mathsf{BT}(*) + (\mathsf{T\text{-}I}_\mathsf{W})$$
$$\mathsf{PSET} \equiv \mathsf{BT}(*,\times) + (\mathsf{T\text{-}I}_\ell) \qquad \mathsf{LSET} \equiv \mathsf{BT}(*) + (\mathsf{T\text{-}I}_\ell)$$

⊛

Since all theories share the same type existence axioms and the proof of Theorem 3.5 does depend on neither multiplication nor induction, comprehension is available for the same class of restricted elementary formulas, $\Sigma^{\mathsf{b}}_{\mathrm{T}}$, as introduced in Section 3.1.2.

3.5.1 Polynomial Time and Simultaneously Linear Space

In order to show that the provably total functions are those computable in polynomial time and linear space, we follow the same course of action as we did for PET. We start by proving parts of Lemma 3.7 in our current setting. As transitivity of the subword relation is stated as an axiom, we can omit this part.

Lemma 3.20 *The following statements are provable in* PLSET*:*

1) $x \in \mathsf{W} \land z \in \mathsf{W} \land x \subseteq \mathsf{p}_{\mathsf{W}} z \to x \subseteq z$,

2) $x \in \mathsf{W} \land y \in \mathsf{W} \land x \subseteq y \to x \leq y$.

Proof

1) Immediate from the axioms.

2) We can write this property as $\mathsf{c}_{\subseteq} xy = 1 \lor \mathsf{c}_{\subseteq}(\mathsf{l}_{\mathsf{W}} x)(\mathsf{l}_{\mathsf{W}} y) = 0$ which obviously is a $\Sigma^{\mathsf{b}}_{\mathrm{T}}$ formula. We will concentrate on the case $x \subseteq y$ in the induction on y.

   $y = \epsilon$: This implies that also $x = \epsilon$ by axiom (17). Therefore obviously $\mathsf{l}_{\mathsf{W}} x = \mathsf{l}_{\mathsf{W}} y$.

   Assume the assertion holds for $\mathsf{p}_{\mathsf{W}} y$. To prove that $x \leq y$ if $x \subseteq y$ we make a case distinction according to axiom (18): if $x = y$ then $x \leq y$ is again obvious. Otherwise $x \subseteq \mathsf{p}_{\mathsf{W}} y$. By induction hypothesis we have $\mathsf{l}_{\mathsf{W}} x \subseteq \mathsf{l}_{\mathsf{W}}(\mathsf{p}_{\mathsf{W}} y)$ (i). Further, $\mathsf{l}_{\mathsf{W}} y = \mathsf{s}_1(\mathsf{l}_{\mathsf{W}} \mathsf{p}_{\mathsf{W}} y)$ (ii) with axiom (21). With the first part, we get $\mathsf{l}_{\mathsf{W}} x \subseteq \mathsf{l}_{\mathsf{W}} y$ from (i) and (ii). □

Since the rest of the proof of the lower bounds remains the same as before, we have established the following lower bounds:

Theorem 3.21 PTLS^- *is contained in* PLSET.

Proof Lemma 3.9 also holds for PLSET. The proof that (T-I_W) entails (T-$\mathsf{I}^\mathsf{b}_\mathsf{W}$) does not refer to multiplication and therefore does not need adjustment. Multiplication is not used in the construction of the functional max, either and thus, PET can be replaced by PLSET in Lemma 3.10.

The final step in the proof of the lower bounds of PET was proving (Σ^{b-}_W-I_W). Since we have already seen that the auxiliary lemmas still apply, the same proof also works for PLSET. □

We can now state the upper bounds for PLSET which are again proved with a model theoretic argument.

Theorem 3.22 PLSET *is a conservative extension of* PTLS^- *with respect to* $\mathcal{L}_\mathsf{W}$ *formulas*

Proof We employ the model construction as described for PET on page 50, but starting off from a model of PTLS^- instead of PT^-. We replace PT^- and PET by PTLS^- and PLSET in Theorem 3.14 and the same proof still runs through. □

From the previous two theorems we immediately derive that the provably total functions of PLSET are the functions computable simultaneously in polynomial time and linear space.

3.5.2 Polynomial Space and Linear Space

In order to establish the proof-theoretic strength of PSET and LSET, we follow the same procedure again, though adapting the lemmas and theorems for lexicographic induction instead of induction on notation. As we have seen above, the proofs are not altered depending on the presence of the axioms about multiplication. Therefore, we treat both theories in parallel.

We first turn to the lower bounds and hence prove the following auxiliary lemma.

Lemma 3.23 *The following statement is provable in* $\mathsf{BT}(*) + (\mathsf{T}\text{-}\mathsf{I}_\ell)$*:*

$$x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x \leq y \rightarrow \mathsf{p}_\ell x \leq y$$

Proof Because of axioms (22) and (23), either $\mathsf{I}_\mathsf{W} x = \mathsf{I}_\mathsf{W}(\mathsf{p}_\ell x)$ or $\mathsf{I}_\mathsf{W} x = \mathsf{s}_1(\mathsf{I}_\mathsf{W}(\mathsf{p}_\ell x))$. The statement now follows with axiom (24). □

We give an equivalent reformulation of lexicographic type induction in the form of a bounded induction scheme. For induction on notation, we have to consider all subwords of the bound in the bounded induction scheme, whereas the induction condition runs over all words which are shorter or have the same length as the bound when dealing with lexicographic induction.

Lemma 3.24 *Over* $\mathsf{BT}(*)$, $(\mathsf{T}\text{-}\mathsf{I}^\mathsf{b}_\ell)$ *and* $(\mathsf{T}\text{-}\mathsf{I}_\ell)$ *are provably equivalent where bounded lexicographic type induction is defined as follows:*

$$(\mathsf{T}\text{-}\mathsf{I}^\mathsf{b}_\ell) \qquad a \in \mathsf{W} \wedge \epsilon \in X \wedge (\forall x \leq a)(\mathsf{p}_\ell x \in X \rightarrow x \in X) \rightarrow a \in X$$

Proof The fact that $(\mathsf{T}\text{-}\mathsf{I}^\mathsf{b}_\ell)$ entails $(\mathsf{T}\text{-}\mathsf{I}_\ell)$ is trivial: Assume $\epsilon \in X$ (i) and $(\forall x \in \mathsf{W})(\mathsf{p}_\ell x \in X \rightarrow x \in X)$ (ii). Now take any $s \in \mathsf{W}$. From (ii) we immediately get $(\forall x \leq s)(\mathsf{p}_\ell x \in X \rightarrow x \in X)$ and with (i) we are now able to apply $(\mathsf{T}\text{-}\mathsf{I}^\mathsf{b}_\ell)$ to get $s \in X$.

It remains to prove that $\mathsf{BT} + (\mathsf{T}\text{-}\mathsf{I}_\ell) \vdash (\mathsf{T}\text{-}\mathsf{I}_\ell)^\mathsf{b}$: We assume that the premises of $(\mathsf{T}\text{-}\mathsf{I}^\mathsf{b}_\ell)$ hold, i.e.

$$s \in \mathsf{W} \tag{3.6}$$

$$\epsilon \in X \tag{3.7}$$

$$(\forall x \leq s)(\mathsf{p}_\ell x \in X \rightarrow x \in X) \tag{3.8}$$

Now we want to prove that $s \in X$. Because of comprehension, we can construct a type

$$Y = \{x : \mathsf{c}_\subseteq(\mathsf{I}_\mathsf{W} x)(\mathsf{I}_\mathsf{W} s) = 1 \vee x \in X\}$$

For binary words w, this membership condition is equivalent to $w \leq s \to w \in X$.

We aim at showing $(\forall x \in \mathsf{W})(x \in Y)$:

$\epsilon \in Y$ as $\epsilon \in X$ by assumption (3.7).

Now we assume that $\mathsf{p}_\ell x \in Y$. Either $x \not\leq s \equiv \mathsf{c}_\subseteq(\mathsf{l}_\mathsf{W} x)(\mathsf{l}_\mathsf{W} s) = 1$ and therefore $x \in Y$. Otherwise (i.e. $x \leq s$), $x \in X$ because of (3.8) and Lemma 3.23. This enables us now to apply $(\mathsf{T}\text{-}\mathsf{I}_\ell)$ to get the desired result.

Due to (3.6), also $s \in Y$. Since $s \leq s \equiv \mathsf{c}_\subseteq(\mathsf{l}_\mathsf{W} s)(\mathsf{l}_\mathsf{W} s) = 0$, $s \in X$ is immediate from the construction of Y. □

Analogous to the functional max utilised for PET, we construct a functional max_ℓ such that, informally, $\mathsf{max}_\ell\, fx = \max_{y \leq x} fy$. Again $\subseteq$ is now replaced by $\leq$, and thus the maximum functional now runs over all words that are shorter of have the same length, while max only considers subwords. However, the result is again only maximised with regard to the tally length.

Lemma 3.25 *There is a closed term* max_ℓ *such that* $\mathsf{BT}(*) + (\mathsf{T}\text{-}\mathsf{I}_\ell)$ *proves the following:*

1) $f : \mathsf{W} \mapsto \mathsf{W} \to \mathsf{max}_\ell\, f : \mathsf{W} \mapsto \mathsf{W}$
2) $f : \mathsf{W} \mapsto \mathsf{W} \wedge f^* = \mathsf{max}_\ell\, f \wedge x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x \leq y \to f^* x \leq f^* y$
3) $f : \mathsf{W} \mapsto \mathsf{W} \wedge f^* = \mathsf{max}_\ell\, f \wedge x \in \mathsf{W} \to fx \leq f^* x$
4) $f : \mathsf{W} \mapsto \mathsf{W} \wedge f^* = \mathsf{max}_\ell\, f \wedge x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x \leq y \to fx \leq f^* y$

Proof Before we can define the functional max_ℓ, we need to define a functional $\mathsf{max}_\ell^{\mathrm{arg}}$ finding the argument maximising the function up to the given argument:

$$\mathsf{max}_\ell^{\mathrm{arg}}\, f\epsilon \simeq \epsilon$$

$$\mathsf{max}_\ell^{\mathrm{arg}}\, f(\mathsf{s}_\ell x) \simeq \begin{cases} \mathsf{max}_\ell^{\mathrm{arg}}\, fx & \text{if} f(\mathsf{s}_\ell x) \leq f(\mathsf{max}_\ell^{\mathrm{arg}}\, fx) \\ \mathsf{s}_\ell x & \text{otherwise} \end{cases}$$

Formally, $\mathsf{max}_\ell^{\mathrm{arg}} = (\lambda f.\mathsf{fix}tf)$ where

$$t = (\lambda hx.\mathsf{d_W}\epsilon[\mathsf{d_W}(h\mathsf{p}_\ell x)x(\mathsf{c}_\subseteq(\mathsf{l_W}fx)(\mathsf{l_W}f(h(\mathsf{p}_\ell x))))0]x\epsilon)$$

From this construction, we can prove that $\mathsf{max}_\ell^{\mathrm{arg}}\, f : \mathsf{W} \mapsto \mathsf{W}$ provided $f : \mathsf{W} \mapsto \mathsf{W}$. Actually, we will show that for any fixed $a \in \mathsf{W}$, we have $(\exists y \leq a)(\mathsf{max}_\ell^{\mathrm{arg}}\, fa = y)$ by making use of $(\mathsf{T\text{-}I}_\ell^{\mathsf{b}})$, which immediately implies the claimed totality of $\mathsf{max}_\ell^{\mathrm{arg}}\, f$. Now we fix some $a \in \mathsf{W}$. $\mathsf{max}_\ell^{\mathrm{arg}}\, f\epsilon = \epsilon \leq a$ is obvious. Assume $\mathsf{max}_\ell^{\mathrm{arg}}\, f(\mathsf{p}_\ell x) \leq a$ and $x \leq a$. Then $f(\mathsf{max}_\ell^{\mathrm{arg}}\, f(\mathsf{p}_\ell x)) \in \mathsf{W}$ and definition by cases can therefore be decided. The result is either x and $x \leq a$ by assumption; or $\mathsf{max}^{\mathrm{arg}} fx = \mathsf{max}_\ell^{\mathrm{arg}}\, f(\mathsf{p}_\ell x) \leq a$ by induction hypothesis. We can now apply $(\mathsf{T\text{-}I}_\ell^{\mathsf{b}})$ to get $(\mathsf{max}_\ell^{\mathrm{arg}}\, fa \leq a)$.

We can now define the maximising functional $\mathsf{max}_\ell = (\lambda fx.f(\mathsf{max}_\ell^{\mathrm{arg}}\, fx))$. The following proofs are similar to those of Lemma 3.10 and hence omitted.

1) Immediate from the totality of $\mathsf{max}_\ell^{\mathrm{arg}}$.

2) Proof by induction on y from the construction of $\mathsf{max}_\ell^{\mathrm{arg}}\, f$.

3) Proof by induction on x by construction of f^*

4) Immediate from (2) and (3) by transitivity of $\leq$. □

After this preparatory work, we can now prove the lower bounds:

Theorem 3.26

1) PS^- *is contained in* PSET.

2) LS^- *is contained in* LSET.

Proof It remains to be shown that PSET and LSET prove $(\Sigma^{b-}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$. Since this proof does not employ $\times$, it remains the same for both theories.

We take any Σ^{b-}_{W}-formula $A[x] = (\exists y \leq fx)B[f,x,y]$ and assume

$$f : \mathsf{W} \mapsto \mathsf{W} \wedge A[\epsilon] \wedge (\forall x \in \mathsf{W})(A[\mathsf{p}_\ell x] \to A[x]) \tag{3.9}$$

With Lemma 3.25, we get for any $c \in \mathsf{W}$ and $x \leq c$

$$(\exists y \leq fx)B[f,x,y] \leftrightarrow (\exists y \leq \mathsf{max}_\ell\, fc)(y \leq fx \wedge B[f,x,y]) \tag{3.10}$$

For any binary word c, comprehension lets us construct a type X_c such that

$$x \in X_c \leftrightarrow (\exists y \leq \mathsf{max}_\ell\, fc)(y \leq fx \wedge B[f,x,y]) \tag{3.11}$$

By assumption (3.9) $\epsilon \in X_c$ and $(\forall x \leq c)(\mathsf{p}_\ell x \in X_c \to x \in X_c)$. We can now apply $(\mathsf{T\text{-}I}^{\mathsf{b}}_\ell)$ to get $c \in X_c$. With (3.10) and (3.11) we get $A[c]$. □

As a consequence, bounded recursion on lexicographic notation is available in PSET and LSET. The polynomial space (linear space) computable functions are therefore provably total in PSET (LSET).

As usual, lexicographic induction implies induction on notation and therefore we also have bounded recursion on notation for these theories. This property is crucial for the upper bounds since we depend on the provably total pairing and projection functions defined in Appendix C for PLSET.

Lemma 3.27 $(\mathsf{T\text{-}I}_\ell)$ *entails* $(\mathsf{T\text{-}I}_{\mathsf{W}})$ *over* BT.

Proof The recursion operator r_ℓ is available in our setting as a consequence of the previous theorem. Therefore, the reasoning follows the proof of the analogous property for induction on Σ^b_{W} formulas as originally used by Buss e.g. in [3] and adapted for this setting in Strahm [53]:

Thus, we have functions msp, $|\cdot|$ and $\dot{-}$ behaving according the following descriptions. The most significant part function $\mathsf{msp} : \mathsf{W}^2 \mapsto \mathsf{W}$ fulfils for all $a, b \in \mathsf{W}, b \neq \epsilon$

$$\mathsf{msp}a\epsilon = a, \qquad \mathsf{msp}ab = \mathsf{p}_{\mathsf{W}}(\mathsf{msp}a(\mathsf{p}_\ell b)), \qquad \mathsf{msp}ab \leq a$$

In other words, it cuts off b bits from the back of a where b is seen as the representation of a natural number. The cut off function $\dot{-} : \mathsf{W}^2 \mapsto \mathsf{W}$ works along the lexicographic successors for both arguments:

$$a \dot{-} \epsilon = a, \qquad a \dot{-} b = \mathsf{p}_\ell(a \dot{-} (\mathsf{p}_\ell b)), \qquad a \dot{-} b \leq a$$

Furthermore, the length function $|\cdot| : \mathsf{W} \mapsto \mathsf{W}$ measures the length of a word in the sense of the lexicographic ordering as follows:

$$|\epsilon| = \epsilon, \qquad |a| = \begin{cases} |\mathsf{p}_\ell a| & \mathsf{l}_\mathsf{W}(\mathsf{p}_\ell a) = \mathsf{l}_\mathsf{W} a \\ \mathsf{s}_\ell(|\mathsf{p}_\ell a|) & \mathsf{p}_\ell a < a \end{cases}, \qquad |a| \leq a$$

We now assume

$$\epsilon \in X \wedge (\forall x \in \mathsf{W})(\mathsf{p}_\mathsf{W} x \in X \to x \in X) \tag{3.12}$$

Now we take some $a \in \mathsf{W}$ and aim at proving $a \in X$. Therefor, we first generate the type

$$Y := \{x : \mathsf{msp}a(|a| \dot{-} x) \mathrel{\dot{\in}} X\}$$

Y can obviously be constructed with axiom (inv) and we have

$$\Re(\mathsf{inv}((\lambda x.\mathsf{msp}a(|a| \dot{-} x)), X), Y)$$

$\epsilon \in Y$ as $\mathsf{msp}a(|a| \dot{-} \epsilon) = \mathsf{msp}a(|a|) = \epsilon$ and $\epsilon \in X$ by (3.12).

Now we assume $x \in Y$ and aim at showing $\mathsf{s}_\ell x \in Y$ which is equivalent to $\mathsf{msp}a(|a| \dot{-} \mathsf{s}_\ell x) \in X$. We have $\mathsf{msp}a(|a| \dot{-} \mathsf{s}_\ell x) = \mathsf{msp}a(\mathsf{p}_\ell(|a| \dot{-} x))$. Furthermore, $\mathsf{msp}a(|a| \dot{-} x) = \mathsf{p}_\mathsf{W}(\mathsf{msp}a(\mathsf{p}_\ell(|a| \dot{-} x)))$. Thus, by assumption (3.12) and induction hypothesis $\mathsf{msp}a(\mathsf{p}_\ell(|a| \dot{-} x)) \in X$. We can now apply ($\mathsf{T}$-$\mathsf{l}_\ell$) to get $(\forall x \in \mathsf{W})(x \in Y)$. From $a \in Y$ we immediately get $a \in X$ as $\mathsf{msp}a(|a| \dot{-} a) = \mathsf{msp}a\epsilon = a$. □

We are now ready to prove the upper bounds.

