Weak applicative theories, truth, and computational complexity

Inauguraldissertation
der philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von
Sebastian Eberhard
von Schänis SG

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

Weak applicative theories, truth, and computational complexity

Inauguraldissertation
der philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von
Sebastian Eberhard
von Schänis SG

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

Von der philosophisch-naturwissenschaftlichen Fakultät angenommen.

Der Dekan:
Prof. Dr. S. Decurtins

Bern, 4.6.2013

Acknowledgements

First of all, I would like to thank Prof. Dr. Strahm for his valuable guidance and advice during my PhD studies, and Prof. Dr. Jäger who made this thesis possible. I also thank Prof. Dr. Halbach, the co-referee, for helping me to improve the thesis.

Many thanks to the members of the Logic and Theory group, who have provided a stimulating environment. In particular, I will remember our interesting discussions at coffee - and lunch breaks.

I thank my family and girlfriend for their constant support during my PhD studies.

The research for the thesis was supported by the Swiss National Science Foundation.

Contents

Abstract

The aim of our thesis is the design of applicative theories on words of strength below Primitive Recursive Arithmetic to give implicit characterisations of complexity classes in the style of Strahm's [79]. In more detail, we analyse the effect of adding second-order notions as types or truth to an applicative base theory, and obtain theories of great expressive power, but still polynomial strength. The relation between second-order extensions featuring types, or truth, respectively, is analysed in detail. We also design theories corresponding naturally to several logarithmic complexity classes using two different word predicates reflecting safe and normal arguments, respectively. In addition, we suggest a new two-sorted characterisation of logspace which is very similar to Cook and Bellantoni's B [7].

Introduction

Combinatory logic was introduced by Schönfinkel [70] (or [71] for the English translation) and analysed by Curry in [25, 26, 27]. Despite its simplicity, containing only two combinators and variables, combinatory logic is complete in the sense that for any recursive function it contains a corresponding term. Therefore, combinatory logic can be seen as abstract model of computation. Combinatory logic was also employed for foundational projects as e.g. in Gödels Dialectica interpretation where he proves the consistency of Heyting arithmetic [48]. For extensive surveys in this area, we recommend Beeson's [4] and Troelstra and van Dalen's [82].

In the mid seventies, Feferman used applicative theories, a formalisation of combinatory logic, as base theory of his systems of explicit mathematics [33]. These theories employ the logic of partial terms, which was introduced by Beeson in [4, 5]. Feferman's theories introduce a typing discipline in the style of set theory, and can be used to formalise Bishop style constructive mathematics [8, 9]. His theories are called explicit because all types are named by objects of the combinatorial universe, and operations on types have as counterparts operations of the combinatory algebra. The applicative base theory is very flexible and has a high expressive power, especially in combination with types. This allows to obtain many interesting theories of the strength of various subsystems of second order analysis by varying type construction principles, induction principles, or the operational base theory. Theories over a strengthened applicative base theory, allowing to represent non-recursive functions, are considered in [41, 42, 56, 57, 55]. Theories that expand Feferman's system T_0 of explicit mathematics by universes are given in [35, 77, 54, 58]. For theories that restrict the type construction - or induction principles, see. e.g. [75, 63, 74].

Another possibility to allow typing over a combinatorial setting is the extension of applicative theories by a truth predicate, which mimics the properties of truth, and uses a coding mechanism for formulas build into the applicative base structure. The notion of a partial, self-referential truth predicate is rooted in Kripke's seminal work [64]. Theories which expand an *applicative* core with such a truth predicate are introduced by Aczel [1] and Beeson [4]. A nice feature of these theories is that they avoid Gödelisation, and that the fixed point theorem for applicative theories allows a straightforward production of liar sentences. Theories of truth also allow to interpret naive set theory by stipulating $x \in a$ as $\mathsf{T}(ax)$. By varying truth reflection and induction principles, Cantini [13, 14] and Kahle [59, 60] obtained applicative theories of truth of strengths between Primitive Recursive Arithmetic and $\widehat{ID_1}$. Stronger applicative theories of truth are proposed by introducing levels [13], or universes [60]. For important results in the realm of truth theories over *arithmetical* base theories, we recommend Feferman [36, 40], and Friedman and Sheard [46]. For a comprehensive overview and newer results see Halbach's [49].

Applicative theories of truth and explicit mathematics have also been used in Feferman's unfolding program [38], whose idea is to expand a schematic theory by the operations, predicates and deduction principles which are implicit in the acceptance of the theory. The great expressive power of applicative theories makes the precise formalisation of this enterprise possible. Feferman and Strahm carried out this program for schematic systems of infinitist, and finitist arithmetic in [43, 44].

Weak logical theories can be used to justify computation - and especially recursion principles present in implicit characterisations of complexity classes. Mostly, recursion principles are matched by suitable induction principles. Research on this subject was started by Buss in [11] where theories of bounded arithmetic for the subclasses of the polynomial hierarchy are introduced. Further theories of bounded arithmetic have been introduced by Clote and Takeuti [22] amongst others for the logarithmic hierarchy, alternating logarithmic time, logarithmic space, and various circuit complexity classes. Weak arithmetic *second-order* theories corresponding to various complexity classes, analysed by various researchers, are collected in Cook and Nguyen's [24]. *Applicative* theories corresponding to complexity classes have been introduced by Strahm [78, 79] for linear and polynomial time - and space classes, by Cantini for polynomial time [15], and by Kahle and Oitavem [62] for the polynomial hierarchy. In contrast to corresponding theories of bounded arithmetic, applicative theories allow to prove the totality of the functions of the suitable complexity class without coding. This makes the lower bound proofs typically easier and more transparent. An overview of weak applicative theories is given in Strahm's [81].

Altogether, we have pointed out the usefulness of applicative theories in formalizing the following three mathematical notions: sets, truth, and complexity classes. In this thesis, we present new results in all of these domains: In chapter 2, we analyse a theory of truth, T_{PT}, of polynomial strength over a base theory of combinatory algebra. The intended universe of T_{PT} contains the binary words, and the theory is equipped with polynomially growing functions. It is a close companion to the earlier mentioned [18], where the only essential difference is that T_{PT} allows the reflection of elementship in the words only for bounded words. So, we have the following axiom

$$w \in \mathsf{W} \to (v \in \mathsf{W} \wedge v \leq w \leftrightarrow \mathsf{T}(\dot{\mathsf{W}}wv)),$$

where $v \leq w$ is intended to hold if v is a word of smaller length than w [1]. As Cantini's theory, T_{PT} is appealing because of its simplicity, especially if one compares it to the theory PETJ (see [74]) of explicit mathematics of equal strength. Despite its weakness, T_{PT} has a high expressive power, allowing straightforward embeddings of Strahm's PT, the theory PETJ, and also of systems of bounded arithmetic of matching strength. The theory T_{PT} also supports the proof-theoretic analysis of various theories of the feasible unfolding program as we will explain later.

As usual for applicative theories of low proof theoretic strength, we prove the upper bound of T_{PT} using a realisation approach. Nevertheless, new ideas

[1]The length of a word is its number of bits, $\dot{\mathsf{W}}$ is an applicative constant allowing the coding of formulas containing W.

are needed because realisation approaches introduced by Strahm [79] and Cantini [18] are unable to deal with the possibly exponential copying process of information in T_{PT}. This means that in T_{PT}, we can define a function r satisfying the following recursion equations for any $w \in \mathsf{W}$.

- $r(\epsilon) = 0 \dot{=} 0$ [2]
- $r(\mathsf{s}_i w) = r(w) \dot{\wedge} r(w)$

Using the compositional truth-axiom for $\wedge$ and truth induction, we can prove the sequent

$$\mathsf{T}_{\mathsf{PT}} \vdash x \in \mathsf{W} \Rightarrow \mathsf{T}(rx),$$

which cannot be realised by the mentioned approaches. To deal with this problem, we define a new realisation approach manipulating addresses and pointers. It allows to represent data more efficiently by sharing realisation information between different formulas and subformulas. Technically this is implemented using pointers which allow the sharing of data by pointing to the same storage address. This approach can also be applied to obtain an alternative proof of the feasibility of a two-sorted algebra on sets recently introduced by Arai [2], where a similar problem of a possibly exponential copying process occurs.

The theory T_{PT} does not allow the reflection of negative formulas. Nevertheless, we strongly assume that at least the following principle,

$$a \neq b \leftrightarrow \mathsf{T}(a \neq b),$$

can be added without strengthening the theory. The reason why this principle is problematic is that all realisation approaches used until now in the applicative setting require a restriction to positive cut-formulas in the setting of classical logic. Obviously, the addition of the above mentioned axiom makes such a cut-elimination impossible. This is also the reason why until now no applicative theory of strength below primitive recursive arithmetic has been introduced which allows induction for formulas containing negated

[2] $\dot{=}$ and $\dot{\wedge}$ are applicative constants allowing the coding of equations, and conjunctions, respectively.

equations. In chapter 3, we made first steps in solving this difficult problem at least in an intuitionistic setting.

We succeeded in proving that the above mentioned reflection of negated equations can be added to an *intuitionistic* version of T_{PT} without changing its proof-theoretic strength. By inspection of our proof it becomes clear immediately that the same technique allows to find upper bounds for intuitionistic versions of the applicative systems presented in Strahm's [79] enhanced by induction over formulas containing negated equations.

We also proved that the additional extension of an intuitionistic version of T_{PT} by an axiom of choice does not increase its strength. As in Cantini's [18], who also equips his theory of truth by choice principles, this is proved by a realisation approach using feasible functionals. Nevertheless, in our case additional difficulties occur due to the fact that the implementation of the address-pointer realisation approach requires the functionals not only to be feasible but also to fulfil certain bounding conditions.

In chapter 4, we study the relation between systems of explicit mathematics and theories of truth by finding embeddings of theories of truth into theories of explicit mathematics and vice versa. The systems have primitive recursive or polynomial strength, respectively. While the embedding of systems of explicit mathematics into systems of truth is straightforward, for the opposite embedding additional principles are necessary. They are motivated by an exact analysis of the reason of the lack of expressive power of weak theories of explicit mathematics compared with weak theories of truth: Interpretations for both predicates, truth and elementship, are build using a Σ_1 fixed point. But for formulas of the form $a \in b$ for a fixed b, the restriction of the elementship predicate to some finite stage of the fixed point still delivers a correct interpretation while for formulas of the form $\mathsf{T}(ba)$ this cannot be guaranteed. In consequence, the addition of universes that have to be interpreted using the full Σ_1 fixed point for elementship removes this asymmetry. We managed to prove that it gives weak systems of explicit mathematics the expressive power necessary to embed corresponding theories of truth.

We have also examined theories of truth and of explicit mathematics in the context of Feferman's unfolding program. In chapter 5, we carry out the unfolding program for a base theory which only justifies bounded existential quantification, whereas the base system for the finitist unfolding allows unrestricted existential quantification. For this base theory, the unfolding program cannot be carried out in exactly the same way as for the finitist and non-finitist arithmetic. The reason is that allowing a type which reflects the set of words is philosophically questionable in a feasible setting since it immediately gives rise to the definition of exponentially growing functions. Nevertheless, the reference to a set of words seems necessary to restrict the applicative axioms for the constants representing initial functions on the words. We solved this problem by introducing two types of variables, the first intended to range over the words, and the second over the whole applicative universe. Then, restriction to the words can be formulated by imposing syntactical restrictions on the occurring variables.

Modulo these modifications the unfolding program can be executed as expected, resulting in natural theories of truth and explicit mathematics of polynomial strength. As usual for unfoldings in this style, and contrary to PETJ and T_{PT}, these theories allow unrestricted induction. For the upper bound proof, we make use of the theory T_{PT} into which all unfoldings can be embedded easily, using for the systems of explicit mathematics the techniques described in chapter 4.

In chapter 6, we define new applicative systems matching logarithmic complexity classes in the style of Strahm's [79]. The essential difference to characterizations given by Clote and Takeuti [21] is that in our systems recursion principles are justified by induction principles, whereas [21] uses a bit comprehension principle. We succeeded in giving very uniform characterizations for the logarithmic hierarchy, alternating logarithmic time, and polynomial time using known implicit characterisations of these classes with concatenation recursion as main principle. The function algebras are formalised by applicative theories containing two word predicates, corresponding to safe and normal inputs and outputs, and are inspired by Cantini's [17].

This chapter in addition contains a new two-sorted function algebra $\mathfrak{LS}$ for logarithmic space. It differs from Cook and Bellantoni's B [7] for polynomial time only by restricting case distinction, and by allowing the additional initial function mapping an input word to the word corresponding to its length. It is also a close relative to Neergaard's function algebra for logspace [66] but contrary to his algebra, $\mathfrak{LS}$ does not contain any affinity restrictions.

We formalize $\mathfrak{LS}$ as the above mentioned algebras. For the upper bound computation of the resulting theory, we introduce a new realisation approach which also delivers a correct upper bound proof for the theory introduced in Cantini's [17].

Chapters 4, and 5 are based on joint work with Thomas Strahm, which has been published as [32, 30, 31]. Chapters 2, and 6 are based on the articles [29, 28] which have been submitted to journals.

Chapter 1

Logical background

Applicative theories are formalisations of combinatory algebras. They formalise a universe of operations that can freely be applied to each other, without being bound to a specific domain. Some operations are universal and are naturally self-applicable as a result, like the identity operation or the pairing operation. Some are partial, which means that they do not yield a value on each input. Because of this possibility, applicative theories are usually formalised within Beeson's logic of partial terms. The theories guarantee that operations satisfy the laws of a partial combinatory algebra with pairing, projections, and definition by cases. In this chapter, we introduce a specific applicative theory B which is, modulo minor changes, employed as base theory of all introduced applicative systems. We fix notions and conventions used within the whole thesis, in contrast to notations and conventions introduced later, which are only valid in the scope of their chapter. We discuss the proof-theoretic analysis of weak applicative systems in general, and introduce Strahm's realisation approach [79], on which the later applied realisation approaches are based.

1.1 The words

Words are given as elements of $\{0,1\}^*$, so they are finite sequences of zeros and ones. The length $|w| \in \mathbb{N}$ of a word w is defined in the obvious way. The word w consists of $|w|$ bits. Its first bit is the rightmost one and is given by $\mathsf{BIT}(0,w)$, the other bits are given by $\mathsf{BIT}(1,w)$, $\mathsf{BIT}(2,w), \cdots, \mathsf{BIT}(|w|-1,w)$. As usual, the most significant bits are at the left -, the least significant bits at the right side. As usual, we stipulate that the high order bits are at the left -, the low order bits at the right side. The relation $\prec$ orders the words first according to their length, and then lexicographically.

For words w, v and numbers n, m we have $w \prec v$ exactly if

$$\begin{aligned}
&|w| < |v| \vee (|w| = |v| \wedge \\
&(\exists n \leq |w|-1)(\forall m < n \leq |w|-1) \\
&(\mathsf{BIT}(m,w) = \mathsf{BIT}(m,v) \wedge \mathsf{BIT}(n,w) = 0 \wedge \mathsf{BIT}(n,v) = 1)
\end{aligned}$$

We write $w \preceq v$ for $w \prec v \vee w = v$. We call this order the lexicographic order on words in the following.

1.2 The theory B

Let us introduce the applicative base theory B, which is Strahm's $\mathsf{B}(*, \times)$ introduced in [78, 79] with the axioms for tally length, lexicographic successor and - predecessor dropped [1]. The axioms of B contain the usual applicative axioms about the combinators and pairing, and in addition axioms claiming that certain constants represent simple functions on words.

1.2.1 The language L of B

L is a first order language for the logic of partial terms which includes:

- variables $a, b, c, x, y, z, u, v, f, g, h, \ldots$
- constants k, s, p, $\mathsf{p_0}$, $\mathsf{p_1}$, $\mathsf{d_W}$, ϵ, $\mathsf{s_0}$, $\mathsf{s_1}$, $\mathsf{p_W}$, $\mathsf{c_\subseteq}$, $*$, $\times$
- the binary relation symbol $=$ (equality), and the unary relation symbols $\downarrow$ (definedness), and W (binary words)
- the binary function symbol $\circ$ (application)

[1] The tally length of a word x can be defined as $1 \times x$ within B. The lexicographic successor and predecessor are only used in chapter 6.

The meaning of the constants will become clear in the next subsection.

The terms $(r, s, t, p, q, \dots)$ are defined in the expected way, using the function symbol application. The formulas $(A, B, C, \dots)$ of L are build recursively from the connectors $\wedge$, $\vee$, $\rightarrow$, and the quantifiers $\forall$, $\exists$. We call a formula positive, if it does not contain $\rightarrow$. We assume the following standard abbreviations and syntactical conventions:

$$\begin{array}{rcl}
t_1 t_2 \dots t_n & := & (\dots(t_1 \circ t_2) \circ \dots \circ t_n) \\
s(t_1, \dots, t_n) & := & s t_1 \dots t_n \\
t_1 \simeq t_2 & := & t_1{\downarrow} \vee t_2{\downarrow} \;\rightarrow\; t_1 = t_2 \\
\langle t \rangle & := & t \\
\langle t_1, \dots, t_{n+1} \rangle & := & \mathsf{p}\langle t_1, \dots, t_n \rangle t_{n+1} \\
t \in \mathsf{W} & := & \mathsf{W}(t) \\
t : \mathsf{W}^k \rightarrow \mathsf{W} & := & (\forall x_1 \dots x_k \in \mathsf{W}) t x_1 \dots x_k \in \mathsf{W} \\
s \leq t & := & \mathsf{c}_{\subseteq}(1 \times s, 1 \times t) = 0 \\
s \leq_{\mathsf{W}} t & := & s \leq t \wedge s \in \mathsf{W} \\
\neg A & := & A \rightarrow \epsilon = \mathsf{s}_0 \epsilon \\
s \neq t & := & \neg s = t
\end{array}$$

In the following we often write $A[\vec{x}]$ in order to indicate that the variables $\vec{x} = x_1, \dots, x_n$ may occur free in A. We write $A[\vec{s}]$ for $A[\vec{x}]$ with the variables $\vec{x}$ replaced by $\vec{s}$. It will be always clear from the context which variables are replaced. In such a context, we call $[\vec{s}]$ a substitution.