Theorem 3.28

1) PSET *is a conservative extension of* PS^- *for* $\mathcal{L}_{\mathsf{W}}$ *formulas.*

2) LSET *is a conservative extension of* LS^- *for* $\mathcal{L}_{\mathsf{W}}$ *formulas.*

Proof We again employ the same model theoretic argument and construct a model for PSET (LSET) from a model of PS^- (LS^-) using the recipe described in Section 3.3. Since we proved before for PET that types are weakly Σ^{b-}_{W} definable and since (Σ^{b-}_{W}-I_ℓ) holds in the original model, (T-I_ℓ) obviously holds in the thus constructed model. □

It immediately follows that the provably total functions of PSET coincide with the functions computable in polynomial space, whereas those of LSET coincide with the functions computable in linear space as desired.

Chapter 4

Disjoint Union And Realisability

This chapter mainly consists of a detailed discussion of the principle of disjoint union of types. The so-called Join Axioms add a type generator which constructs a type from a given type a and a function f mapping every element of the given type to a name. The join type consists of pairs (x, y) such that $x \mathrel{\dot\in} a$ and $y \mathrel{\dot\in} fx$. As already mentioned in Section 2.4, join has been studied over several theories, see e.g. Krähenbühl [44], Studer [58]. As summarised in Section 2.4, the addition of join does not increase the proof-theoretic strength of some theories, whereas others become stronger or even get inconsistent.

Our aim is to prove that the addition of join does not increase the proof-theoretic strength of PET. The weak theories we are studying in this thesis are more delicate than stronger systems and therefore the investigation of additional principles is more complicated. Techniques used to prove stronger theories proof-theoretically equivalent or tell them apart can not be easily applied in our case. Especially the model-theoretic argument employed in Section 3.3 to obtain the desired upper bounds for PET is not applicable in the presence of join. Thus, we will give a syntactic proof for the upper bounds by means of a realisability re-

lation for positive formulas. This argument is an extension of the one employed by Strahm in [53] for establishing the upper bounds for PT. A similar realisability relation was already used by Cantini in [5].

As already mentioned, the first systems of Explicit Mathematics were formulated in intuitionistic logic. We go back to the roots and formulate the theory PET+J based on intuitionistic logic in Section 4.1. Furthermore, we spell out a translation of PET+J in sequent calculus style. In Section 4.2 we construct in detail a model for PET+J based on the term model $\mathcal{M}(\lambda\eta)$ introduced in Section 2.3. Then we are ready to introduce a realisability relation for positive formulas in Section 4.3 and prove the realisability theorem stating that provable (positive) sequents are realisable by polynomial time computable functions in the sense of Theorem 4.15. Moreover, we are able to extend the realisability to treat also some of the extensions studied in Section 3.4, namely the uniformity principle and the type constructor for universal quantification from Section 3.4.1. Finally, in Section 4.5, we discuss problems arising from adding join to the classical version as well as the extent to which the methods used in this chapter also apply to the classical version of PET.

4.1 The Theory PET+J^i

In this section, we introduce a type constructor for disjoint union, the so-called join principle. As already mentioned, we change to intuitionistic logic. The exact reasons are given later in Section 4.5. To avoid confusion, theories based on intuitionistic logic will be denoted by a superscript T^i.

The theory introduced in this chapter is formulated in a language slightly different from before since this will simplify the realisability used later. To be precise, PET+J is formulated as a first-order axiom system where we only have a collection of names instead of types. We will proof that the original formulation can be translated into this version as we aim at giving a proof-theoretic analysis of the second-order variant of PET+J^i.

In the second part of this section, we reformulate our theory PET+J^i in sequent calculus style in order to state the realisability theorem later in this chapter.

4.1.1 Axiomatisation of PET+J^i

We now introduce the theory PET+J^i which is a first-order system. In this context, we only have names representing types instead of (second-order) types. Therefore, $\Re$ is in this context a unary relation symbol denoting the collection of names. The element relation is defined between two individuals where one of them is expected to be a name. Furthermore, the axioms about extensionality are dropped. This reformulation does not change the proof-theoretic strength as we will prove later.

PET+J is formulated in the language $\mathcal{L}^T_{\mathsf{W}}$ which extends $\mathcal{L}_{\mathsf{W}}$ by a **unary** relation symbol $\Re$, binary relation symbol $\dot\in$ and the (individual) constants w, id, dom, un, int, inv, and j (disjoint union). The relation $\dot\in$ connects a name with the elements of its extension. In contrast to the previous chapters, it is not an abbreviation, but a relation symbol of the language. Of course, the semantics of this relation shall match those of the abbreviation, as we will see in the translation below.

The theory PET+J consists of the axioms of BOW plus the following:

I. Type Existence Axioms

(w_a) $a \in \mathsf{W} \to \Re(\mathsf{w}(a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{w}(a) \leftrightarrow \mathsf{W}_a(x))$

(id) $\Re(\mathsf{id}) \wedge \forall x(x \mathrel{\dot\in} \mathsf{id} \leftrightarrow \exists y(x = (y, y)))$

(un) $\Re(a) \wedge \Re(b) \to \Re(\mathsf{un}(a, b)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{un}(a, b) \leftrightarrow (x \mathrel{\dot\in} a \vee x \mathrel{\dot\in} b))$

(int) $\Re(a) \wedge \Re(b) \to \Re(\mathsf{int}(a, b)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{int}(a, b) \leftrightarrow (x \mathrel{\dot\in} a \wedge x \mathrel{\dot\in} b))$

(dom) $\Re(a) \to \Re(\mathsf{dom}(a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{dom}(a) \leftrightarrow \exists y((x, y) \mathrel{\dot\in} a))$

(inv) $\Re(a) \to \Re(\mathsf{inv}(f, a)) \wedge \forall x(x \mathrel{\dot\in} \mathsf{inv}(f, a) \leftrightarrow fx \mathrel{\dot\in} a)$

(J.1) $\Re(a) \wedge (\forall x \mathrel{\dot\in} a)\Re(fx) \to \Re(\mathsf{j}(a, f))$

(J.2) $\Re(a) \wedge (\forall x \mathrel{\dot\in} a)\Re(fx) \to \forall x(x \mathrel{\dot\in} \mathsf{j}(a,f) \leftrightarrow \exists y \exists z(x = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy)$

II. Type Induction on W

(T-I$_\mathsf{W}$) $\Re(a) \wedge \epsilon \mathrel{\dot\in} a \wedge (\forall x \in \mathsf{W})(\mathsf{p}_\mathsf{W} x \mathrel{\dot\in} a \to x \mathrel{\dot\in} a) \to (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a)$

Let PET+J^i stand for the theory PET+J based on intuitionistic logic instead of classical logic.

In our present setting, we only have names and therefore loose the extensionality of types. Thus, we introduce a new abbreviation $\doteq$ for stating that two names are extensionally equal:

$$a \doteq b := \forall x(x \mathrel{\dot\in} a \leftrightarrow x \mathrel{\dot\in} b)$$

Since our main target is the proof-theoretic strength of the join axioms added to (the original formulation of) PET, we first have to translate formulas from the original version into first-order ones preserving provability. In the following, $\mathcal{L}^2_\mathsf{W}$ is assumed to include also the constant j.

Definition 4.1 (Translation from $\mathcal{L}^2_\mathsf{W}$ to $\mathcal{L}^T_\mathsf{W}$)
$\cdot^\star$ translates any $\mathcal{L}^2_\mathsf{W}$ formula A into a formula $A^\star$ of $\mathcal{L}^T_\mathsf{W}$. First, we need to add for every type variable X a new (individual) variable a_X. The translation is now defined by induction on the construction of A:

A **atomic:**

$$\begin{array}{lcl} A \equiv s = t \mid s{\downarrow} \mid \mathsf{W}(s) & \Longrightarrow & A^\star \equiv A \\ A \equiv X = Y & \Longrightarrow & A^\star \equiv a_X \doteq a_Y \\ A \equiv \Re(s, X) & \Longrightarrow & A^\star \equiv \Re(s) \wedge s \doteq a_X \\ A \equiv s \in X & \Longrightarrow & A^\star \equiv s \mathrel{\dot\in} a_X \end{array}$$

A **composite formula:**

$$\begin{array}{lcl} A \equiv B \wedge C & \Longrightarrow & A^\star \equiv B^\star \wedge C^\star \\ A \equiv B \vee C & \Longrightarrow & A^\star \equiv B^\star \vee C^\star \end{array}$$

$$
\begin{array}{lcll}
A \equiv B \to C & \Longrightarrow & A^\star \equiv B^\star \to C^\star & \\
A \equiv \neg B & \Longrightarrow & A^\star \equiv \neg B^\star & \\
A \equiv \forall x B & \Longrightarrow & A^\star \equiv \forall x B^\star & \\
A \equiv \exists x B & \Longrightarrow & A^\star \equiv \exists x B^\star & \\
A \equiv \forall X B & \Longrightarrow & A^\star \equiv \forall x(\Re(x) \to B^\star[x/a_X]) & \\
A \equiv \exists X B & \Longrightarrow & A^\star \equiv \exists x(\Re(x) \wedge B^\star[x/a_X]) & \circledast
\end{array}
$$

Let PET+J^2 stand for the theory PET augmented by the join axioms formulated in the original setting of $\mathcal{L}^2_{\mathsf{W}}$.

We are ready to state the equivalence of the two theories:

Theorem 4.2 *For any $\mathcal{L}^2_{\mathsf{W}}$ formula $A[\vec{X}]$ where $\vec{X}$ is a conclusive enumeration of* $\mathrm{FV}_T(A)$ *we have:*

$$\mathsf{PET{+}J}^2 \vdash A \quad \Longrightarrow \quad \mathsf{PET{+}J} \vdash \Re(\vec{a}_X) \to A^\star$$

Proof Before we start the actual proof, we will prove two properties of the translation to improve readability. Thereafter, the actual proof will be by induction on the proof length. First, we will prove that the shortcut $s \mathrel{\dot\in} t$ can be directly translated by $\Re(t) \wedge s \mathrel{\dot\in} t$: $s \mathrel{\dot\in} t \equiv \exists X(\Re(t, X) \wedge s \in X)$ in $\mathcal{L}^2_{\mathsf{W}}$ is translated into $\exists x(\Re(t) \wedge x \doteq t \wedge s \mathrel{\dot\in} x)$. With the definition of $\doteq$, we immediately get $\exists x(\Re(t) \wedge x \doteq t \wedge s \mathrel{\dot\in} t)$ which is provably equivalent to $\Re(t) \wedge s \mathrel{\dot\in} t$.

Second, we will prove that the second shortcut of PET+J^2, $\Re(t)$, can be translated as $\Re(t)$ in PET+J. $\Re(t) \equiv \exists X \Re(t, X)$ is translated to $\exists x(\Re(t) \wedge x \doteq t)$. Again, $\mathsf{PET{+}J} \vdash \exists x(\Re(t) \wedge x \doteq t)$ iff $\mathsf{PET{+}J} \vdash \Re(t)$.

After these two preparatory remarks, we can now start the actual proof. The theorem is proved by induction on the proof length. We start with showing that the translation of every axiom of PET+J^2 is provable in PET+J. Therein, the crucial part is the translation of the ontological axioms. The translations of the remaining axioms result more or less in the corresponding axioms of PET+J. As the first order part remains

untouched, the axioms of BOW stay the same and thus need not be considered.

Assume $\mathsf{PET{+}J}^2 \vdash A$ with proof length 0, that is A is one of the axioms:

Axiom of BOW $A^\star \equiv A$ as the first order part remains untouched. These axioms are identical in both theories.

(O.1) $A^\star \equiv \exists x(\Re(x) \wedge x \doteq a_X)$. $\mathsf{PET{+}J} \vdash \Re(a_X) \to \exists x(\Re(x) \wedge x \doteq a_X)$ obvious.

(O.2) $A^\star \equiv \Re(a) \wedge a \doteq a_X \wedge \Re(a) \wedge a \doteq a_Y \to a_X \doteq a_Y$. We directly get $\mathsf{PET{+}J} \vdash \Re(a_X) \wedge \Re(a_Y) \to (\Re(a) \wedge a = a_X \wedge a = a_Y \to a_X \doteq a_Y)$ from the definition of $\doteq$

(O.3) $A^\star \equiv a_X \doteq a_Y \leftrightarrow \forall x(x \mathrel{\dot\in} a_X \leftrightarrow x \mathrel{\dot\in} a_Y)$ This is obviously provable in PET+J, given the definition of $\doteq$.

(w_a) $A^\star \equiv \mathsf{W}(a) \to \Re(\mathsf{w}(a)) \wedge \forall x(\Re(\mathsf{w}(a)) \wedge x \mathrel{\dot\in} \mathsf{w}(a) \leftrightarrow x \leq a)$. Immediate with the corresponding axiom of PET+J.

(id), (un), (int), (dom), (inv) analogous.

(J.1) $A^\star \equiv \Re(a) \wedge \forall x(x \mathrel{\dot\in} a \to \Re(fx)) \to \Re(\mathsf{j}(a, f))$. Obvious.

(J.2) $A^\star \equiv \Re(a) \wedge (\forall x \mathrel{\dot\in} a)\Re(fx) \to \forall x(\Re(\mathsf{j}(a, f)) \wedge x \mathrel{\dot\in} \mathsf{j}(a, f) \leftrightarrow \exists y, z(x = (y, z) \wedge (\Re(a) \wedge y \mathrel{\dot\in} a) \wedge (\Re(fy) \wedge z \mathrel{\dot\in} fy))$. Immediate.

($\mathsf{T\text{-}I_W}$) $A^\star \equiv \epsilon \mathrel{\dot\in} a_X \wedge (\forall x \in \mathsf{W})(\mathsf{p_W}x \mathrel{\dot\in} a_X \to x \mathrel{\dot\in} a_X) \to (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a_X)$. $\mathsf{PET{+}J} \vdash \Re(a_X) \to A^\star$ obvious.

Now we consider Modus Ponens as an example of a rule. Assume we have $\mathsf{PET{+}J}^2 \vdash B[\vec{Y}]$ with length $n{+}1$ by Modus Ponens from $\mathsf{PET{+}J}^2 \vdash A[\vec{X}]$ and $\mathsf{PET{+}J}^2 \vdash A[\vec{X}] \to B[\vec{Y}]$, both with length $\leq n$. By induction hypothesis we have

$$\mathsf{PET{+}J} \vdash \Re(\vec{a}_X) \to A[\vec{a}_X] \tag{4.1}$$

$$\mathsf{PET{+}J} \vdash \Re(\vec{a}_X) \wedge \Re(\vec{a}_Y) \to (A[\vec{a}_X] \to B[\vec{a}_Y]) \tag{4.2}$$

From (4.1) we immediately get

$$\mathsf{PET{+}J} \vdash \Re(\vec{a}_X) \wedge \Re(\vec{a}_Y) \to A[\vec{a}_X] \tag{4.3}$$

From (4.2) and (4.3) we can derive

$$\mathsf{PET{+}J} \vdash \Re(\vec{a}_X) \wedge \Re(\vec{a}_Y) \to (B[\vec{a}_Y]) \tag{4.4}$$

The variables $\vec{a}_X$ in (4.4) are either contained in $\vec{a}_Y$ and therefore $\Re(a_X)$ need not be iterated, or they do not appear $B[\vec{a}_Y]$ and therefore we get

$$\mathsf{PET{+}J} \vdash \exists \vec{x} \Re(\vec{x}) \wedge \Re(\vec{a}_Y) \to (B[\vec{a}_Y]) \tag{4.5}$$

This is equivalent to

$$\mathsf{PET{+}J} \vdash \exists \vec{x} \Re(\vec{x}) \to \Re(\vec{a}_Y) \to (B[\vec{a}_Y]) \tag{4.6}$$

With e.g. (id) we have $\exists x \Re(x)$ and can now apply Modus Ponens to (4.6) to get

$$\mathsf{PET{+}J} \vdash \Re(\vec{a}_Y) \to (B[\vec{a}_Y]) \tag{4.7}$$

The other rules can be treated similarly. □

As we will spell out some formal proofs in detail, we will employ some auxiliary rules to improve readability.

Remark 4.3 (Auxiliary Rules) The following "Auxiliary Rules" are admissible:

1. Transitivity of implication is provable: If $A \to B$ and $B \to C$ were derived by a Hilbert proof, then $A \to C$ is derivable:

1.	$A \to B$	By Assumption
2.	$B \to C$	By Assumption
3.	$(B \to C) \to (A \to (B \to C))$	Axiom
4.	$A \to (B \to C)$	MP on 2 + 3
5.	$(A \to (B \to C))$ $\to ((A \to B) \to (A \to C))$	Axiom

6. $(A \to B) \to (A \to C)$ MP on 4 + 5
7. $A \to C$ MP on 1 + 7

2. If $\vdash A \to B$ and $\vdash A \to C$, then $\vdash A \to (B \wedge C)$.

3. $\vdash A \wedge B \to C \iff \vdash A \to (B \to C)$.

4.1.2 Sequent Calculus Reformulation

As previously announced, we reformulate the theory in sequent calculus style as the realisability relation works in this setting. We are working in the same language $\mathcal{L}^T_{\mathsf{W}}$ as before and make use of the same abbreviations. The theory $\mathsf{PET{+}J}^{iG}$ is the reformulation of the theory $\mathsf{PET{+}J}^{i}$ in Gentzen style based on intuitionistic sequent calculus **G2i** where $\neg A := A \to \bot$. The axioms of the first order part are adopted from [53] and only presented partially. In the following, we will write Γ and Δ for a (fixed) listing of formulas $A_0, \ldots, A_n$.

$\mathsf{PET{+}J}^{iG}$ consists of the following parts axioms and rules:

I. Logical Axioms

$$P, \Gamma \Rightarrow P \quad (P \text{ atomic}) \qquad\qquad \bot, \Gamma \Rightarrow C$$

II. Logical Rules

$$L\wedge \frac{\Gamma, A, B \Rightarrow C}{\Gamma, A \wedge B \Rightarrow C} \qquad R\wedge \frac{\Gamma \Rightarrow A \quad \Gamma \Rightarrow B}{\Gamma \Rightarrow A \wedge B}$$

$$L\vee \frac{\Gamma, A \Rightarrow C \quad \Gamma, B \Rightarrow C}{\Gamma, A \vee B \Rightarrow C} \qquad R\vee \frac{\Gamma \Rightarrow A_i}{\Gamma \Rightarrow A_0 \vee A_1}$$

$$L\to \frac{\Gamma \Rightarrow A \quad \Gamma, B \Rightarrow C}{\Gamma, A \to B \Rightarrow C} \qquad R\to \frac{\Gamma, A \Rightarrow B}{\Gamma \Rightarrow A \to B}$$

$$L\forall\ \frac{\Gamma, A[t] \Rightarrow C}{\Gamma, \forall x A[x] \Rightarrow C} \qquad R\forall\ \frac{\Gamma \Rightarrow A[y]}{\Gamma \Rightarrow \forall x A[x]}\ ^*$$

$$L\exists\ \frac{\Gamma, A[y] \Rightarrow C}{\Gamma, \exists x A[x] \Rightarrow C}\ ^* \qquad R\exists\ \frac{\Gamma \Rightarrow A[t]}{\Gamma \Rightarrow \exists x A[x]}$$

$$\text{Cut}\ \frac{\Gamma \Rightarrow A \quad \Gamma, A \Rightarrow C}{\Gamma \Rightarrow C}$$

*: y not free in Γ, C

III. Structural Rules

$$ex\ \frac{\Gamma, A, B, \Gamma' \Rightarrow C}{\Gamma, B, A, \Gamma' \Rightarrow C} \qquad contr\ \frac{\Gamma, A, A \Rightarrow C}{\Gamma, A \Rightarrow C}$$

IV. BOW^S We will only spell out some examples, the rest is analogous:

(4) $\Gamma, \mathsf{W}(t_0), \mathsf{W}(t_1), t_0 = t_1 \Rightarrow \mathsf{d_W} r_0 r_1 t_0 t_1 = r_0$

(5) $\Gamma, \mathsf{W}(t_0), \mathsf{W}(t_1) \Rightarrow t_0 = t_1 \vee \mathsf{d_W} r_0 r_1 t_0 t_1 = r_1$

(6.1) $\Gamma \Rightarrow \mathsf{W}(\epsilon)$

(6.2) $\Gamma, \mathsf{W}(t) \Rightarrow \mathsf{W}(\mathsf{s}_0 t) \wedge \mathsf{W}(\mathsf{s}_1 t)$

(Eq) $\Gamma, \Re(t), t = s \Rightarrow \Re(s)$

V. Type existence axioms

(w_a.1) $\Gamma, \mathsf{W}(s) \Rightarrow \Re(\mathsf{w}(s))$

(w_a.2) $\Gamma, \mathsf{W}(s), W(t), t \leq s \Rightarrow t \mathrel{\dot\in} \mathsf{w}(s)$

(w_a.3) $\Gamma, \mathsf{W}(s), t \mathrel{\dot\in} \mathsf{w}(s) \Rightarrow \mathsf{W}(t) \wedge t \leq s$

(id.1) $\Gamma \Rightarrow \Re(\mathsf{id})$

(id.2) $\Gamma, \exists u(t = (u, u)) \Rightarrow t \mathrel{\dot\in} \mathsf{id}$

(id.3) $\Gamma, t \mathrel{\dot\in} \mathsf{id} \Rightarrow \exists u(t = (u, u))$

(inv.1) $\Gamma, \Re(s) \Rightarrow \Re(\mathsf{inv}(r, s))$

(inv.2) $\Gamma, \Re(s), rt \mathrel{\dot\in} s \Rightarrow t \mathrel{\dot\in} \mathsf{inv}(r, s)$

(inv.3)	$\Gamma, \Re(s), t \mathrel{\dot\in} \mathsf{inv}(r,s) \Rightarrow rt \mathrel{\dot\in} s$
(dom.1)	$\Gamma, \Re(s) \Rightarrow \Re(\mathsf{dom}(s))$
(dom.2)	$\Gamma, \Re(s), \exists y((t,y) \mathrel{\dot\in} s) \Rightarrow t \mathrel{\dot\in} \mathsf{dom}(s)$
(dom.3)	$\Gamma, \Re(s), t \mathrel{\dot\in} \mathsf{dom}(s) \Rightarrow \exists y((t,y) \mathrel{\dot\in} s)$
(un.1)	$\Gamma, \Re(s_0), \Re(s_1) \Rightarrow \Re(\mathsf{un}(s_0,s_1))$
(un.2)	$\Gamma, \Re(s_0), \Re(s_1), t \mathrel{\dot\in} s_0 \Rightarrow t \mathrel{\dot\in} \mathsf{un}(s_0,s_1)$
(un.3)	$\Gamma, \Re(s_0), \Re(s_1), t \mathrel{\dot\in} s_1 \Rightarrow t \mathrel{\dot\in} \mathsf{un}(s_0,s_1)$
(un.4)	$\Gamma, \Re(s_0), \Re(s_1), t \mathrel{\dot\in} \mathsf{un}(s_0,s_1) \Rightarrow t \mathrel{\dot\in} s_0 \vee t \mathrel{\dot\in} s_1$
(int.1)	$\Gamma, \Re(s_0), \Re(s_1) \Rightarrow \Re(\mathsf{int}(s_0,s_1))$
(int.2)	$\Gamma, \Re(s_0), \Re(s_1), t \mathrel{\dot\in} s_0, t \mathrel{\dot\in} s_1 \Rightarrow t \mathrel{\dot\in} \mathsf{int}(s_0,s_1)$
(int.3)	$\Gamma, \Re(s_0), \Re(s_1), t \mathrel{\dot\in} \mathsf{int}(s_0,s_1) \Rightarrow t \mathrel{\dot\in} s_0$
(int.4)	$\Gamma, \Re(s_0), \Re(s_1), t \mathrel{\dot\in} \mathsf{int}(s_0,s_1) \Rightarrow t \mathrel{\dot\in} s_1$

VI. Join Rules

$$(\mathsf{J.1})\ \frac{\Gamma, x \mathrel{\dot\in} s \Rightarrow \Re(rx) \qquad \Gamma \Rightarrow \Re(s)}{\Gamma \Rightarrow \Re(\mathsf{j}(s,r))}\ *$$

$$(\mathsf{J.2})\ \frac{\Gamma, x \mathrel{\dot\in} s \Rightarrow \Re(rx) \qquad \Gamma \Rightarrow \Re(s)}{\Gamma, t \mathrel{\dot\in} \mathsf{j}(s,r) \Rightarrow t = (t_0,t_1) \wedge t_0 \mathrel{\dot\in} s \wedge t_1 \mathrel{\dot\in} rt_0)}\ *$$

$$(\mathsf{J.3})\ \frac{\Gamma, x \mathrel{\dot\in} s \Rightarrow \Re(rx) \qquad \Gamma \Rightarrow \Re(s)}{\Gamma, t = (t_0,t_1), t_0 \mathrel{\dot\in} s, t_1 \mathrel{\dot\in} rt_0 \Rightarrow t \mathrel{\dot\in} \mathsf{j}(s,r)}\ *$$

VII. Type Induction

$$(\mathsf{T\text{-}I_W})\ \frac{\Gamma \Rightarrow \Re(s) \qquad \Gamma \Rightarrow \epsilon \mathrel{\dot\in} s \qquad \Gamma, \mathsf{W}(x), x \mathrel{\dot\in} s \Rightarrow \mathsf{s}_i x \mathrel{\dot\in} s}{\Gamma, \mathsf{W}(t) \Rightarrow t \mathrel{\dot\in} s}\ {}_{i=0,1}\ *$$

*: x not free in Γ

In the following, we let $\bigwedge \Gamma$ abbreviate $A_0 \wedge \cdots \wedge A_n$ for $\Gamma = A_0, \ldots A_n$. We now prove that this reformulation is indeed adequate:

Theorem 4.4 (Equivalence of PET+J^i and PET+J^{iG}) *For all formulas C and all sequents $\Gamma \Rightarrow C$ we have*

1) $\mathsf{PET{+}J}^i \vdash C \implies \mathsf{PET{+}J}^{iG} \vdash \Rightarrow C$

2) $\mathsf{PET{+}J}^{iG} \vdash \Gamma \Rightarrow C \implies \mathsf{PET{+}J}^i \vdash \bigwedge \Gamma \to C$

Proof Both statements are proved by induction on proof height. The logical axioms and rules will be omitted as this part proceeds as usual.

1. Assume $\mathsf{PET{+}J}^i \vdash C$ with a proof of height n.

   $n = 0$ If C is an axiom of BOW, the proof was already done in [53]. In case C is one of the axioms (w_a), (id), (inv), (un), or (dom), $\Rightarrow C$ can obviously be derived from the axioms of group V.

   In the following formal proofs, structural rules are omitted. Shortcuts are translated back on the fly when needed, indicated by * on the rule name.

   If C is an instance of (J.1), the proof is given in Fig. 4.1.

$$L{\to}\,\frac{\Gamma, y \mathrel{\dot\in} a \to \Re(fy), \underline{y \mathrel{\dot\in} a} \Rightarrow \underline{y \mathrel{\dot\in} a} \quad \Gamma, \underline{\Re(fy)}, y \mathrel{\dot\in} a \Rightarrow \underline{\Re(fy)}}{\overbrace{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), y \mathrel{\dot\in} a \to \Re(fy), y \mathrel{\dot\in} a \Rightarrow \Re(fy)}^{\Gamma}}\,L{\forall}^*$$

$$(\mathsf{J.1})\,\frac{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), y \mathrel{\dot\in} a \Rightarrow \Re(fy) \qquad \underline{\Re(a)}, (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow \underline{\Re(a)}}{R{\to}\,\dfrac{L{\wedge}\,\dfrac{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow \Re(\mathsf{j}(a,f))}{\Re(a) \wedge (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow \Re(\mathsf{j}(a,f))}}{\Rightarrow \Re(a) \wedge (\forall x \mathrel{\dot\in} a)\Re(fx) \to \Re(\mathsf{j}(a,f))}}$$

Figure 4.1: Proof of $\mathsf{PET{+}J}^{iG} \vdash \Rightarrow$ (J.1)

   If C is an instance of (J.2), the proof is given in Fig. 4.2.

   If C is an instance of ($\mathsf{T\text{-}I_W}$), the proof is given in Fig. 4.3.

   $n > 0$ In this case, C was derived by a logical rule and the proof is as usual.

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Analogous (J.1)}}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), w \mathrel{\dot\in} a \Rightarrow \Re(fw)} \quad \underline{\Re(a)}, (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow \underline{\Re(a)}}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), v = (y,z), y \mathrel{\dot\in} a, z \mathrel{\dot\in} fy \Rightarrow v \mathrel{\dot\in} \mathsf{j}(a,f)}\,(\mathsf{J}.3)}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), v = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy \Rightarrow v \mathrel{\dot\in} \mathsf{j}(a,f)}\,2\times L\wedge}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), \exists y \exists z (v = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy) \Rightarrow v \mathrel{\dot\in} \mathsf{j}(a,f)}\,2\times L\exists}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow \exists y \exists z (v = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy) \to v \mathrel{\dot\in} \mathsf{j}(a,f)}\,R{\to}
$$

continued below

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Analogous (J.1)}}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), y \mathrel{\dot\in} a \Rightarrow \Re(fy)} \quad \underline{\Re(a)}, (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow \underline{\Re(a)}}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx), v \mathrel{\dot\in} \mathsf{j}(a,f) \Rightarrow \exists y \exists z (v = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy)}\,(\mathsf{J}.2)}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow v \mathrel{\dot\in} \mathsf{j}(a,f) \to \exists y \exists z (v = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy)}\,R{\to} \quad \text{see above}}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow v \mathrel{\dot\in} \mathsf{j}(a,f) \leftrightarrow \exists y \exists z (v = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy)}\,R{\wedge}^*}{\Re(a), (\forall x \mathrel{\dot\in} a)\Re(fx) \Rightarrow \forall x (x \mathrel{\dot\in} \mathsf{j}(a,f) \leftrightarrow \exists y \exists z (x = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy))}\,R\forall}{\Rightarrow \Re(a) \wedge (\forall x \mathrel{\dot\in} a)\Re(fx) \to \forall x (x \mathrel{\dot\in} \mathsf{j}(a,f) \leftrightarrow \exists y \exists z (x = (y,z) \wedge y \mathrel{\dot\in} a \wedge z \mathrel{\dot\in} fy))}\,R{\to}, L{\wedge}
$$

Figure 4.2: Proof of $\mathsf{PET{+}J}^{iG} \vdash \Rightarrow (\mathsf{J}.2)$

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\Gamma, y \in \mathsf{W}, \underline{y \mathrel{\dot\in} a} \Rightarrow \underline{y \mathrel{\dot\in} a}}
\quad
L\wedge\ \dfrac{\Gamma, \underline{\mathsf{s}_0 y \mathrel{\dot\in} a}, \underline{\mathsf{s}_1 y \mathrel{\dot\in} a}, y \in \mathsf{W}, y \mathrel{\dot\in} a \Rightarrow \underline{\mathsf{s}_i y \mathrel{\dot\in} a}}{\Gamma, \mathsf{s}_0 y \mathrel{\dot\in} a \wedge \mathsf{s}_1 y \mathrel{\dot\in} a, y \in \mathsf{W}, y \mathrel{\dot\in} a \Rightarrow \mathsf{s}_i y \mathrel{\dot\in} a}
}{
L\to\ \dfrac{\Gamma, \underline{y \in \mathsf{W}}, y \mathrel{\dot\in} a \Rightarrow \underline{y \in \mathsf{W}}, y \mathrel{\dot\in} a \to \mathsf{s}_0 y \mathrel{\dot\in} a \wedge \mathsf{s}_1 y \mathrel{\dot\in} a, y \in \mathsf{W}, y \mathrel{\dot\in} a \Rightarrow \mathsf{s}_i y \mathrel{\dot\in} a}{\Gamma, y \in \mathsf{W} \to (y \mathrel{\dot\in} a \to \mathsf{s}_0 y \mathrel{\dot\in} a \wedge \mathsf{s}_1 y \mathrel{\dot\in} a), y \in \mathsf{W}, y \mathrel{\dot\in} a \Rightarrow \mathsf{s}_i y \mathrel{\dot\in} a}
}{
L\forall^*\ \dfrac{\overbrace{\Gamma}}{\overbrace{\Re(a), \epsilon \mathrel{\dot\in} a, (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a)}, y \in \mathsf{W}, y \mathrel{\dot\in} a \Rightarrow \mathsf{s}_i y \mathrel{\dot\in} a}
}{
\text{continued below}
}
$$

$$
(\mathsf{T}\text{-}\mathsf{I}_\mathsf{W})\ \dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\underline{\Re(a)}, \epsilon \mathrel{\dot\in} a, (\forall x \in \mathsf{W})(\dots) \Rightarrow \underline{\Re(a)}
\quad
\Re(a), \underline{\epsilon \mathrel{\dot\in} a}, (\forall x \in \mathsf{W})(\dots) \Rightarrow \underline{\epsilon \mathrel{\dot\in} a}
\quad \text{See above}
}{
\Re(a), \epsilon \mathrel{\dot\in} a, (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a), y \in \mathsf{W} \Rightarrow y \mathrel{\dot\in} a)
}
}{
R\to\ \dfrac{\Re(a), \epsilon \mathrel{\dot\in} a, (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a) \Rightarrow y \in \mathsf{W} \to y \mathrel{\dot\in} a)}{\Re(a), \epsilon \mathrel{\dot\in} a, (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a) \Rightarrow (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a)}
}{
R\forall^*\ \dfrac{\Re(a), \epsilon \mathrel{\dot\in} a, (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a) \Rightarrow (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a)}{\Re(a) \wedge \epsilon \mathrel{\dot\in} a \wedge (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a) \Rightarrow (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a)}
}{
2 \times L\wedge\ \dfrac{\Re(a) \wedge \epsilon \mathrel{\dot\in} a \wedge (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a) \Rightarrow (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a)}{\Rightarrow \Re(a) \wedge \epsilon \mathrel{\dot\in} a \wedge (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a \to \mathsf{s}_0 x \mathrel{\dot\in} a \wedge \mathsf{s}_1 x \mathrel{\dot\in} a) \to (\forall x \in \mathsf{W})(x \mathrel{\dot\in} a)}
}{
R\to
}
$$

Figure 4.3: Proof of $\mathsf{PET{+}J}^{iG} \vdash \Rightarrow (\mathsf{T}\text{-}\mathsf{I}_\mathsf{W})$

2. Assume $\mathsf{PET{+}J}^{iG} \vdash \Gamma \Rightarrow C$ with a proof of height n. For $n = 0$, $\Gamma \Rightarrow C$ is an axiom. Axioms of group V. can obviously be proved from their corresponding axioms in $\mathsf{PET{+}J}^{i}$. If $n > 0$, we need to consider only the rules from groups VI. and VII., since the proof for the logic rules proceeds as usual.

   Assume $\Gamma \Rightarrow \Re(\mathsf{j}(a, f))$ was derived by (J.1). By induction hypothesis, there are proofs in $\mathsf{PET{+}J}^{i}$ for the premises of the rule:

1.	$\Re(t) \wedge (\forall x \mathrel{\dot\in} t)\Re(fx) \to \Re(\mathsf{j}(t, f))$	Axiom (J.1)
2.	$\bigwedge \Gamma \wedge x \mathrel{\dot\in} t \to \Re(fx)$	Induction Hypothesis
3.	$\bigwedge \Gamma \to \Re(t)$	Induction Hypothesis
4.	$\bigwedge \Gamma \to (x \mathrel{\dot\in} t \to \Re(fx))$	Remark 4.3.3 on 2
5.	$\bigwedge \Gamma \to \forall x(x \mathrel{\dot\in} t \to \Re(fx))$	$\forall$-GEN
6.	$\bigwedge \Gamma \to \Re(t) \wedge \forall x(x \mathrel{\dot\in} t \to \Re(fx))$	Remark 4.3.2 on 3, 5
7.	$\bigwedge \Gamma \to \Re(\mathsf{j}(t, f))$	Remark 4.3.1 on 6, 1

   The proofs for the rules (J.2+3) are similar and therefore left out.