1.2.2 The rules and axioms of B

The logic of B is the *classical* logic of partial terms due to Beeson [4, 5]. Let us spell out the *non-logical axioms* of B. They consist of the usual applicative axioms, and axioms expressing that the constants for simple functions on words fulfil the suitable recursion equations. For readability, we divide them into seven groups.

I. Partial combinatory algebra and pairing

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

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

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

II. Definition by cases on W

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

(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) $\epsilon \in \mathsf{W} \wedge (\forall x \in \mathsf{W})(\mathsf{s}_0x \in \mathsf{W} \wedge \mathsf{s}_1x \in \mathsf{W})$,

(7) $\mathsf{s}_0x \neq \mathsf{s}_1y \wedge \mathsf{s}_0x \neq \epsilon \wedge \mathsf{s}_1x \neq \epsilon$,

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

(9) $x \in \mathsf{W} \rightarrow \mathsf{p}_{\mathsf{W}}(\mathsf{s}_0x) = x \wedge \mathsf{p}_{\mathsf{W}}(\mathsf{s}_1x) = x$,

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

V. Initial subword relation.

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

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

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

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

VI. Word concatenation.

(19) $* : \mathsf{W}^2 \rightarrow \mathsf{W}$,

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

(21) $x \in \mathsf{W} \wedge y \in \mathsf{W} \rightarrow x{*}(\mathsf{s}_0y) = \mathsf{s}_0(x{*}y) \wedge x{*}(\mathsf{s}_1y) = \mathsf{s}_1(x{*}y)$.

VII. Word multiplication.

(22) $\times : \mathsf{W}^2 \to \mathsf{W}$,

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

(24) $x \in \mathsf{W} \land y \in \mathsf{W} \;\rightarrow\; x \times (\mathsf{s}_0 y) = (x \times y) * x \;\land\; x \times (\mathsf{s}_1 y) = (x \times y) * x$.

Let us introduce two additional axioms.

Extensionality of operations:

$$(\mathsf{Ext}) \qquad (\forall f)(\forall g)[(\forall x)(fx \simeq gx) \to f = g]$$

Totality of application:

$$(\mathsf{Tot}) \qquad (\forall x)(\forall y)(xy\downarrow)$$

In the later chapters, we tacitly expand B by *extensionality of operations.* For the theories of truth, we expand B additionally by *totality of application* and call the resulting theory $\mathsf{B}\downarrow$. Observe that in the presence of the totality axiom, the logic of partial terms reduces to ordinary classical predicate logic.

1.3 Theorems and models of B

Let us turn to some crucial theorems of the base theory B. For proofs of these standard results, the reader is referred to Beeson [4] or Feferman [33].

Lemma 1 (Explicit definitions and fixed points)

1. *For each* L *term t there exists an* L *term $(\lambda x.t)$ not containing the variable x such that*

$$\mathsf{B} \vdash (\lambda x.t)\downarrow \;\land\; (\lambda x.t)x \simeq t$$

2. *There is a closed* L *term* fix *so that*

$$\mathsf{B} \vdash \mathsf{fix}g\downarrow \;\land\; \mathsf{fix}gx \simeq g(\mathsf{fix}gx)$$

Let us quickly remind the reader of two standard models of B, namely the recursion-theoretic model $\mathcal{PR}$ and the term model $\mathcal{TM}$. For an extensive discussion of many more models of the applicative basis, the reader is referred to Beeson [4] and Troelstra and van Dalen [83].

Example 2 (Recursion-theoretic model $\mathcal{PR}$) Take the universe of binary words $\mathbb{W} = \{0,1\}^*$ and interpret application $\circ$ as partial recursive function application in the sense of ordinary recursion theory.

Example 3 (The open term model $\mathcal{TM}$) Take the universe of open λ terms and consider the usual reduction of the extensional untyped lambda calculus $\lambda\eta$, augmented by suitable reduction rules for the constants other than k and s. Interpret application as juxtaposition. Two terms are equal if they have a common reduct and W denotes those terms that reduce to a "standard" word $\overline{w}$. Note that $\mathcal{M}(\lambda\eta)$ satisfies both (Tot) and (Ext).

1.4 Provably total functions

We intend to measure the proof-theoretic strength of all the systems treated in this thesis by ascertaining their provably total functions. We call an element f of the applicative structure total iff for each word input v, we have that fv is a word too. Observe that using $\epsilon, \mathsf{s}_0, \mathsf{s}_1$, we can produce a canonical closed L term $\overline{w}$ denoting w for each word w. In the following let $\mathcal{L}$ be a language extending our first-order language L. The notion of a *provably total function* is defined for an arbitrary $\mathcal{L}$ theory Th as follows.

Definition 4 *A function $F : \mathbb{W}^n \to \mathbb{W}$ is called provably total in an $\mathcal{L}$ theory* Th, *if there exists a closed* L *term t_F such that*

(i) $\mathsf{Th} \vdash t_F : \mathsf{W}^n \to \mathsf{W}$ *and, in addition,*

(ii) $\mathsf{Th} \vdash t_F\overline{w}_1 \cdots \overline{w}_n = \overline{F(w_1, \dots, w_n)}$ *for all $w_1, \dots, w_n$ in $\mathbb{W}$.*

So, the notion of a provably total function is divided into two conditions (i) and (ii). The first condition (i) expresses that t_F is a total operation from W^n to W, *provably in the $\mathcal{L}$ theory* Th. Condition (ii), on the other hand, claims that t_F indeed represents the given function $F : \mathbb{W}^n \to \mathbb{W}$, for each fixed tuple of words $\vec{w}$ in $\mathbb{W}^n$.

An example of a theory whose provably total functions are exactly the polynomial time computable ones is Strahm's PT [79].

1.5 Proof theoretic analysis of weak applicative systems

We make some general considerations about the analysis of (pure) applicative systems of strength lower than Primitive Recursive Arithmetic, abbreviated as PR in the following. We introduce the most important proof-theoretic tool for the analysis of these theories: Strahm's realisation approach. For overviews about proof-theoretic results for weak applicative theories extended by a truth predicate or types, we refer to sections 2.1 and 4.4, respectively.

The strength of applicative theories weaker than Primitive Recursive Arithmetic is usually measured by giving their provably total functions. We use the same measurement in this thesis. Because of their expressive strength, for weak applicative systems it is usually easy to prove the totality of the functions in the corresponding complexity class, which yields the lower bound. The upper bound proof is more difficult. In general, reductions of weak applicative systems to corresponding systems of bounded arithmetic seem to be impossible because of the Σ_1 completeness of application in the standard models. Therefore, if we translate e.g. the expression $s = t$ for L terms s, t into systems of bounded arithmetic, the resulting formula is Σ_1, independently of whether we use an interpretation according to the recursion theoretic - or the term model. But then, it is difficult to justify even very weak induction schemes. A system for which an embedding into a corresponding system of bounded arithmetic is possible thanks to its weak induction scheme[2] is Strahm's PTO [76].

[2]The schemes allow induction for formulas $fx = 0$, and $y \leq_{\mathsf{W}} fx \wedge gxy = 0$ for total functions f, g, respectively. The schemes are weak in the sense that they cannot be used to prove the totality of functions. Therefore, to justify bounded recursion, Strahm has to include a special recursor term defined analogously as $\mathbf{r}_N$ in Feferman and Jäger's BON

In his [78, 79], Strahm introduces a new realisation approach able to deal with induction schemes strong enough to justify bounded recursion. This allows him to characterise several polynomial and linear complexity classes in a very elegant way. Since then, Strahm's realisation technique or modifications and extensions thereof have been used to determine the upper bounds of several applicative systems, possibly extended by a truth predicate or types (see [18, 19, 61, 74]). In his [80], Strahm shows that his realisation approach proves the feasibility of the provably total *functionals* of his polynomial theory PT introduced in [79].

The idea of realisation goes back to the Brouwer-Heyting-Kolmogorov interpretation of intuitionistic logic. Buss introduced in [11, 12] a witnessing method for systems of bounded arithmetic. Cantini developed an extension of Strahm's approach to formulas containing the truth predicate in [18] [3]. In this thesis, upper bound results are mostly proved by modifications and generalisations of Strahm's and Cantini's approaches. This is the reason why we introduce Strahm's realisation approach in the following.

First, we fix the notation. In the context of realisers, $\langle \cdot, \cdots, \cdot \rangle$ denotes in the whole thesis the pairing function (on words) within the logarithmic hierarchy defined by Clote in [20]. It has the property that pairs of different arities are different for any arities. There should be no danger of confusion with $\mathsf{p}xy$ which is abbreviated in the same way. $< \cdot, \cdots, \cdot >$ denotes in the whole thesis a set theoretic pairing function. For both introduced pairing functions and an object w, we denote its components relative to the suitable pairing function by $w_1, \cdots, w_n$.

For Strahm's realisation approach, we work with a realisation relation $\mathfrak{R}$ that holds between individuals (words in our case) and *positive* formulas. If $w \; \mathfrak{R} \; A$ holds, we call w a realiser of A. Realisers intuitively give a reason, why the realised formula is true. In the following, we write $s = t$ if these terms are equal in the term model $\mathcal{TM}$. $\mathfrak{R}$ is defined recursively on the

[41, 42].

[3]Let us also mention Cantini's [16], which proves feasibility of a total version of PTO by introducing a realisation approach very different from Strahm's.

complexity of its formula argument as follows.

$$\begin{array}{llll}
\rho \ \mathfrak{R} \ \mathsf{W}(t) & \text{iff} & \overline{\rho} = t \\
\rho \ \mathfrak{R} \ (t_1 = t_2) & \text{iff} & \rho = \epsilon \\
\rho \ \mathfrak{R} \ (A \wedge B) & \text{iff} & \rho = \langle \rho_1, \rho_2 \rangle \wedge \rho_1 \ \mathfrak{R} \ A \text{ and } \rho_2 \ \mathfrak{R} \ B, \\
\rho \ \mathfrak{R} \ (A \vee B) & \text{iff} & \rho = \langle \rho_1, \rho_2 \rangle \text{ and either } \rho_1 = 0 \text{ and } \rho_2 \ \mathfrak{R} \ A \text{ or} \\
 & & \qquad\qquad\qquad\qquad \rho_1 = 1 \text{ and } \rho_2 \ \mathfrak{R} \ B, \\
\rho \ \mathfrak{R} \ (\forall x)A(x) & \text{iff} & \rho \ \mathfrak{R} \ A(u) \text{ for a fresh variable } u, \\
\rho \ \mathfrak{R} \ (\exists x)A(x) & \text{iff} & \rho \ \mathfrak{R} \ A(t) \text{ for some term } t.
\end{array}$$

The realisation relation can be generalised to sequences of positive formulas:

$$< \rho_1, \cdots, \rho_n > \ \mathfrak{R}\ A_1, \cdots, A_n :\Leftrightarrow \rho_1 \ \mathfrak{R}\ A_1, \cdots, \rho_n \ \mathfrak{R}\ A_n$$

A crucial property of $\mathfrak{R}$ is the trivialisation of quantifiers: Compared to the Brouwer-Heyting-Kolmogorov interpretation of intuitionistic logic the realiser of an existential quantifier does not deliver a witness, and the realiser of a formula $\forall x A[x]$ delivers a realiser of any $A[t]$ independently of t. Let us give justification for these trivialisations later.

Strahm's realisation approach can be best applied to systems formulated in a Gentzen-style sequent calculus, see e.g. Buss' [10] for the basic definitions. We let Γ, Δ denote sequencies of formulas, and Σ sequents, also given as $\Gamma \Rightarrow \Delta$. For $\Gamma \equiv A_1, \cdots, A_n$ and a substitution $[\vec{s}]$, we write $\Gamma[\vec{s}]$ or $A_1, \cdots, A_n[\vec{s}]$ for $A_1[\vec{s}], \cdots, A_n[\vec{s}]$ as long as misunderstandings are excluded.

We sketch the application of Strahm's realisation approach to a system formulated in Gentzen-style sequent calculus, where induction is formulated as a rule, and where the main formulas of all axioms and rules are positive. An example for a theory which allows such a formulation is Strahm's PT [79]. A standard cut elimination argument (see e.g. Buss [10] for explanations) yields that all cuts except the ones with positive cut formula can be eliminated. Since the totality of a function f can be expressed by the positive sequent $\vec{x} \in \mathsf{W} \Rightarrow f\vec{x} \in \mathsf{W}$, it has a proof containing only positive sequents. Now, the goal is to find iteratively a so called realisation function for each sequent of this proof, starting with its leaves. Let us define realisation functions: If $A_1, \cdots, A_n \Rightarrow D_1, \cdots, D_m$ is a positive sequent, we call a function $f : \mathbb{W}^n \rightarrow \mathbb{W}$ a realisation function of this sequent iff for any substitution $[\vec{s}]$ and any $\rho \in \mathbb{W}^n$ we have

$$\rho_1 \mathfrak{R} A_1[\vec{s}], \cdots, \rho_n \mathfrak{R} A_n[\vec{s}] \Rightarrow f(\rho)_2 \mathfrak{R} D_{f(\rho)_1}[\vec{s}]$$

Note, that the cut-elimination argument given above is crucial to realise positive theorems of the system since only positive formulas are realised. To get an upper bound result for the analysed theory, we have to deliver realisation functions in the corresponding complexity class $\mathcal{C}$. For PT we choose $\mathcal{C}$ to be the set of polynomial time computable functions. If we succeed in finding such realisation functions for all provable, positive sequents, an easy argument shows that all provably total functions of the theory lie within $\mathcal{C}$. In addition, Strahm's approach yields that for formulas of the form

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

with A being a positive formula the dependence of the witness y on x is given by a function within the complexity class $\mathcal{C}$. This is the analogon of Buss' realisation theorem for S^1_2 in [11], page 86 [4].

Let us now explain why we can realise equations and quantifiers trivially in Strahm's realisation approach. The trivialisation of equations is standard also for realisation approaches of bounded arithmetic, see e.g. Buss' [11]. Nevertheless, this may seem inadequate in an applicative setting since in the standard models $\mathcal{PR}$ and $\mathcal{TM}$ the property of equality for applicative terms is Σ_1 complete. Still, the trivialisation works because the axioms only force us to decide equality on words (cf. the axiom for case distinction), where it is decided using the given realisers for formulas of the form $t \in \mathsf{W}$. The trivialisation of the quantifiers works because the axioms give only very little information about arbitrary members of the universe, possibly not being words. Note that Strahm's realisation approach does not trivialise quantifiers

[4]Compared with Buss' realisation theorems, Strahm's approach only allows A to be a positive formula. For an intuitionistic version of PT, Cantini proves in [18] the above mentioned result for A being any formula by a realisation approach, which realises negative formulas using functionals.

restricted to words what allows to prove claims about the provably total functions of the analysed systems.

Chapter 2

A feasible theory of truth over combinatory logic

The contents of this chapter have been submitted for publication as [29].

2.1 Introduction

Weak theories of truth over an applicative setting have been analysed by Cantini in several papers. First examples of very expressive and natural truth theories of primitive recursive strength are presented in Cantini [14]. The proof-theoretic tools used in this article are the technique of asymmetric interpretation as well as subtle formalizations in the subsystem of Peano arithmetic with induction restricted to Σ^0_1 statements. In his more recent [18], Cantini studies a rich family of truth theories of primitive recursive strength including additional principles such as positive choice and uniformity. Special emphasis is put on the reduction of classical truth theories to their intuitionistic counterparts using a forcing relation. The computational content of the intuitionistic truth theories is analysed by means of suitable realisability techniques. A direct companion to [18] is Cantini [19], which deals with further extensions of the theories in [18]. In his [18] Cantini developed a theory of truth of polynomial strength as a guiding technical tool in order to deal with additional principles.

In this chapter, we introduce the theory T_{PT} which is the feasible analogon of

Cantini's primitive recursive theory of truth in [18]. $\mathsf{T_{PT}}$ extends $\mathsf{B}{\downarrow}$ and, as Cantini's theory, contains unrestricted truth induction and natural axioms for compositional truth. The only difference between the two theories is that $\mathsf{T_{PT}}$ reflects only elementhood in the words for terms that have smaller length than a given word. The main difference between $\mathsf{T_{PT}}$ and Cantini's above mentioned theory of polynomial strength is that $\mathsf{T_{PT}}$ allows unrestricted truth induction.

The idea to restrict the reflection of elementhood in the words in order to obtain weak theories was also used in explicit mathematics where types for the initial segments of the words were introduced by Spescha and Strahm in [73, 74] for their system PETJ. This system can indeed be seen as analogue of the theory $\mathsf{T_{PT}}$ in explicit mathematics, and will be discussed in detail in chapter 4.