   We now have to consider the rule ($\mathsf{T\text{-}I_W}$). Assume $\Gamma, u \in \mathsf{W} \Rightarrow u \mathrel{\dot\in} t$ was derived by an application of ($\mathsf{T\text{-}I_W}$):

1.	$\bigwedge \Gamma \to \Re(t)$	Induction Hypothesis
2.	$\bigwedge \Gamma \to \epsilon \mathrel{\dot\in} t$	Induction Hypothesis
3.	$\bigwedge \Gamma \wedge u \in \mathsf{W} \wedge u \mathrel{\dot\in} t \to \mathsf{s}_0 u \mathrel{\dot\in} t$	Induction Hypothesis
4.	$\bigwedge \Gamma \wedge u \in \mathsf{W} \wedge u \mathrel{\dot\in} t \to \mathsf{s}_1 u \mathrel{\dot\in} t$	Induction Hypothesis
5.	$\bigwedge \Gamma \wedge u \in \mathsf{W} \wedge u \mathrel{\dot\in} t \to (\mathsf{s}_0 u \mathrel{\dot\in} t \wedge \mathsf{s}_1 u \mathrel{\dot\in} t)$	Remark 4.3.2 on 3, 4
6.	$\bigwedge \Gamma \wedge u \in \mathsf{W} \to (u \mathrel{\dot\in} t \to (\mathsf{s}_0 u \mathrel{\dot\in} t \wedge \mathsf{s}_1 u \mathrel{\dot\in} t))$	Remark 4.3.3 on 5
7.	$\bigwedge \Gamma \to (u \in \mathsf{W} \to (u \mathrel{\dot\in} t \to (\mathsf{s}_0 u \mathrel{\dot\in} t \wedge \mathsf{s}_1 u \mathrel{\dot\in} t)))$	Remark 4.3.3 on 6
8.	$\bigwedge \Gamma \to \forall x(x \in \mathsf{W} \to (x \mathrel{\dot\in} t \to (\mathsf{s}_0 x \mathrel{\dot\in} t \wedge \mathsf{s}_1 x \mathrel{\dot\in} t)))$	$\forall$-GEN on 7
9.	$\bigwedge \Gamma \to \Re(t) \wedge \epsilon \mathrel{\dot\in} t \wedge \forall x(x \in \mathsf{W} \to (x \mathrel{\dot\in} t \to (\mathsf{s}_0 x \mathrel{\dot\in} t \wedge \mathsf{s}_1 x \mathrel{\dot\in} t)))$	Remark 4.3.2 on 1, 2, 8

10.	$(\mathsf{T}\text{-}\mathsf{I}_\mathsf{W})$	Axiom
11.	$\bigwedge \Gamma \to \forall x(x \in \mathsf{W} \to x \mathrel{\dot\in} t)$	Remark 4.3.1 on 9, 10
12.	$\forall x(x \in \mathsf{W} \to x \mathrel{\dot\in} t)$ $\to (u \in \mathsf{W} \to u \mathrel{\dot\in} t)$	Axiom
13.	$\bigwedge \Gamma \to (u \in \mathsf{W} \to u \mathrel{\dot\in} t)$	Remark 4.3.1 on 11, 12
14.	$\bigwedge \Gamma \wedge u \in \mathsf{W} \to u \mathrel{\dot\in} t$	Remark 4.3.3 on 13

This ends our proof. □

For the realisability, we depend on partial cut elimination, i.e. only cuts with positive formulas are allowed in our proofs. Therefore, we first need to define the cut rank of a formula. It is defined such that positive formulas have cut rank 0 and non-positive formulas have usual cut rank.

Definition 4.5
The cut rank $\mathrm{rk}(A)$ of a formula A is defined by a case distinction:

$$\mathrm{rk}(A) = \begin{cases} 0 & A \text{ positive} \\ \mathrm{rk}'(A) & \text{otherwise} \end{cases}$$

where $\mathrm{rk}'(A)$ is defined as follows:

1) $A \equiv B \wedge C$ or $B \vee C$: $\mathrm{rk}'(A) = \max(\mathrm{rk}(B), \mathrm{rk}(C)) + 1$
2) $A \equiv \forall x B$ or $\exists x B$: $\mathrm{rk}'(A) = \mathrm{rk}(B) + 1$
3) $A \equiv B \to C$: $\mathrm{rk}'(A) = \max(\mathrm{rk}(B), \mathrm{rk}(C)) + 1$ ⊛

The proof height of a proof is defined as usual: the height is 0 if the proof is an axiom and $\max(n_0, \ldots, n_i) + 1$ for a proof where the last rule has i premises and premise j has proof height n_j, except for structural rules.

Definition 4.6
$\mathsf{PET}\text{+}\mathsf{J}^{iG} \mid\!\!\frac{n}{r}\ \Gamma \Rightarrow C$ is defined by induction on the proof structure:

1) $0 \leq n, r$ and $\Gamma \Rightarrow C$ is an axiom.

2) $\Gamma \Rightarrow C$ was derived by a structural rule from $\Gamma_1 \Rightarrow C$ and $\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n}{r}$ $\Gamma_1 \Rightarrow C$

3) $\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n}{r}$ $\Gamma \Rightarrow C$ if the last rule was the cut rule and $\mathrm{rk}(A) = r_A$ and $\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n_0}{r_0}$ $\Gamma \Rightarrow A$ and $\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n_1}{r_1}$ $\Gamma, A \Rightarrow C$ where $n_0, n_1 < n$ and $r \geq \max(r_A + 1, r_0, r_1)$

4) $\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n}{r}$ $\Gamma \Rightarrow C$ if the last rule was one of the other rules with k premises and $\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n_i}{r_i}$ $\Gamma_i \Rightarrow C_i$ $(i = 0, \ldots, k)$ and $r \geq \max(r_i)$ and $n > \max(n_i)$

n is called the proof height and r the cut rank of the proof.

Furthermore, we write $\mathsf{PET}{+}\mathsf{J}^{iG} \vdash_r \Gamma \Rightarrow C$ if there exists a n such that $\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n}{r}$ $\Gamma \Rightarrow C$ ⊛

Lemma 4.7 (Admissibility of Weakening)
Weakening is height-preserving admissible in $\mathsf{PET}{+}\mathsf{J}^{iG}$*:*

$$\mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n}{r} \Gamma \Rightarrow C \qquad \Longrightarrow \qquad \mathsf{PET}{+}\mathsf{J}^{iG} \mid\!\!\frac{n}{r} \Gamma, \Gamma' \Rightarrow C$$

After this preparatory work, we are now ready to establish partial cut elimination:

Theorem 4.8 (Partial Cut Elimination)
If $\mathsf{PET}{+}\mathsf{J}^{iG} \vdash_r \Gamma \Rightarrow C$*, then also* $\mathsf{PET}{+}\mathsf{J}^{iG} \vdash_1 \Gamma \Rightarrow C$

This theorem follows immediately from the following lemma:

Lemma 4.9 (Partial Cut Reduction) *If* $\mathsf{PET}{+}\mathsf{J}^{iG} \vdash_r \Gamma \Rightarrow A$ *and* $\mathsf{PET}{+}\mathsf{J}^{iG} \vdash_r \Gamma', A \Rightarrow C$ *with* $\mathrm{rk}(A) = r > 0$*, then* $\mathsf{PET}{+}\mathsf{J}^{iG} \vdash_r \Gamma, \Gamma' \Rightarrow C$

Proof As the main formulas of all non-logical axioms and rules are positive, the proof is analogous to the usual proof as spelled out in detail for example in [60]. □

4.2 A Model for PET+J^i

In this section, we elaborately describe a model for PET+J^i based on the term model $\mathcal{M}(\lambda\eta)$ introduced in Section 2.3. The construction is similar to the one presented in Section 3.3, but the stages for the construction of the interpretation of the names run over all ordinals and not just over the natural numbers as before. This expansion is necessary because of the join constructor which allows references to uniformly generated names.

First, we give a definition of a $\mathcal{L}^T_{\mathsf{W}}$-structure. In contrast to models of PET, we only need interpretations for names and elementship.

Definition 4.10 ($\mathcal{L}^T_{\mathsf{W}}$-Structure)
A $\mathcal{L}^T_{\mathsf{W}}$-structure $\mathcal{M}^\star$ is a tuple

$$(\mathcal{M}, \mathcal{R}, \mathcal{E}, \underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{dom}}, \underline{\mathsf{inv}}, \underline{\mathsf{j}})$$

with the following conditions:

(i) $\mathcal{M}$ is a $\mathcal{L}_{\mathsf{W}}$-structure,
(ii) $\mathcal{R}$ is a non-empty subset of $|\mathcal{M}|$,
(iii) $\mathcal{E}$ is a binary relation on $|\mathcal{M}| \times \mathcal{R}$, and
(iv) $\underline{\mathsf{w}}, \underline{\mathsf{id}}, \underline{\mathsf{un}}, \underline{\mathsf{int}}, \underline{\mathsf{dom}}, \underline{\mathsf{inv}}, \underline{\mathsf{j}}$ are elements of $|\mathcal{M}|$. ⊛

For the construction we take the model $\mathcal{M}(\lambda\eta)$ of PT as introduced in Section 2.3. Recall that we have usual $\beta\eta$-reduction adapted to fit our axioms and $|\mathcal{M}| = \{t : t \text{ term}\}$ and the constants are interpreted by themselves. We write $t \overset{\text{red}}{\Longrightarrow} s$ for regular reduction of terms and define the abbreviation $\overset{\beta\eta}{=}$ as $t_1 \overset{\beta\eta}{=} t_2 :\Longleftrightarrow t_1 \overset{\text{red}}{\Longrightarrow} s$ and $t_2 \overset{\text{red}}{\Longrightarrow} s$ for some term s.

Now we generate the model $\mathcal{M}^\star$ of PET+J by adding interpretations $\mathcal{R}$ and $\mathcal{E}$ for $\Re$ and $\dot\in$ respectively. For the construction of $\mathcal{R}$ we introduce sets $\mathcal{R}_\alpha \subseteq |\mathcal{M}^\star|$ by induction on the ordinal α and simultaneously

establish a set $\mathcal{E}_\alpha \subseteq |\mathcal{M}| \times \mathcal{R}_\alpha$. Then we set

$$\mathcal{M}^\star_\alpha := (\mathcal{M}(\lambda\eta), \mathcal{R}_\alpha, \mathcal{E}_\alpha, \mathsf{w}, \mathsf{id}, \mathsf{un}, \mathsf{int}, \mathsf{dom}, \mathsf{inv}, \mathsf{j}).$$

For every ordinal number α, $\mathcal{R}_\alpha$ and $\mathcal{E}_\alpha$ are constructed as follows where $r, s, t \in |\mathcal{M}(\lambda\eta)|$.

$\alpha = 0$**:** $\mathcal{R}_0$ contains the names of the base types, i.e. formally $s \in \mathcal{R}_0$ iff

- $s \overset{\beta\eta}{=} \mathsf{id}$ and $(t, s) \in \mathcal{E}_0$ iff $t \overset{\beta\eta}{=} (m, m)$ for some $m \in |\mathcal{M}(\lambda\eta)|$.
- $s \overset{\beta\eta}{=} \mathsf{w}a$ with $a \in \mathsf{W}^{\mathcal{M}^\star}$ and $(t, s) \in \mathcal{E}_0$ iff $\mathcal{M}^\star \models t \in \mathsf{W} \wedge t \leq a$.

$\alpha' = \alpha + 1$**:** $\mathcal{R}_{\alpha'}$ contains $\mathcal{R}_\alpha$. In addition, for $s_0, s_1 \in \mathcal{R}_\alpha$, $s \in \mathcal{R}_{\alpha'}$ iff

- $s \overset{\beta\eta}{=} \mathsf{un}(s_0, s_1)$ and $(t, s) \in \mathcal{E}_{\alpha'}$ iff $(t, s_0) \in \mathcal{E}_\alpha$ or $(t, s_1) \in \mathcal{E}_\alpha$.
- $s \overset{\beta\eta}{=} \mathsf{int}(s_0, s_1)$ and $(t, s) \in \mathcal{E}_{\alpha'}$ iff $(t, s_0) \in \mathcal{E}_\alpha$ and $(t, s_1) \in \mathcal{E}_\alpha$.
- $s \overset{\beta\eta}{=} \mathsf{dom}(s_0)$ and $(t, s) \in \mathcal{E}_{\alpha'}$ iff there is a $m \in |\mathcal{M}^\star|$ such that $((t, m), s_0) \in \mathcal{E}_\alpha$.
- $s \overset{\beta\eta}{=} \mathsf{inv}(r, s_0)$ and $(t, s) \in \mathcal{E}_{\alpha'}$ iff $(rt, s_0) \in \mathcal{E}_\alpha$
- $s \overset{\beta\eta}{=} \mathsf{j}(s_0, r)$ and $rt \in \mathcal{R}_\alpha$ for all t such that $(t, s_0) \in \mathcal{E}_\alpha$. Furthermore, $(t, s) \in \mathcal{E}_{\alpha'}$ iff $t \overset{\beta\eta}{=} (m, n)$ such that $(m, s_0) \in \mathcal{E}_\alpha$ and $(n, rm) \in \mathcal{E}_\alpha$.

$\alpha =$ **limit ordinal:** $\mathcal{R}_\alpha = \bigcup_{\beta<\alpha} \mathcal{R}_\beta$ and $\mathcal{E}_\alpha = \bigcup_{\beta<\alpha} \mathcal{E}_\beta$.

Finally, we define $\mathcal{R} := \bigcup_{\alpha\in\Omega} \mathcal{R}_\alpha$ and $\mathcal{E} := \bigcup_{\alpha\in\Omega} \mathcal{E}_\alpha$ where Ω stands for the ordinals. Then our desired $\mathcal{L}^T_{\mathsf{W}}$ structure is given by

$$\mathcal{M}^\star \;:=\; (\mathcal{M}(\lambda\eta), \mathcal{R}, \mathcal{E}, \mathsf{w}, \mathsf{id}, \mathsf{un}, \mathsf{int}, \mathsf{dom}, \mathsf{inv}, \mathsf{j}).$$

For any $s \in \mathcal{R}_\alpha$, we define $ext(s) := \{t \in |\mathcal{M}^\star| : (t, s) \in \mathcal{E}_\alpha\}$. In the following, we use the abbreviation $t \; \varepsilon_\alpha \; s := (t, s) \in \mathcal{E}_\alpha$.

It immediately follows from Theorem 3.14 that $\mathcal{M}^\star$ indeed is a model for $\mathsf{PET{+}J}^i$. When referencing $\mathcal{M}^\star$ from now on, we always relate to the specific model of $\mathsf{PET{+}J}^i$ as constructed here, unless explicitly stated otherwise.

4.3 Realisability for Positive Formulas

In this section we define the notion of realisability of positive formulas. Realisers are binary words and shall contain some computational content. They can be seen as witnesses for the statement of the formula. The definition in this thesis is an extension of the one introduced in Strahm [53]. The notion of realisers of $\mathcal{L}_{\mathsf{W}}$ formulas is taken from there. In the spirit of the mentioned paper, we are mainly interested in statements concerning the predicate W, as the realisability is a means to prove that the provably total functions are the ones computable in polynomial time. This is reflected in the definition of the realisers of a formula: realisers of W contain more detailed information about the individuals considered than realisers of other formulas. Theorem 4.15 proves that the conclusion of every provable sequent can be realised by a polynomial time function from the realisers of the premise. The desired upper bounds immediately follow from this fact.

For realising provable sequents, we work in the theory $\mathsf{PET{+}J}^{iG}$ augmented by the axioms for totality (Tot) and extensionality (Ext) as introduced in Section 2.1.3. In this context, the relation symbol $\downarrow$ becomes superfluous and can therefore be neglected.

Furthermore, the definition of the realisability depends heavily on the model $\mathcal{M}^\star$ presented in the previous section as we only want to realise formulas which are modelled. Therefore, we first define the notion of realisability for formulas of the form $t \mathrel{\dot\in} s$. We remind the reader that $\langle \cdot, \cdot \rangle$ is the polynomial time pairing function introduced in Appendix C.

Definition 4.11 (Realisability: $\rho \Vdash t \mathrel{\dot\in} s$)
For any $s \in \mathcal{R}$ and $\rho \in \mathbb{W}$, $\rho \Vdash_\alpha t \mathrel{\dot\in} s$ is defined by induction on the level α when s was added to $\mathcal{R}$.

$\alpha = \mathbf{0}$:

$$\rho \Vdash_0 t \mathrel{\dot\in} \mathsf{id} \iff \rho = \epsilon \text{ and } \mathcal{M}^\star \models t = (t_0, t_0) \text{ for some term } t_0$$

$$\rho \Vdash_0 t \mathrel{\dot\in} \mathsf{w}(s) \iff \mathcal{M}^\star \models t = \overline{\rho} \wedge \overline{\rho} \leq s$$

$\alpha' = \alpha + 1$ **successor ordinal** where $s, s_0, s_1 \in \mathcal{R}_\alpha$

$$
\begin{array}{lcl}
\rho \,\check{\mathrm{r}}_{\alpha'}\, t \mathrel{\dot\in} s & \iff & s \in \mathcal{R}_\alpha \text{ and } \rho \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s \\
\rho \,\check{\mathrm{r}}_{\alpha'}\, t \mathrel{\dot\in} \mathsf{dom}(s) & \iff & \rho \,\check{\mathrm{r}}_\alpha\, (t, t_0) \mathrel{\dot\in} s \text{ for a term } t_0 \\
\rho \,\check{\mathrm{r}}_{\alpha'}\, t \mathrel{\dot\in} \mathsf{un}(s_0, s_1) & \iff & \rho = \langle i, \rho_0 \rangle \text{ and either} \\
 & & i = 0 \text{ and } \rho_0 \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s_0 \text{ or} \\
 & & i = 1 \text{ and } \rho_0 \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s_1 \\
\rho \,\check{\mathrm{r}}_{\alpha'}\, t \mathrel{\dot\in} \mathsf{int}(s_0, s_1) & \iff & \rho = \langle \rho_0, \rho_1 \rangle \text{ and } \rho_0 \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s_0 \text{ and } \rho_1 \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s_1 \\
\rho \,\check{\mathrm{r}}_{\alpha'}\, t \mathrel{\dot\in} \mathsf{inv}(r, s) & \iff & \rho \,\check{\mathrm{r}}_\alpha\, rt \mathrel{\dot\in} s \\
\rho \,\check{\mathrm{r}}_{\alpha'}\, t \mathrel{\dot\in} \mathsf{j}(s, r) & \iff & \rho = \langle \rho_0, \rho_1 \rangle \text{ and} \\
 & & \rho_0 \,\check{\mathrm{r}}_\alpha\, (t)_0 \mathrel{\dot\in} s \text{ and } \rho_1 \,\check{\mathrm{r}}_\alpha\, (t)_1 \mathrel{\dot\in} r(t)_1
\end{array}
$$

α **limit ordinal**

$$
\rho \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s \iff \rho \,\check{\mathrm{r}}_\beta\, t \mathrel{\dot\in} s \text{ for some } \beta < \alpha
$$

In this definition, *the name s is a placeholder* for all terms t such that $t \overset{\beta\eta}{=} s$ to improve readability. ⊛

We will also write $\rho \,\check{\mathrm{r}}\, t \mathrel{\dot\in} s$ for $\rho \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s$ for some ordinal α. We can now define the actual realisability for positive formulas.