That $\mathsf{T_{PT}}$ proves totality for all polynomial time computable functions follows easily using Cobham's well-known function algebra description (cf. [23, 20]) that characterises polynomial time as closure of initial functions under composition and bounded recursion. Therefore we will focus on the proof of the upper bound of $\mathsf{T_{PT}}$. A new realisation approach will be developed which uses a system of addresses and pointers to store and manipulate realisation information more efficiently than in Strahm's and Cantini's approaches.

We conclude the introduction with an outline of the chapter. In Section 2, we introduce the theory $\mathsf{T_{PT}}$ and discuss some of its theorems. The rest of the chapter is devoted to the upper bound proof of $\mathsf{T_{PT}}$. First, we introduce Cantini's realisation approach [18] which allowed him to find upper bounds for theories of truth with additional principles such as choice and uniformity. We do so because the new realisation approach is based on Cantini's approach. It will also be shown were Cantini's approach fails when it is applied to $\mathsf{T_{PT}}$, which also motivates the new approach. In Section 4, we give its technical details. We define a special set of words, construction descriptions, which are interpreted as coding realisation information and define functions to manipulate them.

In section 5, we show how this approach can be used to find the upper bound for an intuitionistic version of $\mathsf{T_{PT}}$. The restriction to intuitionistic logic allows us to present the ideas more transparently. Nevertheless, the approach could be easily adapted to deal with classical logic using Strahm's ideas in [79]. This is sketched in section 6. Finally, in section 7 we sketch an alternative upper bound proof for Arai's function algebra PCSF on sets using the realisation technique developed in this chapter.

Most of the work has to be done to realise the induction rule using bounded recursion. An important difference to realisations of other applicative theories of polynomial strength, such as PT introduced by Strahm in [79] or PETJ, is that the necessary bound cannot be constructed directly from the form of the induction formula and the realisation function for a special induction premise. Instead the bound must be established using bounding conditions which are proved to hold for all used realisation functions by induction on the depth of the corresponding proof.

2.2 The system $\mathsf{T_{PT}}$

The system $\mathsf{T_{PT}}$ contains a predicate T that mimics the properties of truth. The axiomatisation of this predicate relies on a coding mechanism for formulas. In the applicative framework, we code formulas using new constants designating logical operations.

2.2.1 The language L_{T} of positive truth

The (first order) language of $\mathsf{T_{PT}}$ is an extension of the language L by

- a new unary predicate symbol T for *truth*
- new individual constants $\dot{=}$, $\dot{\mathsf{W}}$, $\dot{\wedge}$, $\dot{\vee}$, $\dot{\exists}$, $\dot{\forall}$

The new constants allow the coding of positive formulas. We will use infix notation for $\dot{=}$, $\dot{\wedge}$ and $\dot{\vee}$.

2.2.2 The axioms and rules of $\mathsf{T_{PT}}$

The theory $\mathsf{T_{PT}}$ with language L_{T} is an extension of $\mathsf{B}{\downarrow}$ by compositional truth axioms and truth induction. Accordingly, its underlying logic is simply first order classical predicate logic.

Compositional truth

(C1) $\mathsf{T}(a \doteq b) \leftrightarrow a = b$

(C2) $a \in \mathsf{W} \rightarrow (\mathsf{T}(\dot{\mathsf{W}}ab) \leftrightarrow b \leq_{\mathsf{W}} a)$

(C3) $\mathsf{T}(a \dot{\vee} b) \leftrightarrow \mathsf{T}(a) \vee \mathsf{T}(b)$

(C4) $\mathsf{T}(a \dot{\wedge} b) \leftrightarrow \mathsf{T}(a) \wedge \mathsf{T}(b)$

(C5) $\mathsf{T}(\dot{\exists} a) \leftrightarrow \exists x \mathsf{T}(ax)$

(C6) $\mathsf{T}(\dot{\forall} a) \leftrightarrow \forall x \mathsf{T}(ax)$

Additionally, we have unrestricted truth induction.

Truth Induction

$$\mathsf{T}(a\epsilon) \wedge (\forall x \in \mathsf{W})(\mathsf{T}(ax) \rightarrow \mathsf{T}(a(\mathsf{s}_0 x)) \wedge \mathsf{T}(a(\mathsf{s}_1 x))) \rightarrow (\forall x \in \mathsf{W})(\mathsf{T}(ax))$$

2.2.3 Theorems of T_{PT}

Let us give the set of formulas for which the Tarski biconditionals hold.

Definition 5 (Simple L_{T} formulas) *Let A be a positive L_{T} formula and u be a variable not occurring in A. Then the formula A^u which is obtained by replacing each subformula of the form $t \in \mathsf{W}$ of A by $t \leq_{\mathsf{W}} u$ is called simple.*

Definition 6 *For each simple formula A^u of L_{T} we inductively define a term $\langle A \rangle$ whose free variables are exactly the free variables of A:*

$$\begin{array}{rcl}
\langle t = s \rangle & := & t \doteq s \\
\langle \mathsf{T}(t) \rangle & := & t \\
\langle s \leq_{\mathsf{W}} u \rangle & := & \dot{\mathsf{W}} us \\
\langle A \wedge B \rangle & := & \langle A \rangle \dot{\wedge} \langle B \rangle \\
\langle A \vee B \rangle & := & \langle A \rangle \dot{\vee} \langle B \rangle \\
\langle (\forall x) A \rangle & := & \dot{\forall}(\lambda x. \langle A \rangle) \\
\langle (\exists x) A \rangle & := & \dot{\exists}(\lambda x. \langle A \rangle)
\end{array}$$

The following lemma is proved by an easy external induction on the complexity of A.

Lemma 7 *Let A be a positive* L_{T} *formula. Then, we have*

$$\mathsf{T}_{\mathsf{PT}} \vdash u \in \mathsf{W} \rightarrow (\mathsf{T}(\langle A^u \rangle) \leftrightarrow A^u).$$

We can easily show that all polynomial time computable functions are provably total in T_{PT}. This is done by an external induction on the rank of the function relative to the usual function algebra description given by Clote in [20].

An interesting consequence of the biconditionals is a second recursion or fixed point theorem for positive, respectively simple predicates. This theorem can be obtained by lifting the fixed point theorem for combinatory logic (cf. Lemma 1) to the truth-theoretic language, cf. Cantini [13, 18].

2.2.4 Sequent style formulation of $\mathsf{T}^i_{\mathsf{PT}}$

As mentioned before, we will detail the upper bound proof for the intuitionistic version $\mathsf{T}^i_{\mathsf{PT}}$ of T_{PT} formulated in sequent style. The realisation approach is best formulated for systems in sequent style, and it is routine to formulate $\mathsf{T}^i_{\mathsf{PT}}$ or T_{PT} in this way. We can assume that the axioms contain only positive formulas and are conjunction free. Induction is formulated as a rule with positive main formulas in the usual way. Because of this restrictive formulation of the sequent calculus, a standard cut elimination argument yields the following lemma.

Lemma 8 *Let T be the theory T_{PT} or $\mathsf{T}^i_{\mathsf{PT}}$. Let Γ, D be a sequence of positive formulas such that $\mathsf{T} \vdash \Gamma \Rightarrow D$. Then there exists a T proof of $\Gamma \Rightarrow D$ that contains only positive, conjunction free formulas.*

2.3 The standard realisation approach

We denote by standard realisation approach the realisation technique executed in Cantini [18] for weak theories of truth, and in Strahm [79] for feasible applicative theories. We present a slight modification of the realisation relation the standard approach uses in detail because the new realisation relation presupposes it.

2.3.1 Cantini's realisation relation

Our variant of Cantini's realisation relation allows to discriminate realisers of different atoms, disjunctions and conjunctions. All relevant properties are unchanged by these modifications.

We will define the realisation relation with the help of an abstract derivability relation $d \vdash^m t$ where $d \in \mathbb{W}$, $m \in \omega$, and t is an arbitrary term, by means of a set of introduction rules, where m measures the length of proof. Assume that $\ulcorner = \urcorner, \ulcorner \mathsf{T} \urcorner, \ulcorner \mathsf{W} \urcorner, \ulcorner \wedge \urcorner, \ulcorner \vee \urcorner$ are different words. We write $s = t$ meaning that these terms are equal in $\mathcal{TM}$, analogously for $\leq$. Remember that $\langle \cdots \rangle$ denotes Clote's pairing function [20] within the logarithmic hierarchy.

- $\dot{=}$-rule
$$\frac{t = a \dot{=} b \quad a = b}{\langle \ulcorner \mathsf{T} \urcorner, \epsilon \rangle \vdash t}$$
- $\dot{\mathsf{W}}$-rule
$$\frac{t = \dot{\mathsf{W}} r s \quad s = \overline{\rho} \quad s \leq t}{\langle \ulcorner \mathsf{T} \urcorner, \rho \rangle \vdash t}$$
- $\dot{\vee}$-rule
$$\frac{t = r \dot{\vee} s \quad d \vdash r \quad (\text{or } d \vdash s)}{\langle \ulcorner \vee \urcorner, 0, d \rangle \vdash t \quad (\text{or } \langle \ulcorner \vee \urcorner, 1, d \rangle \vdash t)}$$
- $\dot{\wedge}$-rule
$$\frac{t = r \dot{\wedge} s \quad d \vdash r \quad e \vdash s}{\langle \ulcorner \wedge \urcorner, d, e \rangle \vdash t}$$
- $\dot{\forall}$-rule (assume $x \notin FV(rt)$)
$$\frac{t = \dot{\forall} r \quad d \vdash rx}{d \vdash t}$$
- $\dot{\exists}$-rule
$$\frac{t = \dot{\exists} r \quad d \vdash rq \text{ for some } q}{d \vdash t}$$

We abbreviate $(\exists m)(d \vdash^m t)$ by $d \vdash t$. Now we are in the position to give the realisation relation for all positive formulas of L_{T}.

$$
\begin{array}{lll}
\rho \ \mathfrak{R} \ \mathsf{T}(t) & \text{iff} & \rho \vdash t \\
\rho \ \mathfrak{R} \ \mathsf{W}(t) & \text{iff} & \rho = \langle \ulcorner \mathsf{W} \urcorner, \rho_1 \rangle \text{ and } t = \overline{\rho_1} \\
\rho \ \mathfrak{R} \ (t_1 = t_2) & \text{iff} & \rho = \langle \ulcorner \mathsf{W} \urcorner, \epsilon \rangle \text{ and } \rho_1 = \epsilon \text{ and } t_1 = t_2 \\
\rho \ \mathfrak{R} \ (A \wedge B) & \text{iff} & \rho = \langle \ulcorner \wedge \urcorner, \rho_1, \rho_2 \rangle \text{ and } \rho_1 \ \mathfrak{R} \ A \text{ and } \rho_2 \ \mathfrak{R} \ B, \\
\rho \ \mathfrak{R} \ (A \vee B) & \text{iff} & \rho = \langle \ulcorner \vee \urcorner, \rho_1, \rho_2 \rangle \text{ and either } \rho_1 = 0 \text{ and } \rho_2 \ \mathfrak{R} \ A \text{ or} \\
 & & \qquad \qquad \qquad \qquad \rho_1 = 1 \text{ and } \rho_2 \ \mathfrak{R} \ B, \\
\rho \ \mathfrak{R} \ (\forall x) A(x) & \text{iff} & \rho \ \mathfrak{R} \ A(u) \text{ for a fresh variable } u, \\
\rho \ \mathfrak{R} \ (\exists x) A(x) & \text{iff} & \rho \ \mathfrak{R} \ A(t) \text{ for some term } t.
\end{array}
$$

This definition assures that we can discriminate realisers of atoms of the form $\mathsf{W}(t)$ and $\mathsf{T}(\dot{\mathsf{W}} st)$, which is crucial for the new realisation approach.

2.3.2 Problems of the standard realisation approach

In the following, we derive a sequent for which there is no polynomial time computable realisation function relative to the standard approach. In T_{PT} we have totality and the λ-theorem holds because it includes B. Therefore, there is a closed term r which satisfies the following recursion equations for any $w \in \mathsf{W}$.

- $r(\epsilon) = 0 \dot{=} 0$
- $r(\mathsf{s}_i w) = r(w) \mathbin{\dot{\wedge}} r(w)$

Using logical and applicative axioms, $\mathsf{C}1$, $\mathsf{C}4$ and truth induction we get:

$$\mathsf{T}_{\mathsf{PT}} \vdash x \in \mathsf{W} \Rightarrow \mathsf{T}(rx)$$

But we can not find a (standard) polynomial time computable realisation function for this sequent: Internal as well as external conjunctions are realised (roughly) by a pair which contains the realisers of both conjuncts. Therefore, using natural assumptions about the pairing function, realisation functions of the above displayed sequent must grow exponentially.

2.3.3 Inefficiencies in the standard realisation approach

Two inefficiencies of the standard realisation approach, which are closely related, will be demonstrated in the following. We will overcome them using the new realisation approach.

Let us look first at the realisers of the formulas $\mathsf{T}(r\overline{w})$ for the function r defined as before and $w \in \mathbb{W}$. Intuitively, these realisers do not contain much information, they just contain, repeatedly paired, the information ϵ. The realisers only grow that fast in w because we ask for realisation information for each internal conjunct of each internal conjunction of $r\overline{w}$ even if two such conjuncts have always the same realiser. Our formalism will take advantage of this by allowing that the same piece of realisation information can be used for several (internal) subformulas.

Another closely related source of inefficiency in the standard realisation approach can be demonstrated for the realisation of the conclusion of the cut rule. Let the used cut rule have the following form.

$$\frac{\Gamma \Rightarrow A \qquad \Gamma, A \Rightarrow D}{\Gamma \Rightarrow D}$$

We assume realisation functions p and q for the premises. To produce a realiser of D, we will first produce realisation information for A, and add this information to the tuple of realisers of Γ. Then we will apply the realisation function q. This is inefficient because realisation information that is necessary for A may already be contained in the realisers of Γ. This means that we apply the realisation function q to an input that is larger than it has to be. The formalism developed in this section allows to use the same realisation information for the subformulas of *several* formulas in a sequence and therefore overcomes this inefficiency.

2.3.4 Sketch of the new approach

In the following, we roughly describe how the new approach works. We have as inputs and outputs of our realisation functions strings of information which are construction descriptions for standard realisers (that is realisers in the sense of $\mathfrak{R}$). In this strings, all information is stored under specific addresses and organised using a system of pointers. A string which describes the standard realiser of $\mathsf{T}(r\overline{w})$ for $w \in \mathbb{W}$ under address a_w will roughly have the following form.

$$\left[\begin{array}{l}
a_w \text{ stores a pair whose components are stored under } a_{\mathsf{pW}w}.\\
a_{\mathsf{pW}w} \text{ stores a pair whose components are stored under } a_{\mathsf{pW}(\mathsf{pW}w)}.\\
\dots\\
a_{\mathsf{pW}(\cdots(\mathsf{pW}w)\cdots)} \text{ stores a pair whose components are stored under}\\
a_{\mathsf{pW}[\mathsf{pW}(\cdots(\mathsf{pW}w)\cdots)]}.\\
\dots\\
a_\epsilon \text{ stores the content } \epsilon.
\end{array}\right]$$

Note that this construction description has polynomial size in w, in contrast to the standard realiser of $\mathsf{T}(r\overline{w})$, because it allows to use the same pieces of information for several internal subformulas. Note also that for all words $v \subseteq w$ the realiser of $\mathsf{T}(r\overline{v})$ is simultaneously stored at address v.

The price we pay for working with construction descriptions of standard realisers instead of the standard realisers themselves is that we get in polynomial time only a construction description of a realiser of a $D \in \Delta$. The actual construction of the standard realiser from a construction description using a fixed construction function could take exponential time. This has to be avoided, because it would spoil the upper bound proof of T_{PT}. Nevertheless, we have to construct standard realisers from construction descriptions to show that the realisation function f of sequents of the form $x \in \mathsf{W} \Rightarrow tx \in \mathsf{W}$ contains in some way the interpretation of t in the standard model. The solution is to define a polynomial time function which approximates the above mentioned construction function such that at least for construction descriptions of realisers of formulas of the form $t \in \mathsf{W}$ the two functions yield the same result.

2.4 The new formalism

Now, we present a formalisation of the ideas sketched in the previous section.

2.4.1 Construction descriptions

The construction descriptions (short: CDs) are finite sets of CD parts which are build from addresses, pointers, colons, and words. Nevertheless, CDs and all its components are just special words due to a natural polynomial time coding function which we silently assume. The CD parts and the addresses are denoted using the additional signs $., :, \to$.

Definition 9 (address, address head) *Assume that w is a word. Then w, $w.0$ and $w.1$ are addresses with address head w.*

We use $\breve{a}$, $\breve{b}$, $\breve{v}$, $\breve{w}$ to denote words that are intended to be addresses.

Definition 10 (CD part) *Let $\breve{a}$ and $\breve{b}$ be addresses. Let w be a word. A CD part can have the following three forms:*

- *$\breve{a} \to \breve{b}$ where the address head of $\breve{b}$ is contained in the address head of $\breve{a}$.*
- *$\breve{a} : \langle \ulcorner \mathsf{W} \urcorner, w \rangle$*
- *$\breve{a} : \langle \ulcorner \mathsf{T} \urcorner, w \rangle$*

The pairing with $\ulcorner \mathsf{T} \urcorner$ or $\ulcorner \mathsf{W} \urcorner$ has the purpose of discriminating realisation information for formulas of the form $\mathsf{T}(\dot{\mathsf{W}}st)$ and $t \in \mathsf{W}$. In the following we abbreviate $\breve{a} : \langle \ulcorner \mathsf{W} \urcorner, w \rangle$ as $\breve{a} : w$. $:$ or $\to$, respectively, separate the left and the right side of a CD part.

Definition 11 (CD) *A CD ρ is a finite set of CD parts where the following two conditions are fulfilled.*

- *If a word w is the left side of a CD part of ρ then neither $w.0$ nor $w.1$ are the left side of any CD part of ρ.*
- *No address occurs twice as the left side of a CD part of ρ.*

The two conditions guarantee the unambiguous construction of standard realisers from CDs. We display the CD $\{a_0, a_1, \cdots, a_n\}$ as $a_0/a_1/\cdots/a_n$. We use α and ρ for inputs which are intended to be CDs.

2.4.2 The realisation relation for CDs

In this section we show how CDs can realise sequences of formulas by defining the earlier mentioned construction function con. con translates construction descriptions, which are given by CDs, into standard realisers.

Definition 12 (Construction function con) *The function* $\mathsf{con} : \mathbb{W} \times \mathbb{W} \to \mathbb{W}$ *is given by the following algorithm to calculate* $\mathsf{con}(\rho, \breve{v})$*:*

If ρ is not a CD or $\breve{v}$ is not an address, we return a fixed word ε (error) which is not a realiser of any formula. In all other cases, we execute the following definition by cases.

Case 1 There is a CD part of the form $\breve{v} \to \breve{w}$:

$$\mathsf{con}(\rho, \breve{v}) := \mathsf{con}(\rho, \breve{w}).$$

Case 2 $\breve{v}.0$ and $\breve{v}.1$ occur as left sides of CD parts:

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner \wedge \urcorner, \mathsf{con}(\rho, \breve{v}.0), \mathsf{con}(\rho, \breve{v}.1) \rangle.$$

Case 3 Only $\breve{v}.i$ but not $\breve{v}.j$, for $0 \leq i, j \leq 1$ and $i \neq j$, occurs as left side of a CD part:

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner \vee \urcorner, i, \mathsf{con}(\rho, \breve{v}.i) \rangle.$$

Case 4 There is a CD part of the form $\breve{v} : w$:

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner \mathsf{W} \urcorner, \epsilon, w \rangle.$$

Case 5 There is a CD part of the form $\breve{v} : \langle \ulcorner \mathsf{T} \urcorner, w \rangle$:

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner \mathsf{T} \urcorner, \epsilon, w \rangle.$$

Case 6 Cases 1 until 5 are not satisfied. Then $\mathsf{con}(\rho, \breve{v}) := \varepsilon$.

The well-definedness of this function follows from the conditions imposed on CDs.

Example 13 *Let us use another time the function r on page 25. In section 2.3.3 we have written down a description for a standard realiser of $\mathsf{T}(r\overline{w})$ for $w \in \mathbb{W}$. In our formalism a CD α which describes this realiser has a similar form:*

$$\left.\begin{array}{l} w.0 \to \mathsf{p_W}(w)/w.1 \to \mathsf{p_W}(w)/ \\ \mathsf{p_W}(w).0 \to \mathsf{p_W}(\mathsf{p_W}(w))/\mathsf{p_W}(w).1 \to \mathsf{p_W}(\mathsf{p_W}(w))/ \\ \cdots \end{array}\right\} \textit{analogously } |w| \textit{ times}$$

$$\epsilon : \epsilon$$

Let us calculate $\mathsf{con}(\alpha, w)$. *(We suppress pairing with* $\ulcorner\wedge\urcorner$.*)*

$$\mathsf{con}(\alpha, w) = \langle \mathsf{con}(\alpha, w.0), \mathsf{con}(\alpha, w.1)\rangle = \langle \mathsf{con}(\alpha, \mathsf{p_W}(w)), \mathsf{con}(\alpha, \mathsf{p_W}(w))\rangle = \Big\langle \langle \mathsf{con}(\alpha, \mathsf{p_W}(w).0), \mathsf{con}(\alpha, \mathsf{p_W}(w).1)\rangle, \langle \mathsf{con}(\alpha, \mathsf{p_W}(w).0), \mathsf{con}(t, \mathsf{p_W}(w).1)\rangle \Big\rangle = \cdots$$

This calculation finally delivers the standard realiser of $\mathsf{T}(r\overline{w})$.

Based on the construction function, we define a realisation relation between CDs and sequences of formulas. The realisation is always relative to an address finder b which finds within a CD the address head which stores the standard realisation information for a certain formula. If the address finder b finds an address head with this information stored for each formula in a certain sequence, then the CD realises this sequence relative to b.

To give the formal definition, we use natural numbers i, n to denote words to increase readability. The natural number $n > 0$ denotes the word $\underbrace{00\cdots0}_{n \text{ times}}$.

Definition 14 (Realisation relation) *Let $A_1, \ldots, A_n$ be a sequence of positive formulas. Let $b : \mathbb{W}^2 \to \mathbb{W}$ be a polynomial time computable function. Then the following holds.*

$$\rho \ \mathfrak{r}_b \ A_1, \ldots, A_n \ \ :\Leftrightarrow \ \textit{For all } i \textit{ with } 1 \le i \le n : \mathsf{con}(\rho, b(\rho, i)) \ \ \mathfrak{R} \ \ A_i$$

From now on, in such a context, b is called an address finder. We call the words denoted by i with $1 \le i \le n$ its relevant inputs.

Note that only CDs can realise sequences of formulas since otherwise the construction function returns the error output ε. For the rest of the paper, we use the term realiser in the sense given above. It is easy to show the usual elementary properties for the above defined realisation relation since it is based on $\mathfrak{R}$ which has the same properties.

Lemma 15 *Let b be an address finder. Let $A_1, \dots, A_n$ be a sequence of positive formulas. We let $\vec{s} = \vec{t}$ abbreviate $s_1 = t_1 \wedge \cdots \wedge s_m = t_m$. Then the following assertions hold.*

- *$\rho\ \mathfrak{r}_b\ A_1, \dots, A_n[\vec{x}]$ implies $\rho\ \mathfrak{r}_b\ A_1, \dots, A_n[\vec{s}]$ for all $\vec{s}$.*
- *$\rho\ \mathfrak{r}_b\ A_1, \dots, A_n[\vec{s}]$ and $\mathcal{TM} \vDash \vec{s} = \vec{t}$ implies $\rho\ \mathfrak{r}_b\ A_1, \dots, A_n[\vec{t}]$ for all $\vec{s}, \vec{t}$.*

2.4.3 Technically important functions on CDs

As we sketched in the previous section, we need an additional construction function which is polynomial time computable. This function should correctly construct standard realisers of formulas of the form $t \in \mathsf{W}$. We will define a function that fulfils this task, but will also use it to bound realisation functions. In order to fulfil both tasks, we choose a more complex definition than one might expect.

Definition 16 (Related address relation) *The related address relation R^*_ρ is a binary relation on addresses dependent on a word ρ. We assume that R^*_ρ is empty if ρ is not a CD. In all other cases, R^*_ρ is the reflexive, transitive closure of the following relation R_ρ.*

$$\begin{aligned} R_\rho(\breve{v}, \breve{w}) :\Leftrightarrow\ & \breve{w} \text{ occurs on the left side of a CD part of } \rho \text{ and} \\ & (\breve{v} \to \breve{w} \text{ occurs in } \rho \quad \text{or} \\ & \breve{w} = \breve{v}.0 \text{ or } \breve{w} = \breve{v}.1) \end{aligned}$$

Definition 17 (con_W) *The function $\mathsf{con}_\mathsf{W} : \mathbb{W} \times \mathbb{W} \to \mathbb{W}$ is defined by the following algorithm for the calculation of $\mathsf{con}_\mathsf{W}(\rho, \breve{v})$:*

Step 1: Find all addresses $\breve{w}$ for which $R_\rho(\breve{v}, \breve{w})$ holds. They form a set M.

Step 2: Output the maximum with respect to the lexicographic ordering over all words u such that $\breve{w} : u$ is a part of ρ for a $\breve{w} \in M$. If the set we take the maximum over is empty, output ϵ.

Lemma 18 *The function* $\mathsf{con}_{\mathsf{W}}$ *is polynomial time computable.*

Proof. It can be checked in polynomial time if ρ is a CD. If not, $\mathsf{con}_{\mathsf{W}}(\rho, w)$ is evaluated immediately as ϵ. If ρ is a CD, for each address $\breve{v}$ occurring in ρ the addresses $\breve{w}$ with $R_\rho(\breve{v}, \breve{w})$ have to be searched at most once, and they can be found in polytime. The number of addresses occurring in ρ is bounded by the length of ρ [1]. This yields that the set M can be constructed in polynomial time relative to ρ. Then the required maximum can be found in polynomial time relative to M and ρ. $\Box$

Example 19 *Let* $\alpha := 000 \to 00/00.0 : 100/0000.1 : 11111/00.1 : 1/0 : \epsilon$. *Then* $\mathsf{con}_{\mathsf{W}}(\alpha, 000)$ *is the maximum of the set* $\{100, 1\}$.

The function $\mathsf{con}_{\mathsf{W}}$ can be interpreted as finding the maximal computational content stored under a certain address. Note that for a CD ρ that describes a realiser of a formula of the form $t \in \mathsf{W}$ under a certain address $\breve{v}$, $\mathsf{con}_{\mathsf{W}}(\rho, \breve{v})$ correctly constructs the value of t in $\mathcal{TM}$. This is so because exactly one address occurs in the set M we take the maximum over. Note that CD parts of the form $\breve{a} : \langle \ulcorner \mathsf{T} \urcorner, w \rangle$, that are used to realise formulas of the form $\mathsf{T}(\dot{\mathsf{W}}st)$, are ignored by $\mathsf{con}_{\mathsf{W}}$.

As we indicated before, the polynomial time computable construction function $\mathsf{con}_{\mathsf{W}}$ will be used also for technical reasons. The function W_b, which depends on $\mathsf{con}_{\mathsf{W}}$, helps to bound realisation functions.

Definition 20 (W_b) *Let b be an address finder. The function* $\mathsf{W}_b : \mathbb{W} \times \mathbb{W} \to \mathbb{W}$ *is defined as*

$$\mathsf{W}_b(w, \rho) := max\{\mathsf{con}_{\mathsf{W}}(\rho, b(\rho, v)) : \epsilon \subset v \subseteq w\}$$

W_b is polytime because $\mathsf{con}_{\mathsf{W}}$ is as well.

[1]We use here natural assumptions about the function coding CDs as words that is silently assumed.

Definition 21 ($\mathsf{con_T}$) *The function* $\mathsf{con_T} : \mathbb{W} \times \mathbb{W} \to \mathbb{W}$ *is defined as* $\mathsf{con_W}$ *with the only difference that it outputs the maximum over all words u such that $\breve{w} : \langle \ulcorner \mathsf{T} \urcorner, u\rangle$ is a part of ρ for a $\breve{w} \in M$. If M is empty, it also outputs ϵ.*

The function $\mathsf{con_T}$ is polynomial time computable for the same reasons as $\mathsf{con_W}$. The polynomial time computable functions defined below are important for technical reasons too. We always assume that they give the error output if their inputs are not as intended.

Definition 22 (Maximal address function) *The function* $\mathsf{MA} : \mathbb{W} \to \mathbb{W}$ *applied to a CD ρ returns its maximal address head with respect to the lexicographic order.*

Definition 23 ($\to$-path) *A $\to$-path in ρ is a sequence of addresses $\breve{a}_1, \cdots, \breve{a}_n$ such that for all $1 \leq i < n$ we have $\breve{a}_i \to \breve{a_{i+1}} \in \rho$.*

Definition 24 ($\to$-path-end function) *The function* $\downarrow : \mathbb{W} \times \mathbb{W} \to \mathbb{W}$ *applied to a CD ρ and an address $\breve{v}$ returns the address at the end of a maximal $\to$-path in ρ starting at $\breve{v}$. We suppress the first argument of $\downarrow$ if it is clear from the context.*

2.5 Applying the formalism to $\mathsf{T}^i_{\mathsf{PT}}$

From now on, we work with a sequent style formulation of $\mathsf{T}^i_{\mathsf{PT}}$ which we call $\mathsf{T}^i_{\mathsf{PT}}$ as well.

2.5.1 Stating the main claim

We use the following conventions and notations.

- Γ is always a sequence of positive formulas of the form $A_1, \ldots, A_n$. $|\Gamma|$ gives its length n.
- $\Gamma, A[\vec{s}]$ denotes $\Gamma[\vec{s}], A[\vec{s}]$.

- We often use $+$ and $\cdot$ instead of $*$ and $\times$. In such contexts natural numbers $n > 0$ abbreviate the word $\underbrace{00\cdots 0}_{n \text{ times}}$.
- The function W_b will always occur in connection with a sequence of formulas of a length n and we will always take $\underbrace{00\cdots 0}_{n \text{ times}}$ as its first argument. Therefore, we suppress it always.
- For a CD ρ, $\rho^{\ominus}$ denotes the CD produced by deleting the CD part of ρ which contains the maximal address head. $\rho^{\ominus}$ is the empty word if ρ is not a CD. This assures $w^{\ominus} \leq w$ for all words w [2].
- We write $value(t)$ for the word with $\mathcal{TM} \vDash t = \overline{value(t)}$ if there exists any.

Theorem 25 *Let Γ, D be a sequence of positive formulas. Assume that there is a proof of $\Gamma \Rightarrow D$ in $\mathsf{T}^i_{\mathsf{PT}}$ containing only positive formulas. Assume that a polytime address finder b is given. Then there exist polytime functions $p^{-1}, \delta, \kappa, \gamma$ (independent of b) and a polytime realisation function p_b such that for all $\vec{s}$ and for all α that are realisers of $\Gamma[\vec{s}]$ relative to b the following five properties hold:*

(1)
- $p^{-1}(p_b(\alpha)) = \alpha$.
- $p^{-1}(w) \leq w$ *for all* $w \in \mathbb{W}$.

(2) $p_b(\alpha)\ \mathfrak{r}_{b^*}\ \Gamma, D[\vec{s}]$, *where b^* is the following address finder.*

$$b^*(\rho, i) = \begin{cases} b(p^{-1}(\rho), i), & \text{if } 1 \leq i \leq |\Gamma| \\ \mathsf{MA}(\rho), & \text{if } i = |\Gamma| + 1 \\ \epsilon, & \text{else} \end{cases}$$

(3) $\mathsf{MA}(p_b(\alpha)) \leq \mathsf{MA}(\alpha) + \kappa(\mathsf{W}_b(\alpha))$.

(4) $p_b(\alpha) \leq \alpha + \delta(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha))$.

[2] Again, we use natural assumptions about the function coding CDs as words that is silently assumed.

(5) $\mathsf{W}_{b^*}(p_b(\alpha)) \leq \gamma(\mathsf{W}_b(\alpha))$.

(1) claims that we have an inverse function for the realisation function. The inverses can be defined because the realisation functions always add something to the already existing realiser (we will assume this tacitly in the whole realisation proof). The realisation functions will always store the new information under an address which is not used yet. This guarantees that we construct again a CD.

(2) claims that the application of the realisation function to a realiser of $\Gamma[\vec{s}]$ delivers a realiser of $\Gamma, D[\vec{s}]$ such that the standard realisers of the formulas of $\Gamma[\vec{s}]$ are constructed from the same address heads as before. The standard realiser of $D[\vec{s}]$ is constructed from the maximal address head. All realisation functions we use apply the address finder only to relevant inputs. Therefore, we will tacitly assume that for two address finders b and b' that fulfil $b(w,i) = b'(w,i)$ for all relevant inputs i and all words w, the same realisation functions are produced. This allows us to define address finders only for relevant inputs in the following.

(3) claims that we can control the length of the maximal address head. It is important that the bound depends only on $\mathsf{W}_b(\alpha)$ but not on $\mathsf{MA}(\alpha)$.

(4) and (5) make analogue statements for the whole realiser. The hidden first arguments in (5) are $\underbrace{00\cdots0}_{|\Gamma|+1 \text{ times}}$ or $\underbrace{00\cdots0}_{|\Gamma| \text{ times}}$, respectively.

We will prove the main theorem by simultaneous induction on the depth of the positive proof of $\Gamma \Rightarrow D$ in $\mathsf{T}^i_{\mathsf{PT}}$. The bounding properties 3 and 4 will be needed to deal with induction, property 5 for cut. Because it increases legibility, we will always find first the p_b-functions, and only then construct the other polytime functions $(p^{-1}, \delta, \kappa, \gamma)$. This is legitimate because these functions will always be constructed independently of b or p_b.

2.5.2 Realisation functions for the axioms

Let us show that for proof depth 0 the claim holds. We illustrate some interesting or explanatory examples.