Definition 4.12 (Realisability: $\rho \Vdash_{\mathrm{r}} A$)
The realisability relation $\Vdash_{\mathrm{r}} \subseteq \mathbb{W} \times \mathsf{Pos}$ for positive formulas is defined by induction on the construction of the formula:

Atomic formulas:

$$
\begin{array}{lcl}
\rho \Vdash_{\mathrm{r}} \mathsf{W}(t) & \iff & \mathcal{M}^\star \models \overline{\rho} = t \\
\rho \Vdash_{\mathrm{r}} t_0 = t_1 & \iff & \rho = \epsilon \text{ and } \mathcal{M}^\star \models t_0 = t_1 \\
\rho \Vdash_{\mathrm{r}} t \mathrel{\dot\in} s & \iff & s \in \mathcal{R} \text{ and } \rho \,\check{\mathrm{r}}_\alpha\, t \mathrel{\dot\in} s \text{ for some } \alpha \\
\rho \Vdash_{\mathrm{r}} \Re(s) & \iff & s \in \mathcal{R} \text{ and } \forall \sigma, t : \sigma \Vdash_{\mathrm{r}} t \mathrel{\dot\in} s \implies \sigma \leq \rho
\end{array}
$$

Composite formulas:

$$\begin{array}{lcl}
\rho \circledR A_0 \wedge A_1 & \iff & \rho = \langle \rho_0, \rho_1 \rangle \text{ and } \rho_0 \circledR A_0 \text{ and } \rho_1 \circledR A_1 \\
\rho \circledR A_0 \vee A_1 & \iff & \rho = \langle i, \rho_0 \rangle \ (i \in \{0,1\}) \text{ and } \rho_0 \circledR A_i \\
\rho \circledR \forall x A[x] & \iff & \rho \circledR A[u] \text{ for a fresh variable } u \\
\rho \circledR \exists x A[x] & \iff & \rho \circledR A[t] \text{ for some term } t
\end{array}$$

$\vec{\rho} \circledR \Gamma$ for a sequence $\Gamma = A_0, \ldots, A_n$:

$$\vec{\rho} \circledR \Gamma \quad \iff \quad \vec{\rho} = \rho_0, \ldots, \rho_n \text{ and } \rho_i \circledR A_i \qquad \circledast$$

Before we can prove the main theorem of this section, we first need to state two important properties of the defined realisability. First, the realiser of a formula shall not be able to distinguish between two terms having a common reduct. Furthermore, whenever we have a realiser for a formula containing free variables, it realises all substitution instances.

Lemma 4.13 *For positive formulas A and terms t, s, we have*

1) *If $\rho \circledR A[t]$ and $t \overset{\beta\eta}{=} s$, then $\rho \circledR A[s]$*
2) *If $\rho \circledR A[u]$, then $\rho \circledR A[t]$.*

Proof This is obvious from the definition of realisability. □

Second, we also require that we have realisers for all statement of the form $t \doteq s$ modelled by $\mathcal{M}^\star$.

Lemma 4.14 *For all terms t, s*

$$\mathcal{M}^\star \models t \doteq s \quad \Longrightarrow \quad \text{there is a } \rho \in \mathbb{W} \text{ such that } \rho \,\check{\mathrm{r}}\, t \doteq s$$

Proof Assume that $\mathcal{M}^\star \models t \doteq s$. This implies that $s \in \Re^{\mathcal{M}^\star}$ and $t \in ext(s) \equiv (t, s) \in \mathcal{E}$. $s \in \Re^{\mathcal{M}^\star}$ iff $s \in \mathcal{R}_\alpha$ for some α. Therefore we will prove the existence of a realiser ρ by induction on α such that $s \in \mathcal{R}_\alpha$ but $s \notin \mathcal{R}_\beta$ for $\beta < \alpha$.

$\alpha = 0$ We have either $s \overset{\beta\eta}{=} \mathsf{id}$ or $s \overset{\beta\eta}{=} \mathsf{w}a$ for some $a \in \mathsf{W}^{\mathcal{M}^\star}$. In the first case $t\ \varepsilon_0\ s$ iff $t \overset{\beta\eta}{=} (m, m)$ for some $m \in |\mathcal{M}^\star|$ which is equivalent to writing $\mathcal{M}^\star \models t = (m, m)$. Set $\rho = \epsilon$.

In the latter case we have $t\ \varepsilon_0\ s$ iff $\mathcal{M}^\star \models \mathsf{W}(t) \wedge t \leq a$. Choose ρ such that $\overline{\rho} \overset{\beta\eta}{=} t$ which exists since $t \in \mathsf{W}^{\mathcal{M}^\star}$.

$\alpha' = \alpha + 1$ s must be equivalent to one of $\mathsf{inv}(r, s_0)$, $\mathsf{dom}(s_0)$, $\mathsf{un}(s_0, s_1)$, $\mathsf{int}(s_0, s_1)$, or $\mathsf{j}(s_0, r)$ where $s_0, s_1 \in \mathcal{R}_\alpha$.

If $s \overset{\beta\eta}{=} \mathsf{int}(s_0, s_1)$, then $t\ \varepsilon_{\alpha'}\ s$ iff $t\ \varepsilon_\alpha\ s_0$ and $t\ \varepsilon_\alpha\ s_1$. By induction hypothesis there are ρ_0, ρ_1 such that $\rho_i\ \check{\mathsf{r}}_\alpha\ t \mathrel{\dot\in} s_i (i = 0, 1)$. Take $\rho = \langle \rho_0, \rho_1 \rangle$.

Realisers for $\mathsf{inv}, \mathsf{dom},$ and un are constructed analogously.

In the case, $s \overset{\beta\eta}{=} \mathsf{j}(s_0, r)$, we know that $rm \in \mathcal{R}_\alpha$ for all $m \in \mathit{ext}(s_0)$ by the construction of the model. $s\ \varepsilon_{\alpha'}\ \mathsf{j}(r, s_0)$ iff $s \overset{\beta\eta}{=} (m, n)$ for some $m, n \in |\mathcal{M}^\star|$ such that $m\ \varepsilon_\alpha\ s_0$ and $n\ \varepsilon_\alpha\ rm$. By induction hypothesis there are ρ_0 and ρ_1 such that $\rho_0\ \check{\mathsf{r}}_\alpha\ m \mathrel{\dot\in} s_0$ and $\rho_1\ \check{\mathsf{r}}_\alpha\ n \mathrel{\dot\in} rm$. Choose $\rho = \langle \rho_0, \rho_1 \rangle$. □

We are now ready to state the realisability theorem. It claims that whenever we can prove a (positive) sequent $\Gamma \Rightarrow C$ in $\mathsf{PET{+}J}^{iG}$, there is a polynomial time computable function F constructing a realiser for C given realisers for Γ. F may depend on the proof of the sequent and therefore the same sequent can have various realising functions constructed from different deductions.

Theorem 4.15 (Realisability) *For every positive sequent $\Gamma[\vec{x}] \Rightarrow C[\vec{x}]$ (with $\Gamma = A_0[\vec{x}], \ldots, A_n[\vec{x}]$) provable in $\mathsf{PET{+}J}^{iG}$, where $\vec{x}$ is a conclusive enumeration of the free variables, there is a function $F \in \mathrm{FPTIME}$ such that for all terms $\vec{t}$*

$$\vec{\rho}\ \textcircled{r}\ \Gamma[\vec{t}] \implies F(\vec{\rho})\ \textcircled{r}\ C[\vec{t}]$$

Proof This proof is by induction on the derivation of $\Gamma \Rightarrow C$ (where $\Gamma \equiv A_0, \ldots, A_n$).

Derivation length 0, i.e. $\Gamma \Rightarrow C$ is an axiom. The proof for the axioms of BOW is already spelled out by Strahm in [53] and therefore omitted here.

(w_a.1) Assume $\rho_n \,\textcircled{r}\, \mathsf{W}(s)$, that is $s \overset{\beta\eta}{=} \overline{\rho_n}$. We now set F to be $(\lambda\vec{x}.x_n)$. Obviously, $F(\vec{\rho}) \in \mathsf{W}$ and $\mathsf{w}(s) \in \mathcal{R}$ by model construction. Thus, only the last condition remains to be shown:
As $\sigma \,\textcircled{r}\, t \doteq \mathsf{w}(s) \iff \mathcal{M}^\star \models \overline{\sigma} = t \wedge \overline{\sigma} \leq s$, we know that $\sigma \leq s$ if $\sigma \,\textcircled{r}\, t \doteq \mathsf{w}(s)$ for any t and therefore $\sigma \leq \overline{\rho_n} = F(\vec{\rho})$.

(w_a.2) Assume (i) $\rho_{n-2} \,\textcircled{r}\, \mathsf{W}(s)$, (ii) $\rho_{n-1} \,\textcircled{r}\, \mathsf{W}(t)$ and (iii) $\rho_n \,\textcircled{r}\, t \leq s$ where $t \leq s \equiv \mathsf{c}_{\subseteq}(\mathsf{I}_{\mathsf{W}}t)(\mathsf{I}_{\mathsf{W}}s) = 0$. Choose $F = (\lambda\vec{x}.x_{n-1})$. Then $F(\vec{\rho}) = \rho_{n-1}$ and $\rho_{n-1} \,\textcircled{r}\, t \doteq \mathsf{w}(s)$ since $\mathcal{M}^\star \models t = \overline{\rho}_{n-1}$ with (ii) and $\mathcal{M}^\star \models \overline{\rho}_{n-1} \leq s$ because of (iii).

(w_a.3) Set $F = (\lambda\vec{x}.\langle x_{n-1}, \epsilon\rangle)$. $F(\vec{\rho}) \,\textcircled{r}\, \mathsf{W}(t) \wedge t \leq s$: $\rho_{n-1} \,\textcircled{r}\, \mathsf{W}(t)$ as $\rho_{n-1} \,\textcircled{r}\, t \doteq \mathsf{w}(s)$ and $\epsilon \,\textcircled{r}\, \overline{\rho}_{n-1} \leq s$ by assumption.

(id.1) $F = (\lambda\vec{x}.\epsilon)$. By definition, we know that $\rho \,\textcircled{r}\, t \doteq \mathsf{id} \implies \rho = \epsilon$. As $\epsilon \leq \epsilon$, the last part of the condition is fulfilled. Obviously, $\epsilon \in \mathsf{W}$ and $\mathsf{id} \in \mathcal{R}$.

(id.2) Assume that $\rho_n \,\textcircled{r}\, \exists u(t = (u, u))$, i.e. $\rho_n \,\textcircled{r}\, t = (s, s)$ for some term s. By definition, this means $\rho_n = \epsilon$ and $\mathcal{M}^\star \models t = (s, s)$ for some term s. Thus we can choose $F = (\lambda\vec{x}.\epsilon)$ which satisfies the condition for a realiser of $t \doteq \mathsf{id}$.

(id.3) Assume $\rho_n \,\textcircled{r}\, t \doteq \mathsf{id}$ which by definition means that $\rho_n = \epsilon$ and $\mathcal{M}^\star \models t = (s, s)$ for some s. We can choose the same F as in the previous case.

(un.1) Let $F = (\lambda\vec{x}.\langle 1, x_n * x_{n-1}\rangle)$. We need to show that indeed $F(\vec{\rho}) \,\textcircled{r}\, \Re(\mathsf{un}(s_0, s_1))$ if $\rho_{n-1} \,\textcircled{r}\, \Re(s_0)$ (i) and $\rho_n \,\textcircled{r}\, \Re(s_1)$ (ii). From the construction of $\mathcal{M}^\star$, $\mathsf{un}(s_0, s_1) \in \mathcal{R}$ is immediate. Now assume $\sigma \,\textcircled{r}\, t \doteq \mathsf{un}(s_0, s_1)$. By definition, it follows that either $\sigma = \langle 0, \sigma'\rangle$ and $\sigma' \,\textcircled{r}\, s \doteq s_0$ and therefore $\sigma' \leq \rho_{n-1}$ with (i). Or

$\sigma = \langle 1, \sigma' \rangle$ and $\sigma' \circledR t \mathrel{\dot\in} s_1$ and therefore $\sigma' \leq \rho_n$ with (ii). This implies $\sigma' \leq \rho_n * \rho_{n-1}$ and thus $\sigma \leq \langle 1, \rho_n * \rho_{n-1} \rangle$.

(un.2) $F = (\lambda\vec{x}.\langle 0, x_n \rangle)$. Obvious.

(un.3) $F = (\lambda\vec{x}.\langle 1, x_n \rangle)$. Obvious.

(un.4) $F = (\lambda\vec{x}.x_n)$. Obvious.

(int.1) $F = (\lambda\vec{x}.\langle x_{n-1}, x_n \rangle)$. For any $\sigma \circledR t \mathrel{\dot\in} \mathsf{int}(s_0, s_1)$, $\sigma = \langle \sigma_0, \sigma_1 \rangle$ with $\sigma_0 \leq \rho_0$ and $\sigma_1 \leq \rho_1$ and therefore $\sigma \leq \langle \rho_{n-1}, \rho_n \rangle$.

(int.2) $F = (\lambda\vec{x}.\langle x_{n-1}, x_n \rangle)$. Obvious.

(int.3) $F = (\lambda\vec{x}.\langle x_n \rangle_0)$. Obvious

(int.4) $F = (\lambda\vec{x}.\langle x_n \rangle_1)$. Obvious

(dom.1) Assume $\rho_n \circledR \Re(s)$ which means by definition that $s \in \mathcal{R}$ and $\sigma \circledR t \mathrel{\dot\in} s \implies \sigma \leq \rho_n$ (i). If we choose $F = (\lambda\vec{x}.x_n)$, then $F(\vec{\rho}) = \rho_n$. From the construction of $\mathcal{M}^\star$, $\mathsf{dom}(s) \in \mathcal{R}$ if $s \in \mathcal{R}$. Now assume $\sigma \circledR t \mathrel{\dot\in} \mathsf{dom}(s)$ which by definition means $\sigma \circledR (t, r) \mathrel{\dot\in} s$ (ii). From (i) and (ii) obviously $\sigma \leq \rho_n$ and therefore $\rho_n \circledR \Re(\mathsf{dom}(s))$.

(dom.2) For $F = (\lambda\vec{x}.x_n)$, obviously $F(\vec{\rho}) \circledR t \mathrel{\dot\in} \mathsf{dom}(s)$.

(dom.3) Same as above.

(inv.1) Let $F = (\lambda\vec{x}.x_n)$. Obvious

(inv.2) $F = (\lambda\vec{x}.x_n)$. Obvious

(inv.3) $F = (\lambda\vec{x}.x_n)$. Obvious

Induction step:

(T-I_{W}) Assume there are derivations for $\Gamma \Rightarrow \Re(s)$, $\Gamma \Rightarrow \epsilon \doteq s$ and $\Gamma, \mathsf{W}(u), u \doteq s \Rightarrow \mathsf{s}_i u \doteq s$ $(i = 0, 1)$. By induction hypotheses there are functions $E, F, G_0, G_1 \in \mathrm{FP}_{\mathrm{TIME}}$ such that for all terms t, s and $i = 0, 1$

$$\vec{\rho} \mathbin{ⓡ} \Gamma \implies E(\vec{\rho}) \mathbin{ⓡ} \Re(s) \tag{4.8}$$

$$\vec{\rho} \mathbin{ⓡ} \Gamma \implies F(\vec{\rho}) \mathbin{ⓡ} \epsilon \doteq s \tag{4.9}$$

$$\vec{\rho} \mathbin{ⓡ} \Gamma; \sigma \mathbin{ⓡ} \mathsf{W}(t); \tau \mathbin{ⓡ} t \doteq s \implies G_i(\vec{\rho}, \sigma, \tau) \mathbin{ⓡ} \mathsf{s}_i t \doteq s \tag{4.10}$$

The required function H is now defined by recursion on notation:

$$H(\vec{\rho}, \epsilon) = F(\vec{\rho})$$
$$H(\vec{\rho}, \mathsf{s}_i\sigma) = G_i(\vec{\rho}, \sigma, H(\vec{\rho}, \sigma))$$

Now we need to show that $H(\vec{\rho}, \sigma) \mathbin{ⓡ} t \doteq s$. We prove this informally by induction on σ. If $\sigma = \epsilon$, F will construct a realiser for $\epsilon \doteq s$. Assuming that $H(\vec{\rho}, \sigma)$ is a realiser for $\overline{\sigma} \doteq s$, $G_i(\vec{\rho}, H(\vec{\rho}, \sigma)$ will construct a realiser for $\mathsf{s}_i\overline{\sigma} \doteq s$.

To prove that H is a polynomial time function, a bound is needed as H is constructed from functions in $\mathrm{FP}_{\mathrm{TIME}}$. This bound is provided by E. By induction hypothesis, $E(\vec{\rho}) \mathbin{ⓡ} \Re(s)$. Thus $s \in \Re^{\mathcal{M}^\star}$ and $\sigma \leq E(\vec{\rho})$ if $\sigma \mathbin{ⓡ} t \doteq s$ for some t by Definition 4.12. Therefore, $F(\vec{\rho}) \leq E(\vec{\rho})$ since $F(\vec{\rho}) \mathbin{ⓡ} \epsilon \doteq s$, and $G_i(\vec{\rho}, \sigma) \leq E(\vec{\rho})$ if σ is a realiser for $t \doteq s$ for some t. This allows us to apply the schema of bounded recursion on notation to get $H = \mathsf{BRN}(F, G_0, G_1, E)$.

(J.1) Assume we have derivations for $\Gamma, x \doteq s \Rightarrow \Re(rx)$ and $\Gamma \Rightarrow \Re(s)$. By induction hypothesis, there are functions $F, G \in \mathrm{FP}_{\mathrm{TIME}}$ such that

$$\vec{\rho}, \sigma \mathbin{ⓡ} \Gamma, t \doteq s \implies F(\vec{\rho}, \sigma) \mathbin{ⓡ} \Re(rt) \tag{4.11}$$

$$\vec{\rho} \mathbin{ⓡ} \Gamma \implies G(\vec{\rho}) \mathbin{ⓡ} \Re(s) \tag{4.12}$$

We now define the requested function H as

$$H(\vec{\rho}) = \langle G(\vec{\rho}), F'(\vec{\rho}, G(\vec{\rho}))\rangle$$

where F' is a monotone polynomial function majorising F (the polynomial limiting the growth of F).

In general, we know that if $\sigma \Vdash \Re(s)$ and $\sigma < \sigma'$, then $\sigma' \Vdash \Re(s)$ by Definition 4.12. Thus, $F'(\vec{\rho}, \sigma) \Vdash \Re(rx)$ since $F(\vec{\rho}, \sigma) \leq F'(\vec{\rho}, \sigma)$.

Now we need to show that $H(\vec{\rho}) \Vdash \Re(\mathsf{j}(s, r))$, assuming $\vec{\rho} \Vdash \Gamma$. To obtain this, we have to prove that $\mathsf{j}(s, r) \in \mathcal{R}$ and that $\tau \leq H(\vec{\rho})$ if $\tau \Vdash t \mathrel{\dot\in} \mathsf{j}(s, r)$ for some t.

We know that $F(\vec{\rho}, \sigma) \Vdash \Re(rx)$ provided $\sigma \Vdash x \mathrel{\dot\in} s$. This just guarantees only for realisable elements x that rx indeed is a name. But because of Lemma 4.14, every $x \in s$ has a realiser for every $x \in \mathit{ext}(s)$ as (4.12) ensures that $s \in \mathcal{R}$. Thus F constructs a realiser for $\Re(rx)$ for every $x \in \mathit{ext}(s)$. Therefore $\mathsf{j}(s, r) \in \mathcal{R}$ by construction of $\mathcal{M}^\star$.

It remains to be proved that realisers of elements of $\mathsf{j}(s, r)$ actually are smaller or equal to the candidate realiser produced by H. Assume $\tau \Vdash t \mathrel{\dot\in} (\mathsf{j}(s, r))$. Then $\tau = \langle \tau_0, \tau_1\rangle$ such that $\tau_0 \Vdash t_0 \mathrel{\dot\in} s$ and $\tau_1 \Vdash t_1 \mathrel{\dot\in} rt_0$ by Definition 4.12. Therefore, $\tau_0 \leq G(\vec{\rho})$ by (4.12). For any σ with $\sigma \Vdash t \mathrel{\dot\in} s$, we have $\sigma \leq G(\vec{\rho})$ because of (4.12). Thus $\tau_1 \leq F(\vec{\rho}, \sigma) \leq F'(\vec{\rho}, G(\vec{\rho}))$ since F' is monotone. Because of the monotonicity of the pairing function (see Theorem C.3), $\tau \leq H(\vec{\rho})$ as required.

(J.2) We have the same assumptions as in the above case. We now define H to be

$$H(\vec{\rho}, \tau) = \langle \epsilon, \tau\rangle$$

By Definition 4.12 and the above arguments.

(J.3) We have the same assumptions as in the case for (J.1). We now

define H to be

$$H(\vec{\rho}, \epsilon, \tau_0, \tau_1) = \langle \tau_0, \tau_1 \rangle$$

Logical Rules are treated in [53]. Although Strahm's theory was based on classical logic, the proof still works with little alteration. Our functions need not choose which formula to realise in the absence of side formulas on the right side. Hence the adjustment is straight forward and the functions become simpler as we can omit most case distinctions. □

The desired upper bounds for $\mathsf{PET{+}J}^i$ immediately follow from the previous theorem. Assume that we have a provably total function $G : \mathbb{W}^n \to \mathbb{W}$, i.e. for some suitable term t_G

$$\mathsf{PET{+}J}^i \vdash t_G : \mathsf{W}^n \mapsto \mathsf{W} \tag{4.13}$$

$$\mathsf{PET{+}J}^i \vdash t_G \overline{w_1} \cdots \overline{w_n} = \overline{G(w_1 \cdots w_n)} \tag{4.14}$$

Hence, $\mathsf{PET{+}J}^{iG} \vdash \Rightarrow t_G : \mathsf{W}^n \mapsto \mathsf{W}$ with Theorem 4.4. Unfolding abbreviations gives $\mathsf{PET{+}J}^{iG} \vdash \Rightarrow (\forall x_1, \ldots, x_n \in \mathsf{W})(t_G x_1 \cdots x_n \in \mathsf{W})$. We get $\mathsf{PET{+}J}^{iG} \vdash \mathsf{W}(r_1), \ldots, \mathsf{W}(r_n) \Rightarrow \mathsf{W}(t_G r_1 \cdots r_n)$ by applying logical rules. With the realisability theorem, we know that there is a function $F \in \mathrm{FP}_{\mathrm{TIME}}$ such that $F(\sigma_1, \ldots, \sigma_n) \circledR \mathsf{W}(t_G r_1 \cdots r_n)$ if $\sigma_i \circledR \mathsf{W}(r_i) (i = 1, \ldots, n)$. By Definition 4.12, $F(\vec{\sigma}) \circledR \mathsf{W}(t_G r_1 \cdots r_n)$ iff $\overline{F(\vec{\sigma})} \overset{\beta\eta}{=} t_G r_1 \cdots r_n$. Furthermore, $\sigma_i \circledR \mathsf{W}(r_i)$ iff $\overline{\sigma_i} \overset{\beta\eta}{=} r_i$. With (4.14) and equality we get $F(\vec{\sigma}) = G(\vec{\sigma})$.

Theorem 4.16 *The provably total functions of* $\mathsf{PET{+}J}^i$ *coincide with the functions computable in polynomial time.*

4.4 Realising Some Extensions

When we studied the theory PET in the previous chapter, we also considered several extensions not increasing the proof-theoretic strength. Of

particular interest were Cantini's uniformity principle together with the type constructor for universal quantification discussed in Section 3.4.1. We claim that we can also add those two principles to $\mathsf{PET{+}J}^i$ and keep the upper bounds. To establish this result, we extend the theorems proved so far in this chapter to include the uniformity principle and the all constructor. We will mainly spell out the extensions of the important proofs and skip the obvious expansions of definitions.

Definition 4.17
The theory $\mathsf{PET{+}J{+}\forall}^{iG}$ is defined as the theory $\mathsf{PET{+}J}^{iG}$ plus the following axioms:

$$(\mathsf{all}.1) \qquad \Gamma, \Re(s) \Rightarrow \Re(\mathsf{all}(s))$$

$$(\mathsf{all}.2) \qquad \Gamma, \Re(s), \forall y((t,y) \mathrel{\dot\in} s) \Rightarrow t \mathrel{\dot\in} \mathsf{all}(s)$$

$$(\mathsf{all}.3) \qquad \Gamma, \Re(s), t \mathrel{\dot\in} \mathsf{all}(s) \Rightarrow \forall y((t,y) \mathrel{\dot\in} s) \qquad \circledast$$

and the following rule for positive elementary formulas A:

$$(\mathsf{UP}) \ \frac{\Gamma \Rightarrow \forall x(\exists y \in \mathsf{W})A[x,y]}{\Gamma \Rightarrow (\exists y \in \mathsf{W})\forall x A[x,y]}$$

It is easy to see that we can extend the proof of Theorem 4.4 to include the additional axioms. Thus, the axioms and the rule as stated above are adequate reformulations of the axioms of Section 3.4.1.

Before we can adjust the definition of the realisability, we add the following case to the construction of the model $\mathcal{M}^\star$ at successor stages:

– $s \overset{\beta\eta}{=} \mathsf{all}(s_0)$ and $(s,t) \in \mathcal{E}_{\alpha'}$ iff $((t,y),s_0) \in \mathcal{E}_\alpha$ for all $y \in |\mathcal{M}^\star|$

Consequently, we also expand Definition 4.11 by the following case for $\alpha' = \alpha + 1$:

$$\rho \mathrel{\check{\mathrm{r}}}_{\alpha'} t \mathrel{\dot\in} \mathsf{all}(s) \qquad \Longleftrightarrow \qquad \rho \mathrel{\check{\mathrm{r}}}_{\alpha} (t,t_0) \mathrel{\dot\in} s \text{ for all terms } t_0$$

The properties stated in Lemma 4.13 and Lemma 4.14 can easily be checked for the extended definitions. Thus, we can repeat Theorem 4.15 for $\mathsf{PET{+}J{+}\forall}^{iG}$:

Theorem 4.18 *For every positive sequent* $\Gamma[\vec{x}] \Rightarrow C[\vec{x}]$ *provable in* PET+J+$\forall^{iG}$*, there is a function* $F \in$ FPTIME *such that for all terms* $\vec{t}$

$$\vec{\rho} \circledR \Gamma[\vec{t}] \implies F(\vec{\rho}) \circledR C[\vec{t}]$$

Proof Again, the proof is by induction on the length of the derivation. We only need to consider the axioms (all.1-3) for the base case and the rule (UP) in the induction step:

(all.1) Assume $\rho_n \circledR \Re(s)$ and define $F = (\lambda\vec{x}.x_n)$. Then $F(\vec{\rho}) = \rho_n$. By definition, $\sigma \circledR t \doteq \mathsf{all}(s) \iff \sigma \circledR (t, t_0) \doteq s$ for all t_0. By assumption, $\sigma \leq \rho_n$ and therefore $F(\vec{\rho}) \circledR \Re(\mathsf{all}(s))$.

(all.2) $F = (\lambda\vec{x}.x_n)$. By definition of realisability.

(all.3) $F = (\lambda\vec{x}.x_n)$. By definition of realisability.

(UP) By induction hypothesis, there is a function G such that $G(\vec{\rho}) \circledR \forall x(\exists y \in \mathsf{W})A[x, y]$ if $\vec{\rho} \circledR \Gamma$. By Definition 4.12, we know that $G(\vec{\rho}) = \langle \rho_0, \rho_1 \rangle$ such that $\rho_0 \circledR t_0 \in \mathsf{W}$ for some t_0 and $\rho_1 \circledR A[t_1, t_0]$ for all t_1. Therefore $\langle \rho_0, \rho_1 \rangle \circledR t_0 \in \mathsf{W} \wedge \forall x A[x, t_0]$ for some t_0, which means $\langle \rho_0, \rho_1 \rangle \circledR (\exists y \in \mathsf{W})\forall x A[x, y]$. Hence, we define $F(\vec{\rho}) = G(\vec{\rho})$. □

Again, the desired upper bounds are immediately derived from this theorem.

4.5 Classical Version

As already mentioned before, treating the constructor for disjoint union based on weak theories is more delicate than for stronger theories. We are not able to prove the upper bounds of PET+J (based on classical logic) employing the same schema as for the intuitionistic variant: the induction step fails for (J.1) in the proof of Theorem 4.15. Since we

have side formulas on the right side for the classical sequent calculus reformulation, we can not guarantee that F always generates a realiser for $\Re(rx)$, it could also sometimes realise one of the side formulas, depending on σ. Therefore it is impossible to decide whether to realise one of the side formulas or the main formula.

However, the proof concept can be applied to (classical) PET without join. The proof for the upper bounds presented in Section 3.3 employs a model theoretic argument. Adapting the concept of realisability gives a nice syntactical proof. First, the reformulation in sequent calculus, PET^G, has to be adjusted to sequents having side formulas on the right. We can take the classical Gentzen calculus **G3** as underlying structure and add side formulas to every non-logical axiom and rule. The proof of the equivalence of the reformulation (Theorem 4.4) does not depend on the logic being intuitionistic and thus needs only little alteration.

The model construction from Section 4.2 is adopted unaltered, except for omitting join and thus ω many steps suffice as explained above. Also Definition 4.12 for realisability remains the same. As we have a sequence of formulas on the right side now, our realising function has to specify which formula to realise. Therefore for $\Gamma \equiv A_0, \ldots, A_n$ we specify

$$\rho \circledR \bigvee \Gamma \quad \Longleftrightarrow \quad \rho = \langle i, \rho_0 \rangle \text{ with } 0 \leq i \leq n \text{ and } \rho_0 \circledR A_i$$

Then we can modulate the realisability theorem:

Theorem 4.19 (Realisability) *For every positive sequent $\Gamma[\vec{x}] \Rightarrow \Delta[\vec{x}]$ provable in PET^G, there is a function $F \in$ FPTIME such that for all terms $\vec{t}$*

$$\vec{\rho} \circledR \Gamma[\vec{t}] \implies F(\vec{\rho}) \circledR \bigvee \Delta[\vec{t}]$$

The proof follows the same procedure as in the intuitionistic case. The only difference is that we have to choose an i, i.e. we have to decide which formula to realise. For the axioms we always create realisers for the main formula. In the induction step, consisting only of (T-I_W), we make a case distinction whether the functions given by induction hypotheses realise

the main formula or one of the side formulas. Otherwise, the realiser functions remain untouched.

Chapter 5

Epilogue

In the introduction, we described our motivation for the research presented in this thesis. In this chapter, we summarise the accomplished results and quickly outline some open problems for further investigation.

First, we presented the theory PET that corresponds to the complexity class of polynomial time computable functions. To establish the proof-theoretic bounds, we started off from Strahm's theory PT^{-}, a slightly weakened version of PT, but proof-theoretically equivalent to it. We proved that PT^{-} can be embedded in PET, thus confirming the lower bounds. The upper bounds were established by means of a model theoretic argument by proving that PET is a conservative extension of PT^{-} with respect to formulas in the language of PT.

Cantini [6] showed that several extensions can be added to PT without increasing the proof-theoretic strength, among them notably the Uniformity Principle (UP). We presented a nice application of this principle in the form of a supplemental type constructor (all) for universal quantification. In the presence of this type generator, we are able to embed the full theory PT into PET+(all).

Strahm [53] introduced modular Applicative Theories that correspond to the four complexity classes of functions computable in polynomial time,

simultaneously in polynomial time and linear space, in polynomial space and in linear space by using their function algebra characterisations. We were able to illustrate that full systems of Explicit Mathematics of the desired strength can be constructed from the same modular building blocks. To be precise, the requested theories vary in the inclusion or exclusion of word multiplication as a basic operation and in the induction scheme that operates on the binary words either in lexicographic order or along the binary successor operations.

We also studied the addition of the join principle to PET. We changed to intuitionistic logic and established that the provably total functions of PET+J^i are the functions computable in polynomial time. For this proof, we reformulated PET+J^i in sequent calculus style and defined a realisability relation for positive formulas. We were able to show that each provable (positive) sequent can be realised by a polynomial time function. The desired upper bounds immediately follow from this result.

We also sketched how this proof can be adapted for the classical version of PET (without Join), thus giving a syntactical proof for the upper bounds of PET.

Even though this proof via the realisability relation works for classical PET and intuitionistic PET+J^i, it is not clear whether this can somehow be adapted to classical PET+J. Although we strongly conjecture that join does not increase the proof-theoretic strength in the classical setting, either, it would be interesting to investigate this relationship further since the treatment of join is delicate in weak theories.

In Section 4.2, we described a model construction for PET+J in detail. However, it is not clear to us where the construction of names and types actually stops. It is an interesting open question whether transfinite induction over all ordinal numbers is indeed necessary. Since join is an (expressively) powerful principle, this might also lead to a better understanding of its interaction with weak theories.

A further interesting research topic is the study of weak theories of partial truth. Cantini [6] introduced a weak Applicative Theory with a

partial truth predicate. However, the truth predicate is not allowed in induction formulas. It would be interesting to investigate the smalles ordinal number at which the model construction stops. Furthermore, the exact relationship between weak type theories and weak partial truth theories remains an open question.

In Explicit Mathematics, several more extensions were studied over various systems. It could be exciting to study them in the context of PET. Especially, it would be interesting to investigate weak Power Types, i.e. the type of all subtypes of a given type.

Appendices

Appendix A

Logic of Partial Terms

The systems of Explicit Mathematics presented in this thesis are based on the Logic of Partial Terms (LPT) originated by Beeson and Feferman. To make this thesis self-contained, the formal definitions are summarised here. Detailed information is presented e.g. in Beeson [1], Feferman [23].

Definition A.1 (Languages for LPT)
A language $\mathcal{L}$ for the Logic of Partial Terms comprises the following symbols:

1) Countably many variables $v0, v1, \ldots, vi, \ldots$ $(i \in \mathbb{N})$.

2) The logical symbols $\neg$ (negation), $\vee$ (or), $\wedge$ (and), $\rightarrow$ (implication), $\exists$ (exists), and $\forall$ (for all).

3) The unary symbol $\downarrow$ (definedness) and the binary symbol $=$ (equality).

4) For any $n \in \mathbb{N}$, we have a (possibly empty) set of n-ary function symbols. The 0-ary function symbols are called the *constants* of $\mathcal{L}$.

5) For any $n \in \mathbb{N}$, we have a (possibly empty) set of n-ary relation symbols.

6) Auxiliary symbols like brackets (,), ... ⊛

For every language $\mathcal{L}$ we define the terms and formulae as usual.

Definition A.2 ($\mathcal{L}$ Terms)
The terms of $\mathcal{L}$ are inductively defined as follows:

1) Every variable and every constant of $\mathcal{L}$ is an $\mathcal{L}$ term.