Applicative axioms

Let us realise

$$\Gamma, s = t, s \in \mathsf{W}, t \in \mathsf{W} \Rightarrow \mathsf{d}_{\mathsf{W}}(p, q, s, t) = p$$

in full detail, the other applicative axioms can be realised similarly. Assume $\alpha\ \mathfrak{r}_b\ \Gamma, s = t, s \in \mathsf{W}, t \in \mathsf{W}[\vec{s}]$ for an address finder b [3]. This implies the existence of standard realisers for the main formulas relative to the substitution $[\vec{s}]$ and therefore $\mathcal{TM} \vDash \mathsf{d}_{\mathsf{W}}(p, q, s, t) = p[\vec{s}]$. This means that we can realise the succedent trivially and get the realiser we searched by adding the CD part $\mathsf{MA}(\alpha) + 1 : \epsilon$ to α. We define p_b as

$$p_b(\rho) := \rho/\mathsf{MA}(\rho) + 1 : \epsilon.$$

A function that satisfies the requirements of the inverse is p^{-1}, defined as

$$p^{-1}(\rho) := \rho^{\ominus}.$$

Let us check that 2 holds. Because p^{-1} is the inverse of p_b, we have for $1 \leq i \leq |\Gamma|$

$$b^*(p_b(\alpha), i) = b(\alpha, i).$$

Because of the assumption about α, this yields for $1 \leq i \leq |\Gamma|$

$$\mathsf{con}\Big(p_b(\alpha), b^*\big[p_b(\alpha), i\big]\Big)\ \ \mathfrak{R}\ \ A_i[\vec{s}].$$

To show yet is

$$\mathsf{con}\Big(p_b(\alpha), b^*\big[p_b(\alpha), |\Gamma| + 1\big]\Big)\ \ \mathfrak{R}\ \ \mathsf{d}_{\mathsf{W}}(p, q, s, t) = p[\vec{s}].$$

$b^*\big(p_b(\alpha), |\Gamma| + 1\big)$ is equal to $\mathsf{MA}(\alpha) + 1$. So $\mathsf{con}\Big(p_b(\alpha), b^*\big(p_b(\alpha), |\Gamma| + 1\big)\Big)$ is equal to ϵ. This delivers 2.

p_b increases the maximal address head of its argument only by one bit and the length of the information added by p_b can be bounded polynomially in $\mathsf{MA}(\alpha)$. Therefore, 3 and 4 are satisfied. To see that 5 is satisfied, let us calculate $\mathsf{W}_{b^*}(p_b(\alpha))$, which is the maximum of the set

$$\{\mathsf{con}_{\mathsf{W}}\Big(p_b(\alpha), b^*\big[p_b(\alpha), i\big]\Big) : 1 \leq i \leq |\Gamma| + 1\}.$$

[3] Even if s and $\vec{s}$ look related, they are completely independent. Similarly for t.

$\mathsf{W}_b(\alpha)$ is the maximum of the set

$$\{\mathsf{con}_{\mathsf{W}}\Big(\alpha, b\big[\alpha, i\big]\Big) : 1 \leq i \leq |\Gamma|\}.$$

Because of the definition of b^* and because p^{-1} is the inverse function of p_b, the two sets are identical except for the element

$$\mathsf{con}_{\mathsf{W}}\Big(p_b(\alpha), b^*\big[p_b(\alpha), |\Gamma| + 1\big]\Big).$$

But this equals ϵ. Therefore, the two maxima are the same. Other equation axioms can be realised analogously. We note that, given a correct inverse, to prove 2 and 5, we have only to check the content stored at the maximal address.

For axioms where we have to choose which disjunct is realised as e.g.

$$\Gamma, s \in \mathsf{W}, t \in \mathsf{W} \Rightarrow \mathsf{d}_{\mathsf{W}}(u, v, s, t) = v \vee s = t,$$

we read out the values of the given words (here $value(s)$ and $value(t)$) using $\mathsf{con}_{\mathsf{W}}$ and produce the suitable extension of the given realiser. A realisation function for the given axiom can be defined as follows.

$$f_b(\rho) := \begin{cases} \rho/\mathsf{MA}(\rho) + 1.0 : \epsilon, & \text{if } \mathsf{con}_{\mathsf{W}}(b(\rho, |\Gamma| + 1)) = \mathsf{con}_{\mathsf{W}}(b(\rho, |\Gamma| + 2)) \\ \mathsf{MA}(\rho) + 1.1 : \epsilon, & \text{else} \end{cases}$$

Finally, let us realise the following axiom.

$$\Gamma, s \in \mathsf{W}, t \in \mathsf{W} \Rightarrow s \times t \in \mathsf{W}$$

It is easy to see that the realisation function f_b will do the job.

$$f_b(\rho) := \rho/\mathsf{MA}(\rho) + 1 : \mathsf{con}_{\mathsf{W}}(b(\rho, length(\Gamma) + 1)) \times \mathsf{con}_{\mathsf{W}}(b(\rho, length(\Gamma) + 2))$$

Equation axioms

Since the existence of an $\mathfrak{r}_b$ realiser ρ for a sequence Γ implies the existence of $\mathfrak{R}$ -realisers for Γ, these axioms can be realised as

$$\Gamma, s = t, s \in \mathsf{W}, t \in \mathsf{W} \Rightarrow \mathsf{d}_{\mathsf{W}}(p, q, s, t) = p.$$

Compositional truth

To realise these axioms the use of pointers will be crucial not to violate 3 or 4, in contrast to the axioms realised before. We construct the realisation function for the following axiom.

$$\Gamma, \mathsf{T}(s \dot{\vee} t) \Rightarrow \mathsf{T}(s) \vee \mathsf{T}(t)$$

Assume $\alpha \ \mathfrak{r}_b \ \Gamma, \mathsf{T}(s\dot{\vee}t)[\vec{s}]$ for an address finder b. We are interested in the realisation information for $\mathsf{T}(s\dot{\vee}t)[\vec{s}]$. Because this formula is realised exactly as $\mathsf{T}(s) \vee \mathsf{T}(t)[\vec{s}]$, we have only to point to its address. We define p_b as

$$p_b(\rho) := \rho/\mathsf{MA}(\rho) + 1 \rightarrow b(\rho, |\Gamma| + 1).$$

A function p^{-1} that satisfies the requirements of the inverse can be defined as

$$p^{-1}(\rho) := \rho^{\ominus}.$$

By similar reasoning as before, one can show that properties 1 until 5 are satisfied. Observe that condition 3 and 4 might be violated if we would just reproduce the realisation information stored at $b(\rho, |\Gamma| + 1)$ instead of using a pointer.

Let us look now at the axiom

$$\Gamma, s \in \mathsf{W}, \mathsf{T}(\dot{\mathsf{W}}st) \Rightarrow t \in \mathsf{W}.$$

Let b and α be defined as before. The realisation function p_b can be defined as follows.

$$p_b(\rho) := \rho/\mathsf{MA}(\rho) + 1 : \mathsf{con}_\mathsf{T}(\rho, |\Gamma| + 2)$$

The realisation information of the formula $s \in \mathsf{W}$ does not occur in the realisation function, nevertheless the bound s for t is needed. Let us explain why.

The added realisation information for $t \in \mathsf{W}$ could increase the maximum of the computational content. Indeed, the value of t is already present in the realiser of the antecedent. But the function con_W that extracts computational content ignores CD parts of the form $\breve{a} : \langle \ulcorner \mathsf{T} \urcorner, w \rangle$. Therefore, only the presence of the realisation information for $s \in \mathsf{W}$ assures that conditions 4 and 5 are not violated in this case. This shows where our approach would fail for truth theories of the strength PRA with the additional axiom

$$\Gamma, \mathsf{T}(\dot{\mathsf{W}}t) \Rightarrow t \in \mathsf{W}.$$

2.5.3 Realisation functions for the conclusions of rules

$\vee$-rule left

Let the applied $\vee$-rule have the following form.

$$\frac{\Gamma, A \Rightarrow D \qquad \Gamma, B \Rightarrow D}{\Gamma, A \vee B \Rightarrow D}$$

By induction hypothesis, we have realisation functions p and q for both premisses. Assume $\alpha\ \mathfrak{r}_b\ \Gamma, A \vee B[\vec{s}]$ for an address finder b. We have to make a distinction by cases according to the disjunct of $A \vee B[\vec{s}]$ which is realised by α. Depending on this, we can produce from α a realiser α' of $\Gamma, A[\vec{s}]$ or of $\Gamma, B[\vec{s}]$. We will apply p to α^* in the first case and q in the second. To the result we add a marker which tells us which function has been applied. This allows to define an inverse function which works for both cases.

As we have described above, we will modify the input ρ before applying p or q. In the first case, the modified CD is $\rho/\mathsf{MA}(\rho){+}1 \rightarrow b(\rho, |\Gamma| + 1) \downarrow .0$, which we abbreviate as ρ_0 [4]. In the second case, it is $\rho/\mathsf{MA}(\rho){+}1 \rightarrow b(\rho, |\Gamma| + 1) \downarrow .1$, which we abbreviate as ρ_1. We abbreviate $b(\rho, |\Gamma| + 1) \downarrow$ as $\breve{a}_{A \vee B}$.

We find the realisation information contained in ρ_0 or ρ_1, respectively, by the following address finder b'.

$$b'(\rho, i) := \begin{cases} b(\rho^{\ominus}, i), & \text{if } 1 \leq i \leq |\Gamma| \\ \mathsf{MA}(\rho), & \text{if } i = |\Gamma| + 1 \end{cases}$$

We abbreviate $R_\alpha(\breve{a}_{A \vee B}, \breve{a}_{A \vee B}.0)$ by C. Now, we can define f_b as

$$f_b(\rho) := \begin{cases} p_{b'}(\rho_0)/\mathsf{MA}\big[p_{b'}(\rho_0)\big] + 1 : 0/\mathsf{MA}\big[p_{b'}(\rho_0)\big] + 2 \rightarrow \mathsf{MA}\big[p_{b'}(\rho_0)\big], & \text{if } C \\ q_{b'}(\rho_1)/\mathsf{MA}\big[q_{b'}(\rho_1)\big] + 1 : 1/\mathsf{MA}\big[q_{b'}(\rho_1)\big] + 2 \rightarrow \mathsf{MA}\big[q_{b'}(\rho_1)\big], & \text{else} \end{cases}$$

[4]See 24 for the definition of the function $\downarrow$

The marker, stored in the second largest address head, tells us whether p or q was applied. Therefore, we define f^{-1} as

$$f^{-1}(\rho) := \begin{cases} p^{-1}(\rho^{\ominus\ominus})^{\ominus}, & \text{if } \mathsf{con}_{\mathsf{W}}(\rho, \mathsf{MA}(\rho) - 1) = 0 \\ q^{-1}(\rho^{\ominus\ominus})^{\ominus}, & \text{else} \end{cases}$$

We have to show that this function works as an inverse of f_b when f_b is applied to a realiser α of $\Gamma, A \vee B[\vec{s}]$ relative to b. First, we assume that α realises the first disjunct of $A \vee B[\vec{s}]$. The definition of the realisation relation delivers

$$\alpha_0 \ \mathfrak{r}_{b'} \ \Gamma, A[\vec{s}].$$

Therefore, the induction hypothesis delivers $p^{-1}(p_{b'}(\alpha_0)) = \alpha_0$. Similarly, if α realises the second disjunct, we have $q^{-1}(q_{b'}(\alpha_1)) = \alpha_1$. Altogether, this immediately implies property 1.

Let us show that property 2 holds. Again we assume that α realises the first disjunct of $A \vee B[\vec{s}]$, the other case works similarly.

$$\alpha_0 \ \mathfrak{r}_{b'} \ \Gamma, A[\vec{s}]$$

implies because of the induction hypothesis for p

$$\mathsf{con}\Big(p_{b'}(\alpha_0), \mathsf{MA}\big[p_{b'}(\alpha_0)\big]\Big) \ \ \mathfrak{R} \ \ D[\vec{s}],$$

which yields property 2 because of the correctness of the inverse. Now, we prove property 3. Let us again assume that α realises the first disjunct of $A \vee B[\vec{s}]$, the other case works similarly. The induction hypothesis delivers

$$\mathsf{MA}(f_b(\alpha)) \leq \mathsf{MA}(\alpha_0) + \kappa_p(\mathsf{W}_{b'}(\alpha_0)) + 2.$$

(2 corresponds to the marker and the added copy.) Clearly, we have $\mathsf{W}_{b'}(\alpha_0) = \mathsf{W}_b(\alpha)$ and $\mathsf{MA}(\alpha_0) = \mathsf{MA}(\alpha) + 1$. Therefore, we get

$$\mathsf{MA}(f_b(\alpha)) \leq (\mathsf{MA}(\alpha) + 1) + \kappa_p(\mathsf{W}_b(\alpha)) + 2.$$

For the other case, the same bounding polynomial but with κ_p replaced by κ_q could be found. Therefore, for a polynomial bounding κ_p and κ_q property 3 is fulfilled. Property 4 is proved similarly.

Property 5 follows easily from $\mathsf{W}_b(\alpha) = \mathsf{W}_{b'}(\alpha_0)$ and the induction hypothesis for p and q.

$\vee$-rule right

Let the applied rule have the following form.

$$\frac{\Gamma \Rightarrow D}{\Gamma \Rightarrow D \vee E}$$

By induction hypothesis we have the realisation functions p_b for the premise. Obviously, the new realisation function can be defined as follows.

$$f_b(\rho) := p_b(\rho)/\mathsf{MA}(p_b(\rho)) + 1.0 \to \mathsf{MA}(p_b(\rho))$$

Using the suitable inverse function properties 1 until 5 follow immediately.

Cut

Let the applied cut rule have the following form.

$$\frac{\Gamma \Rightarrow A \qquad \Gamma, A \Rightarrow D}{\Gamma \Rightarrow D}$$

By induction hypothesis we have realisation functions p and q for the premisses. Assume $\alpha \; \mathfrak{r}_b \; \Gamma[\vec{s}]$ for an address finder b. We define the new realisation function as composition of p and q. First, we apply p_b to get a realiser of $\Gamma, A[\vec{s}]$ relative to a b'. Then apply $q_{b'}$ to get a realiser of $\Gamma, A, D[\vec{s}]$. This is the realiser we need relative to an address finder that just forgets the address that contains the realisation information for $A[\vec{s}]$. We define b' as follows.

$$b'(\rho, i) := \begin{cases} b(p^{-1}(\rho), i), & \text{if } 1 \le i \le |\Gamma| \\ \mathsf{MA}(\rho), & \text{if } i = |\Gamma| + 1 \end{cases}$$

We define f_b as

$$f_b(\rho) := q_{b'}(p_b(\rho)).$$

We define f^{-1} as

$$f^{-1}(\rho) := p^{-1}(q^{-1}(\rho)).$$

We have to show that this function works as an inverse of f_b when f_b is applied to a realiser α of $\Gamma[\vec{s}]$ relative to b. From the induction hypothesis 2 for p we get

(A) $$p_b(\alpha) \; \mathfrak{r}_{b'} \; \Gamma, A[\vec{s}].$$

Now, the induction hypothesis 1 for q delivers $q^{-1}(q_{b'}[p_b(\alpha)]) = p_b(\alpha)$. Therefore, the induction hypothesis 1 for p delivers property 1.

From (A), we get by induction hypothesis for 2

$$\mathsf{con}(q_{b'}\big[p_b(\alpha)\big], \mathsf{MA}(q_{b'}\big[p_b(\alpha)\big])\ \mathfrak{R}\ D[\vec{s}],$$

which implies property 2.

Let us prove now property 3. Because of the induction hypothesis for 5, we have $\mathsf{W}_{b'}(p_b(\alpha)) \leq \gamma_p(\mathsf{W}_b(\alpha))$. Using induction hypothesis 3, we have additionally

$$\begin{aligned}\mathsf{MA}(f_b(\alpha)) \leq & \mathsf{MA}(p_b(\alpha)) + \kappa_q(\mathsf{W}_{b'}(p_b(\alpha))) \leq \\ & \mathsf{MA}(\alpha) + \kappa_p(\mathsf{W}_b(\alpha)) + \kappa_q(\mathsf{W}_{b'}(p_b(\alpha))) \leq \\ & \mathsf{MA}(\alpha) + \kappa_p(\mathsf{W}_b(\alpha)) + \kappa_q(\gamma_p(\mathsf{W}_b(\alpha)))).\end{aligned}$$

Property 4 is proved similarly.

Let us show now property 5. By induction hypothesis 5, the following two inequations hold.

$$\mathsf{W}_{b'}(p_b(\alpha)) \leq \gamma_p(\mathsf{W}_b(\alpha))$$

$$\mathsf{con}_{\mathsf{W}}(q_{b'}(p_b(\alpha)), \mathsf{MA}(q_{b'}(p_b(\alpha)))) \leq \gamma_q(\mathsf{W}_{b'}(p_b(\alpha)))$$

Therefore, we have for the composition $\gamma_q \circ \gamma_p$

$$\mathsf{W}_{b^*}(q_{b'}(p_b(\alpha))) \leq (\gamma_q \circ \gamma_p)(\mathsf{W}_b(\alpha)).$$

Structural -, and quantifier rules

For contraction, we use that $\rho\ \mathfrak{r}_b\ \Gamma, A$ implies $\rho\ \mathfrak{r}_{b'}\ \Gamma, A, A$ where b' is defined as follows.

$$b'(\rho, i) = \begin{cases} b(\rho, i), & \text{if } 1 \leq i \leq \mathit{length}(\Gamma) \\ b(\rho, \mathit{length}(\Gamma)), & \text{if } i = \mathit{length}(\Gamma) + 1 \end{cases}$$

To realise commutation, and weakening, we modify the given address finder b slightly, similarly as above.