2) If $s_1, \ldots, s_n$ are $\mathcal{L}$ terms and f is a n-ary function symbol of $\mathcal{L}$ ($n \geq 1$), then $f(s_1, \ldots, s_n)$ is a $\mathcal{L}$ term. ⊛

Definition A.3 (Atomic Formulas and Formulas of $\mathcal{L}$)
The atomic formulas of $\mathcal{L}$ are the expressions $s{\downarrow}$, $s_1 = s_2$ and $R(s_1, \ldots, s_n)$ if $s, s_1, \ldots, s_n$ are $\mathcal{L}$ terms and R is a n-ary relation symbol.

The formulas are inductively defined by the following:

1) Every atomic formula of $\mathcal{L}$ is a $\mathcal{L}$ formula.

2) If A is a $\mathcal{L}$ formula, then so is $\neg A$.

3) If A, B are $\mathcal{L}$ formulas, then so are $(A \wedge B)$, $(A \vee B)$, and $A \rightarrow B$.

4) If A is a $\mathcal{L}$ formula and x a variable, then $\exists x A$ and $\forall x A$ are $\mathcal{L}$ formulas. ⊛

We usually omit the reference to $\mathcal{L}$ when the language is either obvious from the context or insignificant. Furthermore, we drop brackets whenever possible without causing ambiguity.

As a convention, we use lowercase letters $a, b, c, f, g, h, x, y, z$ for variables and r, s, t for terms; capital letters $A, B, C, \ldots$ typically denote formulas. We also use vector notation $\vec{a}$ and $\vec{s}$ to abbreviate finite sequences of variables $a_1, \ldots, a_n$ and terms $s_1, \ldots, s_m$ where the length is clear from the context.

Definition A.4 (Substitution)
For every term t and all variables $\vec{x} = x_1, \ldots, x_n$ and all terms $\vec{s} = s_1, \ldots, s_n$, the term $t[\vec{s}/\vec{x}]$ is inductively defined as follows:

1) If $t \equiv x_i$ with $1 \leq i \leq n$, then $t[\vec{s}/\vec{x}] \equiv s_i$.

2) If t is a constant or a variable different from $\vec{x}$, then $t[\vec{s}/\vec{x}] \equiv t$.

3) If f is a m-ary relation symbol ($m \geq 1$) and $t \equiv f(t_1, \ldots, t_m)$, then $t[\vec{s}/\vec{x}] \equiv f(t_1[\vec{s}/\vec{x}], \ldots, t_m[\vec{s}/\vec{x}])$.

For every formula A, the formula $A[\vec{s}/\vec{x}]$ is inductively defined by the following:

1) If $A \equiv t\!\downarrow$, then $A[\vec{s}/\vec{x}] \equiv t[\vec{s}/\vec{x}]\!\downarrow$.

2) If $A \equiv t = s$, then $A[\vec{s}/\vec{x}] \equiv t[\vec{s}/\vec{x}] = s[\vec{s}/\vec{x}]$.

3) If $A \equiv R(\vec{t})$ for some m-ary relation symbol R with $m \geq 1$, then $A[\vec{s}/\vec{x}] \equiv R(t_1[\vec{s}/\vec{x}], \ldots, t_m[\vec{s}/\vec{x}])$.

4) If $A \equiv \neg B$, then $A[\vec{s}/\vec{x}] \equiv \neg B[\vec{s}/\vec{x}]$

5) If $A \equiv B \odot C$, then $A[\vec{s}/\vec{x}] \equiv B[\vec{s}/\vec{x}] \odot C[\vec{s}/\vec{x}]$ ($\odot$ being one of $\wedge, \vee, \rightarrow$)

6) If $A \equiv QxB$ (Q being $\exists$ or $\forall$) and $x_{i_1}, \ldots, x_{i_k}$ the sequence of variables $\vec{x}$ without x, then $A[\vec{s}/\vec{x}] \equiv QyB[s_{i_1}, \ldots, s_{i_k}, y/x_{i_1}, \ldots x_{i_k}, x]$ where y a variable not occurring in $B, s_{i_1}, \ldots, s_{i_k}$. $\circledast$

We often write $A[\vec{x}]$ to indicate that the variables $\vec{x}$ may appear free in A and act as parameters in the current context. In this case, we usually write $A[\vec{s}]$ instead of $A[\vec{s}/\vec{x}]$.

Furthermore, we have the following abbreviations:

$$t \simeq s := t\!\downarrow \vee s\!\downarrow \rightarrow t = s$$
$$t \neq s := \neg(t = s) \wedge s\!\downarrow \wedge t\!\downarrow$$

We will now state the axioms and rules of LPT. They are formulated as a classical Hilbert calculus with equality.

I. Logical Axioms and Rules: We have the usual axioms and rules for classical Hilbert calculus, except that the quantifier axioms are

replaced by the following:

$$A[s/x] \wedge s{\downarrow} \rightarrow \exists x A \qquad \text{and} \qquad \forall x A \wedge s{\downarrow} \rightarrow A[s/x]$$

II. Definedness (Strictness). For any n-ary function symbol f and relation symbol R and all terms $t, t_1, \ldots, t_n$:

(D1) $t{\downarrow}$ if t is a variable or a constant.

(D2) $f(t_1, \ldots, t_n){\downarrow} \rightarrow t_1{\downarrow} \wedge \cdots \wedge t_n{\downarrow}$.

(D3) $t_1 = t_2 \rightarrow t_1{\downarrow} \wedge t_2{\downarrow}$.

(D4) $R(t_1, \ldots, t_n) \rightarrow t_1{\downarrow} \wedge \cdots \wedge t_n{\downarrow}$.

III. Equality. For any n-ary function symbol f and relation symbol R and all terms $t, t_1, \ldots, t_n, s_1, \ldots, s_n$:

(E1) $t = t$ if t is a variable or a constant.

(E2) $t_1 = t_2 \rightarrow t_2 = t_1$.

(E3) $t_1 = t_2 \wedge t_2 = t_3 \rightarrow t_1 = t_3$.

(E4) $R(t_1, \ldots, t_n) \wedge (t_1 = s_1) \wedge \cdots \wedge (t_n = s_n) \rightarrow R(s_1, \ldots, s_n)$.

(E5) $(t_1 = s_1) \wedge \cdots \wedge (t_n = s_n) \rightarrow f(t_1, \ldots, t_n) \simeq f(s_1, \ldots, s_n)$.

Provability of a formula in LPT is defined as usual and therefore not spelled out here. Also the semantics of LPT is the common one and thus omitted. Specific models needed for this thesis are introduced in the respective sections.

Appendix B

Function Algebra Characterisations of Weak Complexity Classes

A comprehensive introduction to complexity theory would go beyond the scope of this thesis. Thus, we only summarise the function algebra characterisation of the complexity classes playing a major role in this thesis. We assume familiarity with the basics of complexity theory, especially the general concept of function classes and Turing machine computability. The notations and results in this chapter are taken from Clote's survey [11] and we refer to this article for more details.

In the context of this thesis, the main focus is on systems characterising weak complexity classes. Thereby, we work with the set of binary words $\mathbb{W} = \{0,1\}^*$. We are interested in the functions over $\mathbb{W}$ which are computable by a Turing machine in polynomial time, simultaneously in polynomial time and linear space, in polynomial space, and in linear space. In this thesis, we write FPtime, FPtimeLinspace, FPspace, and FLinspace for the corresponding function classes.

We use the notation of Clote [11] for representing function algebras.

Hence, we let operators denote mappings (functionals) from functions on $\mathbb{W}$ to functions on $\mathbb{W}$. Let $\mathcal{F}$ a set of functions and $\mathcal{OP}$ a set of operators. Then $[\mathcal{F}; \mathcal{OP}]$ is called a *function algebra* and stands for the smallest set of functions that contains $\mathcal{F}$ and is closed under the operators in $\mathcal{OP}$. We now introduce the functions and operators used to characterise the complexity classes of interest.

In the following, we write ϵ for the 0-ary constant function resulting in the empty word. Furthermore, we let I denote the usual collection of projection functions. To construct binary words, we first need several successor functions on $\mathbb{W}$. Therefore, we let s_0 and s_1 denote the common functions appending 0 and 1 to a given binary word. We also use a lexicographic successor function s_ℓ which creates the successor of the given word according the well-ordering that orders words by length and then lexicographically. Formally, for any $w \in \mathbb{W}$, s_ℓ behaves congruent with the following:

$$\begin{aligned} \mathsf{s}_\ell(\epsilon) &= 0 \\ \mathsf{s}_\ell(\mathsf{s}_0 w) &= \mathsf{s}_1 w \\ \mathsf{s}_\ell(\mathsf{s}_1 w) &= \mathsf{s}_0(\mathsf{s}_\ell w) \end{aligned}$$

The lexicographical successor can also be seen as correspondence to the successor on the natural numbers if we look at the binary words as binary representations of natural numbers.

Furthermore, we have two binary functions on $\mathbb{W}$ representing concatenation and multiplication of words, $*$ and $\times$. Concatenation appends the second word to the first one and multiplication $x \times y$ results in the length of y times concatenation of x to itself.

After spelling out the basic functions, we now turn to the operators. We start with the composition operator COMP which stands for the usual composition of F with $G_1, \ldots, G_n$. In other words, given function $F, G_1, \ldots, G_n$, COMP maps them to the function applying F to the result of $G_1, \ldots, G_n$.

Finally, we now introduce two schemas for bounded recursion. We there-

fore take functions G, H, H_0, H_1, and B on $\mathbb{W}$ of matching arities and let $x \leq y$ stand for "the length of x is smaller or equal to the length of y". The function F is defined by *bounded recursion on notation* (BRN) from G, H_0, H_1, B if we have for all $\vec{x}, y \in \mathbb{W}$

$$\begin{aligned} F(\vec{x}, \epsilon) &= G(\vec{x}) \\ F(\vec{x}, \mathsf{s}_i y) &= H_i(\vec{x}, y, F(\vec{x}, y)) \quad (i = 0, 1) \\ F(\vec{x}, y) &\leq B(\vec{x}, y) \end{aligned}$$

Furthermore, we say that F is constructed by *bounded lexicographic recursion* (BRL) from G, H, B if

$$\begin{aligned} F(\vec{x}, \epsilon) &= G(\vec{x}) \\ F(\vec{x}, \mathsf{s}_\ell y) &= H(\vec{x}, y, F(\vec{x}, y)) \\ F(\vec{x}, y) &\leq B(\vec{x}, y) \end{aligned}$$

We are now ready to replicate the theorem establishing the recursion-theoretic characterisations of the four complexity classes used in this thesis.

Theorem B.1 *We have the following characterisations:*

1) $\mathrm{FP}_{\mathrm{TIME}} = [\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1, *, \times; \mathsf{COMP}, \mathsf{BRN}]$ *(Cobham [12])*

2) $\mathrm{FP}_{\mathrm{TIME}}\mathrm{L}_{\mathrm{INSPACE}} = [\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1, *; \mathsf{COMP}, \mathsf{BRN}]$ *(Thompson [59])*

3) $\mathrm{FP}_{\mathrm{SPACE}} = [\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1, *, \times; \mathsf{COMP}, \mathsf{BRL}]$ *(Thompson [59])*

4) $\mathrm{FL}_{\mathrm{INSPACE}} = [\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1, *; \mathsf{COMP}, \mathsf{BRL}]$ *(Ritchie [50])*

For detailed information refer to Clote [11].

Appendix C

A Pairing Function in PTLS

In the theory B introduced in Section 2.1.4, pairing is available as a built-in function on the universe where we can only prove $(\lambda x.\mathsf{p}_0\mathsf{p}xy) : \mathsf{W} \mapsto \mathsf{W}$ and $(\lambda x.\mathsf{p}_1\mathsf{p}yx) : \mathsf{W} \mapsto \mathsf{W}$. However, for some applications we need pairing and projection functions mapping binary words on binary words. In other words, we want a pairing function $\langle\cdot,\cdot\rangle : \mathsf{W}^2 \mapsto \mathsf{W}$ and corresponding projection functions. Furthermore, the pairing function should be monotone, i.e. if $u \leq x$ and $v \leq y$, then we request that $\langle u, v\rangle \leq \langle x, y\rangle$.

The most natural way to build such a pair as a binary word is by introducing a delimiter symbol and then concatenating the first word, the delimiter and the second word. Therefore, we first need to define some auxiliary functions for the coding mechanism. Some of these functions will be characteristic functions ("Boolean functions"). In this case, true is represented by 0 and false by 1. Furthermore, we can change the recursion operator r_W from Lemma 2.10 such that all arities are reduced by 1, i.e. f is a constant binary word, $g : \mathsf{W}^2 \mapsto \mathsf{W}$, $b : \mathsf{W} \mapsto \mathsf{W}$ and $\mathsf{r}_\mathsf{W} f g b : \mathsf{W} \mapsto \mathsf{W}$.

First, we define a function determining whether the length of a binary word is even:

Lemma C.1 *There is a closed $\mathcal{L}_{\mathsf{W}}$-term t_{even} such that* PTLS *proves*

1) $x \in \mathsf{W} \to (t_{\mathsf{even}}x = 0 \vee t_{\mathsf{even}}x = 1)$

2) $x \in \mathsf{W} \wedge x \neq \epsilon \to (t_{\mathsf{even}}\epsilon = 0 \wedge (t_{\mathsf{even}}x = 0 \leftrightarrow t_{\mathsf{even}}(\mathsf{p}_{\mathsf{W}}x) = 1)$

Proof Choose $t_{\mathsf{even}} := (\lambda x.\mathsf{r}_{\mathsf{W}} f g b x)$ where

$f := 0$	ϵ is even.
$g := (\lambda x z.\mathsf{d}_{\mathsf{W}} 1 0 z 0)$	switch between 0 and 1
$b := (\lambda x.\mathsf{l}_{\mathsf{W}} x)$	bound: length of x

The totality of g and b is obvious and therefore Lemma 2.10 can be applied.

1) Immediate by definition of d_{W}.

2) We work informally in PTLS. Let $A[x]$ be the Σ^{b-}_{W} formula

$$(x = \epsilon \wedge t_{\mathsf{even}}x = 0) \vee \\ (t_{\mathsf{even}}x = 0 \wedge t_{\mathsf{even}}(\mathsf{p}_{\mathsf{W}}x) = 1) \vee (t_{\mathsf{even}}x = 1 \wedge t_{\mathsf{even}}(\mathsf{p}_{\mathsf{W}}x) = 0)$$

With part 1), for any binary word x

$$A[x] \leftrightarrow x \neq \epsilon \to (t_{\mathsf{even}}\epsilon = 0 \wedge (t_{\mathsf{even}}x = 0 \leftrightarrow t_{\mathsf{even}}(\mathsf{p}_{\mathsf{W}}x) = 1)$$

$A[\epsilon]$ obviously holds. Now we assume $A[x]$ and aim at showing $A[\mathsf{s}_i x]$ $(i = 0, 1)$. $t_{\mathsf{even}}(\mathsf{s}_i x) = g(\mathsf{s}_i x)(t_{\mathsf{even}}x)$. With part 1) we know that either $t_{\mathsf{even}}x = 0$ or $t_{\mathsf{even}}x = 1$. In the former case, $t_{\mathsf{even}}\mathsf{s}_i x = g(\mathsf{s}_i x)0 = \mathsf{d}_{\mathsf{W}}1000 = 1$. In the latter case, $g(\mathsf{s}_i x)1 = \mathsf{d}_{\mathsf{W}}1010 = 0$ as desired. We can now apply (Σ^{b-}_{W}-l_{W}) to get $(\forall x \in \mathsf{W})A[x]$. □

The next step is the definition of an encoding function mapping a binary word on a word where 0 is replaced by 00 and 1 by 11. This will enable us to choose a distinguishable delimiter for the pairing function later.

Lemma C.2 *There are closed terms* t_{enc} *and* t_{dec} *such that*

1) $t_{\mathsf{enc}} : \mathsf{W} \mapsto \mathsf{W} \wedge t_{\mathsf{dec}} : \mathsf{W} \mapsto \mathsf{W}$

2) $t_{\mathsf{enc}}\epsilon = \epsilon \wedge t_{\mathsf{dec}}\epsilon = \epsilon$

3) $(\forall x \in \mathsf{W})(t_{\mathsf{dec}}t_{\mathsf{enc}}x = x)$

4) $x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x \leq y \rightarrow t_{\mathsf{enc}}x \leq t_{\mathsf{enc}}y$

Proof Informally, the encoding function shall map 0 onto $00(= \mathsf{s}_0\mathsf{s}_0\epsilon)$ and 1 onto $11(= \mathsf{s}_1\mathsf{s}_1\epsilon)$. Therefore, we define $t_{\mathsf{enc}} := (\lambda x.\mathsf{r}_{\mathsf{W}}afbx)$ where

$$\begin{aligned} a &:= \epsilon \\ f &:= (\lambda xz.z * \mathsf{d}_{\mathsf{W}}(00)(11)(\mathsf{s}_0\mathsf{p}_{\mathsf{W}}y)y) \\ &\equiv (\lambda xz.\mathsf{d}_{\mathsf{W}}(\mathsf{s}_0\mathsf{s}_0z)(\mathsf{s}_1\mathsf{s}_1z)(\mathsf{s}_0\mathsf{p}_{\mathsf{W}}y)y) \\ b &:= (\lambda x.\mathsf{l}_{\mathsf{W}}x * \mathsf{l}_{\mathsf{W}}x) \end{aligned}$$

and set $t_{\mathsf{dec}} := (\lambda x.\mathsf{r}_{\mathsf{W}}cgd)$ where

$$\begin{aligned} c &:= \epsilon \\ g &:= (\lambda xz.\mathsf{d}_{\mathsf{W}}(hxz)z(t_{\mathsf{even}}x)0) \\ h &:= (\lambda xz.(\mathsf{d}_{\mathsf{W}}(z*0)[\mathsf{d}_{\mathsf{W}}(z*1)\epsilon(\mathsf{s}_1\mathsf{s}_1\mathsf{p}_{\mathsf{W}}\mathsf{p}_{\mathsf{W}}x)x](\mathsf{s}_0\mathsf{s}_0\mathsf{p}_{\mathsf{W}}\mathsf{p}_{\mathsf{W}}x)x)) \\ d &:= (\lambda x.\mathsf{l}_{\mathsf{W}}x) \end{aligned}$$

1) $\mathsf{PTLS} \vdash f : \mathsf{W}^2 \mapsto \mathsf{W}$ and $\mathsf{PTLS} \vdash b : \mathsf{W} \mapsto \mathsf{W}$ obvious. Furthermore, it is easy to see that also $g : \mathsf{W}^2 \mapsto \mathsf{W}$ and $d : \mathsf{W} \mapsto \mathsf{W}$. The claim now follows with Lemma 2.10

2) $t_{\mathsf{enc}}\epsilon = \mathsf{r}_{\mathsf{W}}afb\epsilon = a = \epsilon$. Analogous for $t_{\mathsf{dec}}\epsilon$.

3) Take $A[x] \equiv t_{\mathsf{dec}}t_{\mathsf{enc}}x = x$. $A[\epsilon]$ obviously holds with part 2). We assume $A[x]$ and aim at showing $A[\mathsf{s}_i x]$. We have $t_{\mathsf{enc}}\mathsf{s}_0x = t_{\mathsf{enc}}x * 00$ and $t_{\mathsf{dec}}(t_{\mathsf{enc}}x * 00) = (t_{\mathsf{dec}}t_{\mathsf{enc}}x) * 0 = x * 0 = \mathsf{s}_0x$. Analogous for $t_{\mathsf{dec}}t_{\mathsf{enc}}\mathsf{s}_1x$ and therefore $(\Sigma^b_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ gives $(\forall x \in \mathsf{W})A[x]$

4) Let $A[y] \equiv \mathsf{c}_{\subseteq}(\mathsf{l}_{\mathsf{W}}x)(\mathsf{l}_{\mathsf{W}}y) = 1 \vee \mathsf{c}_{\subseteq}(\mathsf{l}_{\mathsf{W}}(t_{\mathsf{enc}}x))(\mathsf{l}_{\mathsf{W}}(t_{\mathsf{enc}}y)) = 0$ which is equivalent to monotonicity for $x, y \in \mathsf{W}$. We prove this by induction on y. If $y = \epsilon$, either $x \not\leq y$ or $x = \epsilon$ and in both cases we have $A[\epsilon]$.

   Assume $A[y]$. In the induction step, we only consider the case $x \leq \mathsf{s}_i y$. We have $\mathsf{l}_{\mathsf{W}}(t_{\mathsf{enc}}(\mathsf{s}_i y)) = \mathsf{l}_{\mathsf{W}}(\mathsf{s}_i\mathsf{s}_i(t_{\mathsf{enc}}y)) = \mathsf{s}_1\mathsf{s}_1(\mathsf{l}_{\mathsf{W}}(t_{\mathsf{enc}}y))$. As we assume $x \leq \mathsf{s}_i y \equiv \mathsf{l}_{\mathsf{W}}x \subseteq \mathsf{l}_{\mathsf{W}}(\mathsf{s}_i y)$, we have either $\mathsf{l}_{\mathsf{W}}x = \mathsf{l}_{\mathsf{W}}\mathsf{s}_i y$ or $\mathsf{l}_{\mathsf{W}}x \subseteq \mathsf{p}_{\mathsf{W}}(\mathsf{l}_{\mathsf{W}}\mathsf{s}_i y) = \mathsf{l}_{\mathsf{W}}y$. In the first case, $x \neq \epsilon$ and thus $x = \mathsf{s}_0\mathsf{p}_{\mathsf{W}}x$ or $x = \mathsf{s}_1\mathsf{p}_{\mathsf{W}}x$. This gives $\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}x = \mathsf{s}_1\mathsf{s}_1(\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}(\mathsf{p}_{\mathsf{W}}x))$.
   If $\mathsf{l}_{\mathsf{W}}x = \mathsf{l}_{\mathsf{W}}\mathsf{s}_i y$, we get $\mathsf{l}_{\mathsf{W}}(\mathsf{p}_{\mathsf{W}}x) = \mathsf{l}_{\mathsf{W}}y$ from the axioms about l_{W}. Therefore we can apply the induction hypothesis and get $\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}\mathsf{p}_{\mathsf{W}}x \subseteq \mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}y$. Hence obviously $\mathsf{s}_1\mathsf{s}_1(\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}\mathsf{p}_{\mathsf{W}}x) \subseteq \mathsf{s}_1\mathsf{s}_1(\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}y)$ as desired. In the latter case ($\mathsf{l}_{\mathsf{W}}x \subseteq \mathsf{l}_{\mathsf{W}}y$), we can immediately apply induction hypothesis to get $\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}x \subseteq \mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}y$. As above, $\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}y \subseteq \mathsf{s}_1\mathsf{s}_1(\mathsf{l}_{\mathsf{W}}t_{\mathsf{enc}}y)$ and with transitivity, we get the desired result.

   Thus we have $(\forall y \in \mathsf{W})A[y]$ if $x \in \mathsf{W}$. □

After preparing the auxiliary functions, the actual pairing and projection functions can be constructed:

Theorem C.3 *There are closed $\mathcal{L}_{\mathsf{W}}$ terms $t_{\mathsf{p}}, t_{\mathsf{p0}}, t_{\mathsf{p1}}$ such that* PTLS *proves*

1) $t_{\mathsf{p}} : \mathsf{W}^2 \mapsto \mathsf{W}$

2) $t_{\mathsf{p0}} : \mathsf{W} \mapsto \mathsf{W} \wedge t_{\mathsf{p1}} : \mathsf{W} \mapsto \mathsf{W}$

3) $x \in \mathsf{W} \wedge y \in \mathsf{W} \rightarrow (t_{\mathsf{p0}}t_{\mathsf{p}}xy = x \wedge t_{\mathsf{p1}}t_{\mathsf{p}}xy = y)$

4) $u \in \mathsf{W} \wedge v \in \mathsf{W} \wedge x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge u \leq x \wedge v \leq y \rightarrow t_{\mathsf{p}}uv \leq t_{\mathsf{p}}xy$

Proof Define $t_{\mathsf{p}} := (\lambda xy.t_{\mathsf{enc}}x{*}10{*}t_{\mathsf{enc}}y)$ and let $t_{\mathsf{pi}} := (\lambda x.t_{\mathsf{dec}}t_{\mathsf{findi}}x)(i = 0, 1)$ such that the actual work is delegated to the function finding the

part left or right of the delimiter. We define $t_{\mathsf{find0}} = (\lambda y.\mathsf{r_W} a f b y)$ where

$$
\begin{aligned}
a &:= \epsilon \\
f &:= (\lambda x z.\mathsf{d_W}[\mathsf{d_W}(\mathsf{p_W}\mathsf{p_W}x)z(\mathsf{s_1}\mathsf{s_0}\mathsf{p_W}\mathsf{p_W}x)x]z(t_{\mathsf{even}}x)0) \\
b &:= (\lambda x.\mathsf{l_W}x)
\end{aligned}
$$

We choose $t_{\mathsf{find1}} = (\lambda y.\mathsf{r_W} c g d)$ such that

$$
\begin{aligned}
c &:= \epsilon \\
g &:= (\lambda x z.\mathsf{d_W}[\mathsf{d_W}\epsilon[hxz](\mathsf{s_1}\mathsf{s_0}\mathsf{p_W}\mathsf{p_W}x)x][hxz](t_{\mathsf{even}}x)0) \\
h &:= (\lambda x z.\mathsf{d_W}[\mathsf{s_1}z][\mathsf{s_0}z](\mathsf{s_1}x)x) \\
d &:= (\lambda x.\mathsf{l_W}x)
\end{aligned}
$$

The properties of the functions are now immediate:

1) Obvious from totality of t_{enc}.

2) Obvious from totality of auxiliary functions.

3) Proof by induction analogous to Lemma C.2.

4) Immediate with monotonicity of t_{enc}. □

In this thesis, we write $\langle x, y\rangle$ for $t_{\mathsf{p}}xy$ and $\langle x\rangle_i$ $(i = 0, 1)$ for $t_{\mathsf{p}i}x$.

Bibliography

[1] Michael J. Beeson. *Foundations of Constructive Mathematics: Metamathematical Studies*. Springer, Berlin, 1985.

[2] Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2:97–110, 1992.

[3] Samuel R. Buss. *Bounded Arithmetic*. Bibliopolis, Napoli, 1986.

[4] Andrea Cantini. Feasible operations and applicative theories based on $\lambda\eta$. *Mathematical Logic Quarterly*, 46(3):291–312, 2000.

[5] Andrea Cantini. Polytime, combinatory logic and positive safe induction. *Archive for Mathematical Logic*, 41(2):169–189, 2002.

[6] Andrea Cantini. Choice and uniformity in weak applicative theories. In M. Baaz, Sy Friedman, and J. Krajíček, editors, *Logic colloquium '01*, volume 20 of *Lecture Notes in Logic*, pages 108–138. Association for Symbolic Logic, 2005.

[7] Andrea Cantini. *Logical frameworks for truth and abstraction*. North-Holland, Amsterdam, 1996.

[8] Andrea Cantini. Proof-theoretic aspects of self-referential truth. In Maria Luisa Dalla Chiara et. al., editor, *Tenth international congress of logic, methodology and philosophy of science, florence, august 1995*, volume 1, pages 7–27. Kluwer, September 1997.

[9] Andrea Cantini. Characterizing poly-time with an intuitionistic theory based on combinatory logic and safe induction. Preprint, Firenze, 1999. 14 pages.

[10] Andrea Cantini and Pierluigi Minari. Uniform inseparability in explicit mathematics. *Journal of Symbolic Logic*, 64(1):313–326, 1999.

[11] Peter Clote. Computation models and function algebras. In E. Griffor, editor, *Handbook of computability theory*, pages 589–681. Elsevier, 1999.

[12] Alan Cobham. The intrinsic computational difficulty of functions. In *Logic, methodology and philosophy of science II*, pages 24–30. North Holland, Amsterdam, 1965.

[13] Stephen A. Cook and Bruce M. Kapron. Characterizations of the basic feasible functionals of finite type. In S. R. Buss and P. J. Scott, editors, *Feasible mathematics*, pages 71–95. Birkhäuser, Basel, 1990.

[14] Stephen A. Cook and Alasdair Urquhart. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic*, 63(2):103–200, 1993.

[15] Solomon Feferman. Notes on Operational Set Theory I. http://math.stanford.edu/~feferman/papers/OperationalST-I.pdf, 2001.

[16] Solomon Feferman. A language and axioms for explicit mathematics. In J.N. Crossley, editor, *Algebra and logic*, volume 450 of *Lecture Notes in Mathematics*, pages 87–139. Springer, Berlin, 1975.

[17] Solomon Feferman. Recursion theory and set theory: a marriage of convenience. In J. E. Fenstad, R. O. Gandy, and G. E. Sacks, editors, *Generalized recursion theory II, Oslo 1977*, volume 94 of

Stud. Logic Found. Math, pages 55–98. North Holland, Amsterdam, 1978.

[18] Solomon Feferman. Constructive theories of functions and classes. In M. Boffa, D. van Dalen, and K. McAloon, editors, *Logic colloquium '78*, pages 159–224. North Holland, Amsterdam, 1979.

[19] Solomon Feferman. Iterated inductive fixed-point theories: application to Hancock's conjecture. In G. Metakides, editor, *The patras symposion*, pages 171–196. North Holland, Amsterdam, 1982.

[20] Solomon Feferman. Polymorphic typed lambda-calculi in a type-free axiomatic framework. In W. Sieg, editor, *Logic and computation*, volume 106 of *Contemporary Mathematics*, pages 101–136. American Mathematical Society, Providence, Rhode Island, 1990.

[21] Solomon Feferman. Logics for termination and correctness of functional programs. In Y. N. Moschovakis, editor, *Logic from computer science*, volume 21 of *MSRI Publications*, pages 95–127. Springer, Berlin, 1991.

[22] Solomon Feferman. Logics for termination and correctness of functional programs II: logics of strength PRA. In P. Aczel, H. Simmons, and S. S. Wainer, editors, *Proof theory*, pages 195–225. Cambridge University Press, Cambridge, 1992.

[23] Solomon Feferman. Definedness. *Erkenntnis*, 43:295–320, 1995.

[24] Solomon Feferman and Gerhard Jäger. Systems of explicit mathematics with non-constructive μ-operator. Part II. *Annals of Pure and Applied Logic*, 79(1):37–52, 1996.

[25] Solomon Feferman and Gerhard Jäger. Systems of explicit mathematics with non-constructive μ-operator. Part I. *Annals of Pure and Applied Logic*, 65(3):243–263, 1993.

[26] Fernando Ferreira. *Polynomial time computable arithmetic and conservative extensions*. PhD thesis, Pennsylvania State University, 1988.

[27] Fernando Ferreira. Polynomial time computable arithmetic. In Wilfried Sieg, editor, *Logic and computation, proceedings of a workshop held at carnegie mellon university, 1987*, volume 106 of *Contemporary Mathematics*, pages 137–156. American Mathematical Society, Providence, Rhode Island, 1990.

[28] Petr Hájek and Pavel Pudlák. *Metamathematics of first-order arithmetic*. Perspectives in Mathematical Logic. Springer, 1993.

[29] Susumu Hayashi and Hiroshi Nakano. *Px: a computational logic*. MIT Press, Cambridge, MA, 1988.

[30] J. Roger Hindley and Jonathan P. Seldin. *Introduction to combinators and λ-calculus*. Cambridge University Press, 1986.

[31] Gerhard Jäger. Induction in the elementary theory of types and names. In E. Börger, H. Kleine Büning, and M.M. Richter, editors, *Computer Science Logic '87*, volume 329 of *Lecture Notes in Computer Science*, pages 118–128. Springer, Berlin, 1988.

[32] Gerhard Jäger. Power types in explicit mathematics? *Journal of Symbolic Logic*, 62(4):1142–1146, 1997.

[33] Gerhard Jäger. On Feferman's operational set theory. *Annals of Pure and Applied Logic*, 150(1-3):19 – 39, 2007. ISSN 0168-0072.

[34] Gerhard Jäger. Applikative Theorien und Explizite Mathematik. Lecture Notes iam-97-001, Institut für Informatik und angewandte Mathematik, Universität Bern, 1997. http://www.iam.unibe.ch/publikationen/techreports/1997/iam-97-001.

[35] Gerhard Jäger and Thomas Strahm. Totality in applicative theories. *Annals of Pure and Applied Logic*, 74(2):105–120, 1995.

[36] Gerhard Jäger and Thomas Studer. Extending the system T_0 of explicit mathematics: the limit and Mahlo axioms. *Annals of Pure and Applied Logic*, 114(1–3):79–101, 2002.

[37] Gerhard Jäger, Reinhard Kahle, and Thomas Strahm. On applicative theories. In A. Cantini, E. Casari, and P. Minari, editors, *Logic and foundations of mathematics*, pages 83–92. Kluwer, 1999.

[38] Gerhard Jäger, Reinhard Kahle, and Thomas Studer. Universes in explicit mathematics. *Annals of Pure and Applied Logic*, 109(3): 141–162, 2001.

[39] David Jansen. *Ontologische Aspekte Expliziter Mathematik*. Master's thesis, Institut für Informatik und angewandte Mathematik, 1997.

[40] Reinhard Kahle. *The Applicative Realm*. Habilitation Thesis, Tübingen, 2007. Appeared in Textos de Mathemática 40, Departamento de Mathemática da Universidade de Coimbra, Portugal, 2007.

[41] Reinhard Kahle. Frege structures for partial applicative theories. Technical Report IAM-96-013, Institut für Informatik und angewandte Mathemati, Universität Bern, September 1996.

[42] Reinhard Kahle. *Applikative Theorien und Frege-Strukturen*. PhD thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 1997.

[43] Bruce Kapron and Stephen Cook. A new characterization of type 2 feasibility. *SIAM Journal on Computing*, 25:117–132, 1996.

[44] Jürg Krähenbühl. *Explicit mathematics with positive existential comprehension and join*. Master's thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 2006. Available at http://www.iam.unibe.ch/til/publications.

[45] J. Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*, volume 60 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1995.

[46] Daniel Leivant. A foundational delineation of poly-time. *Information and Computation*, 110:391–420, 1994.

[47] Kurt Melhorn. Polynomial and abstract subrecursive classes. *Journal of Computer and System Science*, 12:147–178, 1976.

[48] Geoffrey Ostrin and Stan S. Wainer. Elementary Arithmetic. *Annals of Pure and Applied Logic*, 133:275–292, 2005.

[49] Dieter Probst. *Pseudo-Hierarchies in Admissible Set Theory without Foundation and Explicit Mathematics*. PhD thesis, Universität Bern, Institut für Informatik und angewandte Mathematik, 2005.

[50] Robert W. Ritchie. Classes of predictably computable functions. *Transactions of the American Mathematical Society*, 106:139–173, 1963.

[51] Daria Spescha and Thomas Strahm. Elementary explicit types and polynomial time operations. *Mathematical Logic Quarterly*, 55(3): 245–258, 2009.

[52] Thomas Strahm. *Proof-theoretic Contributions to Explicit Mathematics*. Habilitationsschrift, University of Bern, 2001.

[53] Thomas Strahm. Theories with self-application and computational complexity. *Information and Computation*, 185:263–297, 2003.

[54] Thomas Strahm. A proof-theoretic characterization of the basic feasible functionals. *Theoretical Computer Science*, 329:159–176, 2004.

[55] Thomas Strahm. Partial applicative theories and explicit substitutions. *Journal of Logic and Computation*, 6(1):55–77, 1996.

[56] Thomas Strahm. *On the proof theory of applicative theories*. PhD thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 1996.

[57] Thomas Strahm. Polynomial time operations in explicit mathematics. *Journal of Symbolic Logic*, 62(2):575–594, 1997.

[58] Thomas Studer. *Object-oriented programming in explicit mathematics: towards the mathematics of objects*. PhD thesis, Universität Bern, Institut für Informatik und angewandte Mathematik, 2001.

[59] David B. Thompson. Subrecursiveness: machine independet notions of computability in restricted time and storage. *Mathematical Systems Theory*, 6:3–15, 1972.

[60] Anne S. Troelstra and Helmut Schwichtenberg. *Basic Proof Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, first edition, 1996.

Keyword Index

Symbol Index

Axiom Index

List of Definitions and Theorems