The quantifier rules are realised very easily, using lemma 15.

Induction

Let the applied induction rule have the following form.

$$\frac{\Gamma \Rightarrow \mathsf{T}(r\epsilon) \qquad \Gamma, \mathsf{T}(rx), x \in \mathsf{W} \Rightarrow \mathsf{T}(r(\mathsf{s}_i x))}{\Gamma, t \in \mathsf{W} \Rightarrow \mathsf{T}(rt)}$$

By induction hypothesis we have realisation functions p, q_0 and q_1 for the premisses.

As usual, we use recursion to define the realisation function. The main obstacle is to deliver the necessary bound, which will be produced using induction hypotheses 3 and 4.

The recursion works roughly in the following way: Given a realiser α of $\Gamma, t \in \mathsf{W}[\vec{s}]$ relative to b, we get by applying p_b to α a realiser of $\Gamma, \mathsf{T}(r\epsilon)[\vec{s}]$ relative to a b_1. When we add to $p_b(\alpha)$ a suitable CD part, we get a realiser of $\Gamma, \mathsf{T}(r\epsilon), \epsilon \in \mathsf{W}[\vec{s}]$ relative to a b_2. We can apply the functions $(q_0)_{b_2}$ or $(q_1)_{b_2}$ to get a realiser of $\Gamma, \mathsf{T}(r\overline{0})[\vec{s}]$ or $\Gamma, \mathsf{T}(r\overline{1})[\vec{s}]$ relative to a b_3. Then again, by adding a suitable CD part, we get a realiser of for example $\Gamma, \mathsf{T}(r\overline{0}), \overline{0} \in \mathsf{W}[\vec{s}]$ relative to a b_4 and can apply the functions $(q_0)_{b_4}$ or $(q_1)_{b_4}$ to get a realiser of for example $\Gamma, \mathsf{T}(r\overline{00})[\vec{s}]$. This process can be iterated arbitrary often and will deliver after $|value(t[\vec{s}])|$ many iterations the searched realiser.

Nevertheless, two problems have to be solved yet:

1. We have to use always the same recursion step functions. Therefore, we need an address finder $\tilde{b}$ such that for each $w \in \mathbb{W}$, after $|w|$ many recursion steps we still have a realiser of $\Gamma, \mathsf{T}(r\overline{w}), \overline{w} \in \mathsf{W}[\vec{s}]$ relative to $\tilde{b}$.

2. We have to deliver a bound for the sketched recursion.

Our strategy is to define first a binary function f. Its first argument is considered to be a realiser of $\Gamma, t \in \mathsf{W}[\vec{s}]$, the length of the second argument gives the number of iterations of the above described process. Later, from this binary function, we easily define the realisation function.

Let us tackle now the first problem for the above sketched binary function. The $(q_i)_{\tilde{b}}$ which we will apply in the recursion step always ask for the realisation information for $\Gamma[\vec{s}]$, that is stored in the first argument of the function. Therefore $\tilde{b}$ relies on an inverse of f which we define below.

Definition 26 *The function* $f^{-1} : \mathbb{W} \times \mathbb{W} \rightarrow \mathbb{W}$ *is defined by recursion as follows.*

$$\begin{array}{rcl} f^{-1}(\rho, \epsilon) & := & p^{-1}(\rho^{\ominus}) \\ f^{-1}(\rho, \mathsf{s}_i w) & := & f^{-1}(q_i^{-1}(\rho^{\ominus}), w) \end{array}$$

This function is clearly polynomial time computable since it can be given by a recursion bounded by ρ.

Definition 27 *Assume that* b *is an address finder. The function* $\tilde{b} : \mathbb{W} \times \mathbb{W} \rightarrow \mathbb{W}$ *is given by the following definition of cases.*

$$\tilde{b}(\rho, i) = \begin{cases} b\Big(f^{-1}\big[\rho, \mathsf{con}_{\mathsf{W}}(\rho, \mathsf{MA}(\rho))\big], i\Big), & \textit{if } 1 \leq i \leq \textit{length}(\Gamma) \\ \mathsf{MA}(\rho) - 1, & \textit{if } i = \textit{length}(\Gamma) + 1 \\ \mathsf{MA}(\rho), & \textit{if } i = \textit{length}(\Gamma) + 2 \end{cases}$$

We use that the realisation information of the formulas of the form $\mathsf{T}(r\overline{w})[\vec{s}]$ and $\overline{w} \in \mathsf{W}[\vec{s}]$ is always stored at the largest and second largest address head. Using this function the earlier mentioned function f_b can be defined.

Definition 28 *The function* $f_b : \mathbb{W} \times \mathbb{W} \rightarrow \mathbb{W}$ *is defined by recursion as follows.*

$$\begin{array}{rcl} f_b(\rho, \epsilon) & := & p_b(\rho)/\mathsf{MA}(p_b(\rho)) + 1 : \epsilon \\ f_b(\rho, \mathsf{s}_i w) & := & (q_i)_{\tilde{b}}(f_b(\rho, w))/\mathsf{MA}\Big((q_i)_{\tilde{b}}\big(f_b(\rho, w)\big)\Big) + 1 : \mathsf{s}_i w \end{array}$$

Example 29 *Let us give now concrete examples for the above defined functions. We look another time at the function* r *which was defined at page 25 and the sequent*

$$x \in \mathsf{W} \Rightarrow \mathsf{T}(rx),$$

which cannot be realised by a polytime function using the standard realisation approach. It can be derived by induction as follows.

$$\frac{\Rightarrow \mathsf{T}(r\epsilon) \qquad \mathsf{T}(rx), x \in \mathsf{W} \Rightarrow \mathsf{T}(r(\mathsf{s}_i x))}{t \in \mathsf{W} \Rightarrow \mathsf{T}(rt)}$$

So, if we deliver realisation functions for the premisses, we can use the above defined functions to construct a realisation function f *for the conclusion. We* *will construct f_{Id} and use premise realisation functions for suitable address finders, but note that everything is independent of address finders since there are no side formulas. We will use a realisation function p_{Id} for the first premise, e.g.*

$$p_{Id}(\rho) := \rho/\mathsf{MA}(\rho) + 1 : \epsilon.$$

We also use the realisation functions $(q_i)_{\tilde{Id}}$ for the induction step premisses, where $\tilde{Id}$ is the following function (note that Id has no relevant inputs).

$$\tilde{Id}(\rho, i) = \begin{cases} \mathsf{MA}(\rho) - 1, & \textit{if } i = 1 \\ \mathsf{MA}(\rho), & \textit{if } i = 2 \end{cases}$$

Realisation functions $(q_i)_{\tilde{Id}}$ for the induction step can be given as

$$(q_i)_{\tilde{Id}}(\rho) := \rho/\mathsf{MA}(\rho) + 1.0 \rightarrow \mathsf{MA}(\rho) - 1/\mathsf{MA}(\rho) + 1.1 \rightarrow \mathsf{MA}(\rho) - 1.$$

Let us now calculate $f_{Id}(\rho, w)$ for $\rho, w \in \mathbb{W}$ with f defined as in definition 28. We get

- $f_{Id}(\rho, \epsilon) = \rho/\mathsf{MA}(\rho) + 1 : \epsilon/\mathsf{MA}(\rho) + 2 : \epsilon$
- $f_{Id}(\rho, 0) = \rho/\mathsf{MA}(\rho) + 1 : \epsilon/\mathsf{MA}(\rho) + 2 : \epsilon/\mathsf{MA}(\rho) + 3.0 \rightarrow \mathsf{MA}(\rho) + 1/\mathsf{MA}(\rho) + 3.1 \rightarrow \mathsf{MA}(\rho) + 1/\mathsf{MA}(\rho) + 4 : 0$
- $f_{Id}(\rho, 00) = \rho/\mathsf{MA}(\rho) + 1 : \epsilon/\mathsf{MA}(\rho) + 2 : \epsilon/\mathsf{MA}(\rho) + 3.0 \rightarrow \mathsf{MA}(\rho) + 1/\mathsf{MA}(\rho) + 3.1 \rightarrow \mathsf{MA}(\rho) + 1/\mathsf{MA}(\rho) + 4 : 0/\mathsf{MA}(\rho) + 5.0 \rightarrow \mathsf{MA}(\rho) + 3/\mathsf{MA}(\rho) + 5.1 \rightarrow \mathsf{MA}(\rho) + 3/\mathsf{MA}(\rho) + 6 : 00$
- $\cdots$
- $f_{Id}(\rho, \underbrace{00 \cdots 00}_{n\ times}) = \rho/\mathsf{MA}(\rho) + 1 : \epsilon/\mathsf{MA}(\rho) + 2 : \epsilon/\mathsf{MA}(\rho) + 3.0 \rightarrow \mathsf{MA}(\rho) + 1/\mathsf{MA}(\rho) + 3.1 \rightarrow \mathsf{MA}(\rho) + 1/\mathsf{MA}(\rho) + 4 : 0/\mathsf{MA}(\rho) + 5.0 \rightarrow \mathsf{MA}(\rho) + 3/\mathsf{MA}(\rho) + 5.1 \rightarrow \mathsf{MA}(\rho) + 3/\mathsf{MA}(\rho) + 6 : 00/\cdots/\mathsf{MA}(\rho) + (2n+1).0 \rightarrow \mathsf{MA}(\rho) + (2n-1)/\mathsf{MA}(\rho) + (2n+1).1 \rightarrow \mathsf{MA}(\rho) + (2n-1)/\mathsf{MA}(\rho) + (2n+2) : \underbrace{00 \cdots 00}_{ntimes}$

(Analogously for arbitrary words of the same length as second argument.) It can be easily seen that $f_{Id}(\rho, w)$ is a realiser of $\mathsf{T}(r\overline{w}), \overline{w} \in \mathsf{W}$ relative

to $\tilde{Id}$ for any $\rho, w \in \mathbb{W}$. The function f_{Id} is polytime because of its small growth. How do we get from f_{Id} a realisation function $\mathfrak{f}_b$ for the sequent $t \in \mathsf{W} \Rightarrow \mathsf{T}(rt)$? The realisation information for $t \in \mathsf{W}[\vec{s}]$ tells us how many and which recursion steps have to take place which delivers the second argument for f_{Id}. Therefore, we get a realisation function $\mathfrak{f}_b$ for the sequent as

$$\mathfrak{f}_b(\rho) := f_{Id}\Big(\rho, \mathsf{con}_\mathsf{W}\big[\rho, b(\rho,1)\big]\Big).$$

To put the realiser of the formula $\mathsf{T}(rt)$ to the last position, we use a copy.

In the following, we will show how to find the realisation function $\mathfrak{f}_b$ for arbitrary conclusions of the induction rule. The additional difficulty is that in general the function f_b is not polytime. Usually, we have to control the recursion with a bound.

The next lemma claims the correctness of the function f and of its inverse f^{-1}.

Lemma 30 *Let α be a realiser of $\Gamma, t \in \mathsf{W}[\vec{s}]$ relative to b. Then for each $w \in \mathbb{W}$ (A) and (B) hold.*

(A) $f_b(\alpha, w)\ \mathfrak{r}_{\tilde{b}}\ \Gamma, \mathsf{T}(r\overline{w}), \overline{w} \in \mathsf{W}[\vec{s}]$

(B) $f^{-1}(f_b(\alpha, w), w) = \alpha$

Proof. We show (A) and (B) by simultaneous induction on w. If w equals ϵ, both claims follow immediately from properties 1 and 2 for p.

Let us switch to an $\mathsf{s}_i w \in \mathbb{W}$. The induction hypothesis for (A) delivers

$$f_b(\alpha, w)\ \mathfrak{r}_{\tilde{b}}\ \Gamma, \mathsf{T}(r\overline{w}), \overline{w} \in \mathsf{W}[\vec{s}].$$

Therefore property 1 for q_i implies

$$(q_i)^{-1}(f_b(\alpha, \mathsf{s}_i w)^{\ominus}) = f_b(\alpha, w).$$

Together with the induction hypothesis for (B), this delivers (B) for $\mathsf{s}_i w$.

Property 2 of q_i and the induction hypothesis for (A) imply that the maximal address head of $(q_i)_{\tilde{b}}(f_b(\alpha, w))$ contains the realisation information for

$\mathsf{T}(r(\mathsf{s}_i\overline{w}))$. It follows that the second largest - and largest address head of $f_b(\alpha, \mathsf{s}_i w)$ contain the realisation information for $\mathsf{T}(r(\mathsf{s}_i\overline{w}))$ and $\mathsf{s}_i\overline{w} \in \mathsf{W}$, respectively. Together with these facts, (B) for $\mathsf{s}_i w$ and

$$\mathsf{con}_{\mathsf{W}}\Big[f_b(\alpha, \mathsf{s}_i w), \mathsf{MA}\big(f_b(\alpha, \mathsf{s}_i w)\big)\Big] = \mathsf{s}_i w$$

deliver

$$f_b(\alpha, \mathsf{s}_i w) \ \mathfrak{r}_{\tilde{b}} \ \Gamma, \mathsf{T}(r(\mathsf{s}_i\overline{w})), \mathsf{s}_i\overline{w} \in \mathsf{W}[\vec{s}],$$

which finishes the proof. $\square$

To bound the function f_b by a polynomial for first arguments that realise $\Gamma, t \in \mathsf{W}[\vec{s}]$, it will be necessary to bound the values of $\mathsf{W}_{\tilde{b}}(f_b(\alpha, w))$ for $w \in \mathbb{W}$. This is so, because the length of the added parts in each recursion step of f_b depends polynomially on $\mathsf{W}_{\tilde{b}}(f_b(\alpha, w))$ for a certain $w \in \mathbb{W}$.

Lemma 31 *Let α be a realiser of $\Gamma, t \in \mathsf{W}[\vec{s}]$ relative to b and let $w \in \mathbb{W}$ be less or equal value$(t[\vec{s}])$. Then we have*

$$\mathsf{W}_{\tilde{b}}(f_b(\alpha, w)) \leq \mathsf{W}_b(\alpha)$$

Proof. Let us calculate $\mathsf{W}_{\tilde{b}}(f_b(\alpha, w))$. Because of lemma 30, we have for $1 \leq i \leq |\Gamma|$

$$\tilde{b}(f_b(\alpha, w), i) = b(\alpha, i).$$

Therefore, the content at these addresses does not violate the inequation. Let us look at the $|\Gamma| + 1$-th relevant address. Because of lemma 30, we have

$$\mathsf{con}\Big(f_b(\alpha, w), \tilde{b}\big(f_b(\alpha, w), |\Gamma| + 1\big)\Big) \ \mathfrak{R} \ \mathsf{T}(r\overline{w}).$$

Because of the stipulation that CD parts of the form $\breve{w} : \langle \ulcorner \mathsf{T} \urcorner, k\rangle$ do not contribute to the computational content, we have

$$\mathsf{con}_{\mathsf{W}}\Big(f_b(\alpha, w), \tilde{b}\big(f_b(\alpha, w), |\Gamma + 1|\big)\Big) = \epsilon.$$

Because we have $w \leq value(t[\vec{s}])$ also the realisation information stored at the $|\Gamma| + 2$-th relevant address does not violate the inequation. $\square$

The lemma we just proved allows to find bounding polynomials for $f_b(\alpha, w)$ and $\mathsf{MA}(f_b(\alpha, w))$ for suitably chosen α and w.

Lemma 32 *There is a polynomial $\kappa_f : \mathbb{W} \to \mathbb{W}$ such that for all address finders b, all $\vec{s}$, all realisers α of $\Gamma, t \in \mathsf{W}[\vec{s}]$ relative to b, and all $w \leq$ value$(t[\vec{s}])$, we have*

$$\mathsf{MA}(f_b(\alpha, w)) \leq \mathsf{MA}(\alpha) + \kappa_f(\mathsf{W}_b(\alpha)).$$

Proof. Because property 3 holds for p, q_0, q_1, we have MA-bounding polynomials $\kappa_p, \kappa_{q_0}, \kappa_{q_1}$. Let κ_q be a polynomial that bounds κ_{q_0} and κ_{q_1}. Using the properties of the bounding functions, we derive

$$\mathsf{MA}(f_b(\alpha, w)) \leq \mathsf{MA}(\alpha) + \kappa_p(\mathsf{W}_b(\alpha)) + 1 + \sum_{v \subset w} \Big(\kappa_q\big[\mathsf{W}_{\tilde{b}}(f_b(\alpha, v))\big] + 1\Big).$$

Using lemma 31, we get

$$\mathsf{MA}(f_b(\alpha, w)) \leq \mathsf{MA}(\alpha) + \kappa_p(\mathsf{W}_b(\alpha)) + 1 + \kappa_q(\mathsf{W}_b(\alpha)) \cdot w + w.$$

This implies our claim because we have $w \leq \mathsf{W}_b(\alpha)$. $\square$

Lemma 33 *There is a polynomial $\delta_f : \mathbb{W} \to \mathbb{W}$ such that for all address finders b, all $\vec{s}$, all realisers α of $\Gamma, t \in \mathsf{W}[\vec{s}]$ relative to b, and all $w \leq$ value$(t[\vec{s}])$, we have*

$$f_b(\alpha, w) \leq \alpha + \delta_f(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)).$$

Proof. Because property 4 holds for p, q_0, q_1 by induction hypothesis, we have bounding polynomials $\delta_p, \delta_{q_0}, \delta_{q_1}$. Let δ_q be a polynomial that bounds δ_{q_0} and δ_{q_1}. The CD parts of the form $M : w$ we add in the course of the recursion after using an induction premise function can be bounded by a polynomial h in $\mathsf{MA}(\alpha)$ and $\mathsf{W}_b(\alpha)$ because of lemma 32. Altogether, this implies

$$\begin{aligned} f_b(\alpha, w) \leq \alpha + \delta_p\big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big) + h\big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big) + \\ \sum_{v \subset w} \left(\delta_q\Big(\mathsf{W}_{\tilde{b}}\big[f_b(\alpha, v)\big], \mathsf{MA}\big[f_b(\alpha, v)\big]\Big) + h\big[\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big]\right). \end{aligned}$$

Using lemmas 31 and 32 we derive

$$\begin{aligned} f_b(\alpha, w) \leq \alpha + \delta_p\big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big) + h\big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big) + \\ \sum_{v \subset w} \left(\delta_q\Big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha) + \kappa_f(\mathsf{W}_b(\alpha))\Big) + h\big[\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big]\right). \end{aligned}$$

The summands are not dependent on the sum variable v, so we get

$$f_b(\alpha, w) \leq \alpha + \delta_p\big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big) + h\big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big) + \\ w \cdot \left(\delta_q\Big(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha) + \kappa_f(\mathsf{W}_b(\alpha))\Big) + h\big[\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha)\big]\right).$$

This implies our claim because we have $w \leq \mathsf{W}_b(\alpha)$. $\square$

Now, using the binary function f_b, we can define the realisation function $\mathfrak{f}_b$ in the following way. First, we define a polytime variant $\hat{f}_b(\rho, v)$ of $f_b(\rho, v)$ by bounded recursion with bound $\rho + \delta_f(\mathsf{W}_b(\rho), \mathsf{MA}(\rho))$. Because of the previous lemma, $\hat{f}_b(\rho, v)$ equals $f_b(\rho, v)$ if ρ is a realiser of $\Gamma, t \in \mathsf{W}[\vec{s}]$ relative to b and v smaller or equal $value(t[\vec{s}])$.

To get a unary realisation function, we use realisation information for the formula $t \in \mathsf{W}[\vec{s}]$ stored in ρ to determine $value(t[\vec{s}])$. This is the missing second argument of $\hat{f}_b$. Therefore, we define the unary $h_b(\rho)$ as

$$\hat{f}_b\Big(\rho, \mathsf{con}_\mathsf{W}\big[\rho, b(\rho, |\Gamma| + 1)\big]\Big).$$

h_b delivers the realisation information for $\mathsf{T}(rt)[\vec{s}]$, but not under the maximal address head. Therefore, we use a copy and define the realisation function $\mathfrak{f}_b$ as

$$\mathfrak{f}_b(\rho) := h_b(\rho)/\mathsf{MA}(h_b(\rho)) + 1 \rightarrow \mathsf{MA}(h_b(\rho)) - 1.$$

It can be seen immediately that all components of the function $\mathfrak{f}_b$ are polytime. Therefore, the following holds.

Lemma 34 *The function $\mathfrak{f}_b$ is polytime for any address finder b.*

To finish the proof of the main claim, we have to show that properties 1 until 5 hold for $\mathfrak{f}_b$.

For property 1, we have to define an inverse function for $\mathfrak{f}_b$ which must be correct for realiser inputs. For a realiser α of $\Gamma, t \in \mathsf{W}[\vec{s}]$ relative to b, we have because of lemma 33

$$\hat{f}_b\Big(\alpha, \mathsf{con}_\mathsf{W}\big[\alpha, b(\alpha, |\Gamma| + 1)\big]\Big) = f_b\Big(\alpha, \mathsf{con}_\mathsf{W}\big[\alpha, b(\alpha, |\Gamma| + 1)\big]\Big).$$

Therefore, using lemma 30, we get a correct inverse $\mathfrak{f}^{-1}$ defined as follows.

$$\mathfrak{f}^{-1}(\rho) = f^{-1}\Big(\rho^{\ominus}, \mathsf{con}_{\mathsf{W}}\big[\rho^{\ominus}, \mathsf{MA}(\rho^{\ominus})\big]\Big)$$

Lemmas 30 and 33 imply property 2. Property 3 follows from 32. Property 4 follows immediately from the definition of $\mathfrak{f}_b$. Property 5 follows because the formula which is realised additionally is a T-formula. This concludes the proof of the main theorem 25. The feasibility of $\mathsf{T}^i_{\mathsf{PT}}$ follows now as a corollary.

Corollary 35 (of theorem 25) *The provably total functions of* $\mathsf{T}^i_{\mathsf{PT}}$ *are exactly the polynomial time computable functions.*

Proof. Assume that the function $F : \mathbb{W} \to \mathbb{W}$ is provably total in $\mathsf{T}^i_{\mathsf{PT}}$ [5]. Therefore, for a corresponding closed t_F, we have

$$\mathsf{T}^i_{\mathsf{PT}} \vdash x \in \mathsf{W} \Rightarrow t_F x \in \mathsf{W}$$

By cut elimination we have a proof of this sequent only containing positive formulas. We can apply the main theorem 25 and get a polytime function f with properties 1 until 5. For an arbitrary $w \in \mathbb{W}$ we have for the identity address finder Id

$$0 : w \ \mathfrak{r}_{Id} \ \overline{w} \in \mathsf{W}.$$

Property 2 of f_{Id} delivers

$$\mathsf{con}\Big(f_{Id}(0:w), \mathsf{MA}\big[f_{Id}(0:w)\big]\Big) \ \ \mathfrak{R} \ \ t_F\overline{w} \in \mathsf{W},$$

which immediately implies

$$\mathcal{TM} \models \overline{\mathsf{con}_{\mathsf{W}}\Big(f_{Id}(0:w), \mathsf{MA}\big[f_{Id}(0:w)\big]\Big)} = t_F\overline{w}.$$

This implies

$$\mathsf{con}_{\mathsf{W}}\Big(f_{Id}(0:w), \mathsf{MA}\big[f_{Id}(0:w)\big]\Big) = F(w),$$

for all w in $\mathbb{W}$. So, F is a polytime function.

□

[5]The proof is easily adapted to functions with higher arity.

2.6 Applying the formalism to (the classical theory) $\mathsf{T_{PT}}$

For a more detailed proof of the upper bound of (the classical theory) $\mathsf{T_{PT}}$, we refer to the next chapter, where we reduce it to an intuitionistic system that is treated by a generalisation of the above introduced realisation approach.

Nevertheless, a more direct upper bound proof for $\mathsf{T_{PT}}$ is obtained by adopting Strahm's technique in [79] to treat classical systems: The realisation functions always delivers a pair as output, where its first element determines which formula D_i of the succedent is realised, and the second is a realiser of Γ, D_i. We use the following notations which allow to state the new main theorem very similarly as before.

- For any function $F : \mathbb{W} \to \mathbb{W}$ whose image contains exclusively pairs, let f denote the function $\lambda x.F(x)_2$.
- D_j denotes the j-th formula of a sequence Δ of formulas.

Theorem 36 *Let Γ, Δ be a sequence of positive formulas. Assume that there is a $\mathsf{T_{PT}}$ proof of $\Gamma \Rightarrow \Delta$ containing only positive formulas. Assume that a polytime address finder b is given. Then there exist polytime functions $p^{-1}, \delta, \kappa, \gamma$ (independent of b) and a polytime realisation function P_b (with projection p_b) such that for all $\vec{s}$ and for all α that are realisers of $\Gamma[\vec{s}]$ relative to b the following five properties hold:*

(1)
- *$p^{-1}(w) \leq w$ for all $w \in \mathbb{W}$.*
- *$p^{-1}(p_b(\alpha)) = \alpha$.*

(2) $p_b(\alpha)$ $\mathfrak{r}_{b^*}$ $\Gamma, D_j[\vec{s}]$ holds, where $P_b(\alpha)_1 = j$, and where b^* is the following function.

$$b^*(\rho, i) = \begin{cases} b(p^{-1}(\rho), i), & \text{if } 1 \leq i \leq |\Gamma| \\ \mathsf{MA}(\rho), & \text{if } i = |\Gamma| + 1 \end{cases}$$

(3) $\mathsf{MA}(p_b(\alpha)) \leq \mathsf{MA}(\alpha) + \kappa(\mathsf{W}_b(\alpha))$.

(4) $p_b(\alpha) \leq \alpha + \delta(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha))$.

(5) $\mathsf{W}_{b^*}(p_b(\alpha)) \leq \gamma(\mathsf{W}_b(\alpha))$.

This main theorem is again proved by induction on the depth of the positive proof of $\Gamma \Rightarrow \Delta$ in T_{PT} similarly as before. For the treatment of e.g. the $\wedge$ rule right and cut, additional case distinctions are necessary. They can be treated using additional markers.

2.7 Realising Arai's function algebra PCSF on sets

In his [2], Arai proposes a function algebra PCSF on sets in the style of Bellantoni and Cook's B [7]. In the following, we sketch an alternative upper bound proof using the realisation technique developed in section 2.4. As realisers, we use finite directed acyclic forests which nicely represent sets of hereditary finite sets. The upper bound result for PCSF depends on the exact representation of hereditary finite sets. The following example shows that there are exponentially increasing functions in PCSF relative to the usual encoding of hereditary finite sets using *undirected* graphs: Consider the following function

$$f(x;) := \{\{f(z;) : z \in x\}, \{\{f(z;) : z \in x\}\}\}.$$

Applied to the quasi-ordinals $\{\}, \{\{\}\}, \{\{\{\}\}\}, \ldots$ the undirected trees corresponding to the outputs grow exponentially in the undirected trees corresponding to the inputs.

First, we fix some notation. From now on, we tacitly assume all graphs to be finite. Directed acyclic trees are defined as usual. For a set A, a directed acyclic tree build from A (short: dat_A) is given as directed acyclic tree whose leaves are labelled by elements of A or the empty set. For each dat_A there is a unique hereditary finite set build from A corresponding to it in the obvious way. Reversely, each hereditary finite set build from A has a set of corresponding dat_As. We define directed acyclic forests build from A analogously as the earlier mentioned trees. In the following, we fix A and speak only of directed acyclic trees (dats) and - forests (dafs), and hereditary finite sets suppressing that they are build from A.

We code dafs as 0/1-matrices with labels on certain rows which are coded easily in an efficient way as words [6]. This allows us to discuss complexity questions. Additionally, we can interpret all dafs as words. It is well-known that dafs can be enumerated in polynomial time by a function $e : V \to \mathbb{N}$ such that

$$< n_0, n_1 > \in V \Rightarrow e(n_1) < e(n_0)$$

For a daf a with node n, we denote the *dat* rooted at n as a_n. It can be calculated in polynomial time whether for two nodes n, m of a daf a the *dat*s a_n and a_m correspond to the same hereditary finite set.

In the following, we show how dafs can be used to realise a function F of PCSF. The idea is to realise the list of its arguments by a daf. We show that we can efficiently extend this daf such that it realises additionally the result of the application of F to its arguments.

Definition 37 *Let $x_1, \cdots, x_m$ be hereditary finite sets. A daf a realises a sequence $< x_1, \cdots, x_m >$ relative to a feasible function b on the words (short: a $\mathfrak{r}_b$ $\vec{x}$) iff*

- *For each $1 \leq i \leq m$ $a_{b(a,i)}$ corresponds to x_i.*

b takes the same role as the address finder used in the previous sections. Now, we define a function corresponding to W_b for which we use the same denotation. For a daf a, we let $|a|$ denote the number of its nodes.

Definition 38 *For a daf a and a feasible function b, $\mathsf{W}_{b(a,k)}$ is defined as*

$$max_{\preceq}\{|a_{b(a,i)}| : 1 \leq i \leq k\}.$$

Let us formulate the main claim.

Theorem 39 *For any $F(\vec{x}; \vec{y}) \in \mathsf{PCSF}$ of n normal and m safe arguments, and any feasible function b, there are polytime functions $p^{-1}, p_b, h_b, \gamma, \delta$ such that for any a with a $\mathfrak{r}_b$ $< \vec{x}, \vec{y} >$ we have*

- – $p^{-1}(p_b(a)) = a$.

[6]As Arai, we only treat sets A that are codable by the words.

– $p^{-1}(w) \leq w$ *for all* $w \in \mathbb{W}$.

- $p_b(a)\ \mathfrak{r}_{b^*} < \vec{x}, \vec{y}, F(x_1, \cdots, x_n; a_1, \cdots, a_m) >$ *where* b^* *is defined as follows.*

$$b^*(v,i) = \begin{cases} b(p^{-1}(v), i), & \textit{if } 1 \leq i \leq n+m \\ h_b(v), & \textit{if } i = n+m+1 \\ \epsilon, & \textit{else} \end{cases}$$

- $|p_b(a)| - |a| \leq \delta(\mathsf{W}_b(a,n))$.

- *If* $m = 0$*, we have* $\mathsf{W}_{b^*}(a, n+1) \leq \gamma(\mathsf{W}_b(a,n))$

The proof is carried out very similarly as in section 2.5. We can again assume that the realisation functions always extend their arguments. We just sketch the case where a function F, having n normal arguments, is defined by the following safe recursion.

$$F(z, \vec{x}; \vec{y}) := H(z, \vec{x}; \vec{y}, \{F(z', \vec{x}; \vec{y}) : z' \in z\})$$

Given that $a\ \mathfrak{r}_b\ < z, \vec{x}, \vec{y} >$, we find a realiser of $< z, \vec{x}, \vec{a}, F(z, \vec{x}; \vec{y}) >$ as follows: Label all nodes that can be reached from $b(a,1)$ by an enumeration function e. Referring to e, we call them n_i and their corresponding set z_i. By recursion on the label i, we execute the following algorithm.

- Produce a node m for $\{F(z', \vec{x}; \vec{y}) : z' \in z_i\}$ using nodes corresponding to $F(z_j, \vec{x}; \vec{y})$ for some $j < i$ which can be found by induction hypothesis.

- Produce a node corresponding to

$$H(z_i, \vec{x}; \vec{y}, \{F(z', \vec{x}; \vec{y}) : z' \in z_i\})$$

  using n_i and m.

- Remember that this node corresponds to $F(z_i, \vec{x}; \vec{y})$.

For each i the above described steps can be executed in polynomial time assuming the induction hypothesis. Therefore, the whole procedure is executed in polynomial time if we can find a bounding polynomial for the involved dafs. But this is found easily since the number of added nodes in each step is bounded polynomially in $\mathsf{W}_b(a, n)$.

The theorem implies that functions of PCSF are efficiently computable as long as hereditary finite sets are given by dats, it does not directly claim how these functions behave on sets given by other codings. The example given at the beginning of the section shows that we cannot expect a feasible behaviour of functions of PCSF on undirected trees. On pages 6 and 7 of [2], Arai describes how to represent words by hereditary finite sets. Our result readily implies that functions of PCSF that can be restricted to sets representing words in this way are feasible even relative to their coding by *undirected* trees.

Chapter 3

Extensions of T_{PT} by choice and negative reflection

3.1 Introduction

In this chapter, we present extensions of the theory T_{PT} from chapter 2 by the positive axiom of choice and prove the conservativity of this extension. In addition, we extend the intuitionistic theory $\mathsf{T}^{i}_{\mathsf{PT}}$ by reflection principles for negative formulas. The motivation for this extension is to study the difficult question what role induction over negative formulas plays in the setting of weak applicative theories. In the literature, negative induction in the weak applicative setting is only studied by Cantini in his [14] for theories of primitive recursive strength, whose upper bounds are produced by embeddings into PR. The proof of the conservativity of the extension of $\mathsf{T}^{i}_{\mathsf{PT}}$ by reflection principles for particular negative formulas shows that in the *intuitive* setting of polynomial strength one can carefully add induction over negative formulas. The introduced technique of treating negative induction formulas by realisers coding them as finite functions can easily be adapted to further weak applicative theories, e.g. the ones presented in [79].

The (positive) axiom of choice is given as follows for A a positive formula.

$$(\mathsf{Pos-AC}) \qquad (\forall x)(x \in \mathsf{W} \rightarrow (\exists y)(y \in \mathsf{W} \wedge A[x,y])) \rightarrow \\ (\exists f)(\forall x)(x \in \mathsf{W} \rightarrow fx \in \mathsf{W} \wedge A[x,fx])$$

Particularly in a constructive setting, this axiom is very natural since in essence it claims that for provable Π_2 statements (with quantifiers restricted to the words) we can explicitly describe the dependence of the witness of the existential quantifier on x by a function f. The axiom of choice in an applicative setting was first considered by Feferman in [33]. In his [18], Cantini analysed various weak applicative theories of truth containing the axiom of choice. By connecting techniques developed in this paper with the address and pointer formalism developed in the last chapter, we prove that the extension of T_{PT} by the axiom of choice still proves totality exactly for the functions computable in polynomial time. Unfortunately, we succeeded to prove the upper bound for the extension of T_{PT} by truth reflection for negative formulas only in an intuitionistic setting. This is because the realisation approach for applicative theories, developed by Strahm in [79], can hardly be applied to negative formulas in a classical setting [1] . For an applicative theory with a truth predicate reflecting negative formulas of primitive recursive strength, see Cantini's [19].

Let us give a brief outline of the content of this chapter. In the next section, we present natural extensions of T_{PT}. Our strategy to prove the feasibility of the extensions is to reduce them to an intuitionistic theory T^+, as performed by Cantini in [18]. We choose T^+ to be strong enough to allow an embedding of an extension of T_{PT} by the axiom of choice, and an extension of $\mathsf{T}^i_{\mathsf{PT}}$ by the axiom of choice and reflection of particular negative formulas. We sketch these reductions, and give a suitable sequent style formalisation of T^+. In section 3, we develop a realisation approach that fuses the address-pointer formalism presented in the last chapter and Cantini's realisation approach with functionals [18]. In section 4, we use this approach to realise T^+, which implies the feasible upper bound of the theory. In section 5, we sketch how the extensions of other weak applicative theories by induction over negative formulas can be treated.

[1]An exception is Strahm's [80] where the realisation of a sequent calculus including negative formulas works, because they occur only at the left side of $\Rightarrow$.

3.2 Two extensions of T_{PT}

The extensions of T_{PT}, $\mathsf{T}^{i}_{\mathsf{PT}}$, respectively, are formulated in a language $\mathrm{L}^{+}_{\mathsf{T}}$ that extends L_{T} by a constant $\dot{\rightarrow}$ to code implications and a unary predicate Cl which singles out terms that represent formulas possibly occurring in negative positions of reflected formulas [2]. We assume that A is a positive L_{T} formula.

Axiom of choice

$$(\mathsf{Pos}-\mathsf{AC}) \qquad (\forall x)(x \in \mathsf{W} \rightarrow (\exists y)(y \in \mathsf{W} \wedge A[x,y])) \rightarrow (\exists f)(\forall x)(x \in \mathsf{W} \rightarrow fx \in \mathsf{W} \wedge A[x, fx]),$$

Uniformity principle

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

The uniformity principle claims that there are no interesting functions from the universe into the numbers. It was first proposed by Cantini in [18]. The following axioms allow the reflection of non-positive formulas.

Reflection of negative formulas (RN)

$(\mathsf{Cl}_{\dot{=}})$	$\mathsf{Cl}(x \dot{=} y)$
$(\mathsf{Cl}_{\dot{\wedge}})$	$\mathsf{Cl}(x) \wedge \mathsf{Cl}(y) \rightarrow \mathsf{Cl}(x \dot{\wedge} y)$
$(\mathsf{Cl}_{\dot{\vee}})$	$\mathsf{Cl}(x) \wedge \mathsf{Cl}(y) \rightarrow \mathsf{Cl}(x \dot{\vee} y)$
$(\mathsf{Cl}_{\dot{\exists}})$	$(\forall x)\mathsf{Cl}(tx) \rightarrow \mathsf{Cl}(\dot{\exists} t)$
$(\mathsf{Cl}_{\dot{\forall}})$	$(\forall x)\mathsf{Cl}(tx) \rightarrow \mathsf{Cl}(\dot{\forall} t)$
$(\dot{\rightarrow})$	$\mathsf{Cl}(x) \rightarrow (\mathsf{T}(x \dot{\rightarrow} y) \leftrightarrow \mathsf{T}(x) \rightarrow \mathsf{T}(y))$

Note that $\mathsf{Cl}(x)$ implies $\neg\mathsf{T}(x) \leftrightarrow \mathsf{T}(\dot{\neg} x)$ for negation defined as usual.

Definition 40 *Let the set $\mathfrak{S}$ of extended positive formulas be the smallest set S of $\mathrm{L}^{+}_{\mathsf{T}}$ formulas satisfying the following conditions.*

[2] The predicate Cl reminds of standard proposition predicates of theories of truth but is more restrictive since codes containing $\dot{\rightarrow}$ cannot be classes.

- *Each positive formula without occurrences of* Cl *is in* $\mathfrak{S}$.
- S *is closed under the connectors* $\wedge, \vee$ *and the quantifiers.*
- *Let* A *be positive without occurrences of* W, T *and* Cl*, and assume* $B \in S$*. Then* $A \rightarrow B \in S$*.*

The following lemma can be proved by a simple induction on the complexity of A.

Lemma 41 (Tarski biconditionals) *Assume* $A \in \mathfrak{S}$*. Then*

$$\mathsf{T}^i_{\mathsf{PT}} + \mathsf{RN} \vdash u \in \mathsf{W} \rightarrow (A^u \leftrightarrow \mathsf{T}(\ulcorner A^u \urcorner)),$$

where $\ulcorner A^u \urcorner$ *denotes the Gödel code of* A^u*, defined as in chapter 2, and extended in the obvious way to formulas containing implications.*

In this chapter, we deliver upper bounds for the theories

- $\mathsf{T}_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP}$
- $\mathsf{T}^i_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP} + \mathsf{RN}$

Let us introduce a theory T^+ to which both of these can be reduced.

Definition 42 *Let* T^+ *be a theory formulated in the extension* $\mathrm{L}^{+'}_{\mathsf{T}}$ *of* $\mathrm{L}^{+}_{\mathsf{T}}$ *by a new predicate* $\mathcal{T}$*, and new constants* $\dot{\mathsf{Cl}}$*,* $\dot{\mathsf{T}}$ *to code the class - and the truth predicate. It contains the following axioms in addition to the ones of* $\mathsf{T}^i_{\mathsf{PT}}$*.*

Extended induction

$(\mathsf{ext} - \mathsf{Ind})$
$(\mathsf{T}(a\epsilon) \vee C) \wedge (\forall x \in \mathsf{W})[(\mathsf{T}(ax) \vee C) \rightarrow (\mathsf{T}(a(\mathsf{s}_0 x)) \vee C) \wedge (\mathsf{T}(a(\mathsf{s}_1 x))) \vee C)]$
$\rightarrow \forall x \in \mathsf{W}(\mathsf{T}(ax) \vee C)$,

where C is a positive $\mathrm{L}^{+'}_{\mathsf{T}}$ formula.

$\mathcal{T}$-reflection of positive formulas $(\mathcal{T} - \mathsf{Ref})$

$$\begin{array}{lll} (\dot{=}) & \mathcal{T}(x \dot{=} y) & \leftrightarrow \quad x = y \\ (\dot{\mathsf{W}}) & \mathcal{T}(\dot{\mathsf{W}}x) & \leftrightarrow \quad \mathsf{W}(x) \end{array}$$

$$\begin{array}{lllll}
(\dot{\mathsf{Cl}}) & \mathcal{T}(\dot{\mathsf{Cl}}x) & \leftrightarrow & \mathsf{Cl}(x) \\
(\dot{\mathsf{T}}) & \mathcal{T}(\dot{\mathsf{T}}x) & \leftrightarrow & \mathsf{T}(x) \\
(\dot{\wedge}) & \mathcal{T}(x \mathbin{\dot{\wedge}} y) & \leftrightarrow & \mathcal{T}(x) \wedge \mathcal{T}(y) \\
(\dot{\vee}) & \mathcal{T}(x \mathbin{\dot{\vee}} y) & \leftrightarrow & \mathcal{T}(x) \vee \mathcal{T}(y) \\
(\dot{\forall}) & \mathcal{T}(\dot{\forall} f) & \leftrightarrow & (\forall z)\mathcal{T}(fz) \\
(\dot{\exists}) & \mathcal{T}(\dot{\exists} f) & \leftrightarrow & (\exists z)\mathcal{T}(fz)
\end{array}$$

Axiom of choice for $\mathcal{T}$

$$(\mathcal{T} - \mathsf{AC}) \qquad (\forall x)(x \in \mathsf{W} \rightarrow (\exists y)(y \in \mathsf{W} \wedge \mathcal{T}(axy))) \rightarrow \\ (\exists f)(\forall x)(x \in \mathsf{W} \rightarrow fx \in \mathsf{W} \wedge \mathcal{T}(axy)),$$

Uniformity principle for $\mathcal{T}$

$$(\mathcal{T} - \mathsf{UP}) \qquad (\forall x)(\exists y \in \mathsf{W})\mathcal{T}(axy) \rightarrow (\exists y \in \mathsf{W})(\forall x)\mathcal{T}(axy),$$

Constant domains

$$(\mathcal{T} - \mathsf{CD}) \qquad (\forall x)\mathcal{T}(a \dot{\vee} bx) \rightarrow \mathcal{T}(a \dot{\vee} \dot{\forall} b)$$

The truth predicate $\mathcal{T}$ can reflect arbitrary positive $\mathrm{L}_{\mathsf{T}}^{+}$ formulas but does not deliver any additional strength because induction is only allowed for the weaker predicate T.

Lemma 43 *The provably total functions of the following theories are contained in the provably total functions of* T^{+}.

- $\mathsf{T}_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP}$
- $\mathsf{T}_{\mathsf{PT}}^{i} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP} + \mathsf{RN}$

Proof. For the intuitionistic theory the claim follows immediately because $\mathcal{T}$ reflects any positive $\mathrm{L}_{\mathsf{T}}^{+}$ formula. For the classical system, we can prove the claim in a very similar way as Cantini in [18] for his system $\mathcal{S}$. We define the double-negation translation N as usual and extend T^{+} by the following axiom

$$(\mathcal{T}\mathsf{S}) \qquad \neg\neg\mathcal{T}(x) \rightarrow \mathcal{T}(x),$$

We easily derive the following auxiliary lemma, where we use that $\mathsf{Pos} - \mathsf{AC}$, UP follow from $(\mathcal{T} - \mathsf{AC}), (\mathcal{T} - \mathsf{UP})$.

Lemma 44 *Let A be an $\mathrm{L}_{\mathsf{T}}^{+}$ formula.*

$$\mathsf{T}_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP} \vdash A \textit{ implies } \mathsf{T}^{+} + \mathcal{T}\mathsf{S} \vdash A^{N}$$

In a second step, we eliminate $\mathcal{T}\mathsf{S}$ by a forcing approach exactly as in Cantini's [18]: We let $x \leq_{\mathcal{T}} y$ abbreviate $\mathcal{T}(y) \rightarrow \mathcal{T}(x)$, and inductively assign to every $\mathrm{L}_{\mathsf{T}}^{+\prime}$ formula A an $\mathrm{L}_{\mathsf{T}}^{+\prime}$ formula $f \Vdash A$ where f does not occur in A:

$$\begin{gathered} f \Vdash A \Leftrightarrow \mathcal{T}(f) \vee A, \text{ if } A \text{ is atomic} \\ f \Vdash A \rightarrow B \Leftrightarrow (\forall g \leq_{\mathcal{T}} f)(g \Vdash A \rightarrow g \Vdash B) \\ f \Vdash A \circ B \Leftrightarrow f \Vdash A \circ f \Vdash B (\circ = \vee, \wedge) \\ f \Vdash (Qx)A \Leftrightarrow (Qx)(f \Vdash A)(Q = \forall, \exists) \end{gathered}$$

Then the same arguments as in [18] yield the following lemma.

Lemma 45 *For any $\mathrm{L}_{\mathsf{T}}^{+\prime}$ formula A, we have*

$$\mathsf{T}^{+} + \mathcal{T}\mathsf{S} \vdash A \Rightarrow \mathsf{T}^{+} \vdash (\forall f)(f \Vdash A)$$

This easily implies the result. Note that induction can only be forced if the additional disjunct C is allowed, as for Cantini's system PTTC. $\square$

Unfortunately, Cantini's technique used above does not work for truth predicates that reflect negative equations. Therefore it does not allow to reduce $\mathsf{T}_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP} + \mathsf{RN}$ to T^{+}.

For the computation of the upper bound of T^{+}, it will be practical to work in a sequent style formalisation $\mathsf{T}^{\#}$ of T^{+}. For the success of our realisation approach it is crucial to formulate some of the rules in a specific way. The idea is to avoid rules where the antecedent of the premisses is logically weaker than the antecedent of the conclusion. Let us sketch the axioms and rules of $\mathsf{T}^{\#}$, where Γ contains arbitrary $\mathrm{L}_{\mathsf{T}}^{+}$ formulas.

Structural rules: We have the usual structural rules but allow weakening only for negative formulas.

Axioms of T_{PT}: These are formulated in sequent style as for T_{PT}, containing side formulas.

Axioms for $\mathcal{T}$: These are formulated as expected for sequent calculus, containing side formulas. E.g., the axiom of choice is given as follows.

$$\Gamma, (\forall x)(x \in \mathsf{W} \to (\exists y)(y \in \mathsf{W} \wedge \mathcal{T}(axy)) \Rightarrow$$
$$(\exists f)(\forall x)(x \in \mathsf{W} \to fx \in \mathsf{W} \wedge \mathcal{T}(ax(fx)))$$

Reflection of negative formulas: These are formulated as expected for sequent calculus, containing side formulas. E.g., the $\dot{\to}$-axioms are given as follows.

$$\begin{aligned}\Gamma, \mathsf{Cl}(s), \mathsf{T}(s) \to \mathsf{T}(t) &\Rightarrow \mathsf{T}(s \dot{\to} t)\\ \Gamma, \mathsf{Cl}(s), \mathsf{T}(s \dot{\to} t), \mathsf{T}(s) &\Rightarrow \mathsf{T}(t)\end{aligned}$$

Induction: Let C be a positive $\mathrm{L}_{\mathsf{T}}^{+\prime}$ formula.

$$\frac{\Gamma, t \in \mathsf{W} \Rightarrow \mathsf{T}(r\epsilon) \vee C \qquad \Gamma, t \in \mathsf{W}, x \in \mathsf{W}, \mathsf{T}(rx) \vee C \Rightarrow \mathsf{T}(r(\mathsf{s}_i x)) \vee C}{\Gamma, t \in \mathsf{W} \Rightarrow \mathsf{T}(rt) \vee C}$$

Universal quantification left:

$$\frac{\Gamma, (\forall x)A[x], A[t] \Rightarrow D}{\Gamma, (\forall x)A[x] \Rightarrow D}$$

The other rules are given as usual. $\mathsf{T}^{\#}$ can be proved to be equivalent to T^{+} by standard techniques. A standard cut elimination argument allows to restrict cut formulas to the set $\mathfrak{A}$, defined as follows.

Definition 46 *Let $\mathfrak{A}$ be the smallest set S satisfying the following specifications.*

- *S contains all positive formulas of $\mathrm{L}_{\mathsf{T}}^{+}$.*
- *S contains all formulas of the form $A \to B$ where A, B are positive.*
- *If S contains the formula A, then also $(\forall x)A$.*
- *If S contains the formula A, then also $(\exists x)A$.*

Because we have weakening with negative formulas, we can always restrict ourselves to proofs whose leafs contain only positive side formulas.

3.3 A realisation formalism for proofs in T^+

We extend the realisation relation $\mathfrak{R}$ to positive formulas of the extended language $\mathrm{L}_{\mathsf{T}}^{+\prime}$ via a relation $\equiv$. This realisation relation is later extended to negative $\mathfrak{A}$-formulas and to sequences of $\mathfrak{A}$-formulas.

From now on, we silently assume all formulas to be in $\mathfrak{A}$. We call a formula negative if it is not positive and in $\mathfrak{A}$.

3.3.1 The relation $\equiv$

We define a ternary relation $\equiv$ with two words v, w and a formula A as relata, written as $v \equiv_A w$. We interpret $v \equiv_A w$ as "v, w are equivalent realisers of the formula A". The binary relation that holds between a word w and a formula A exactly if $w \equiv_A w$ will be seen to be an extension of the realisation relation $\mathfrak{R}$ defined in chapter 2.

Clearly, the class axioms only force us to allow classes s such that the set

$$\{v \in \mathbb{W} \mid \text{there is a substitution } [\vec{t}] \text{ such that } v \;\mathfrak{R}\; \mathsf{T}(s)[\vec{t}]\}$$

is finite. This motivates the definition of the relation $\equiv$ such that the relation $\mathcal{S}$ defined as $w \;\mathcal{S}\; A :\Leftrightarrow w \equiv_A w$ fulfils the following specifications.

- An $\mathcal{S}$ realiser of $\mathsf{Cl}(s)$ collects all possible $\mathfrak{R}$-realisers of $\mathsf{T}(s)$. That means all $\mathfrak{R}$-realisers which according to the build up of the presupposed normal form t of s could be realisers of $\mathsf{T}(s)[\vec{t}]$ for some substitution $[\vec{t}]$.

- An $\mathcal{S}$ realiser of $\mathsf{T}(s \dot{\rightarrow} t)$ describes a finite function f as a collection of pairs whose domain are the possible realisers of $\mathsf{T}(s)$. If for a possible realiser $v \in \mathbb{W}$ there exists a substitution $[\vec{t}]$ such that $v \;\mathfrak{R}\; \mathsf{T}(s)[\vec{t}]$, then $f(v) \;\mathfrak{R}\; \mathsf{T}(t)[\vec{t}]$. We call the possible realiser v an actual realiser in this case.

Note that the specification for the realisation relation we have given above allows the mentioned function f to return arbitrary outputs if its input is not an actual realiser. The relation $\equiv$ will allow us to identify realisers that only differ in this insignificant way. We define $\equiv$ using an auxiliary relation $P(v, w, q, d)$ on words. Its first two arguments are interpreted as realisers, the third as code of a formula, and the fourth is 0 or 1. $P(v, w, q, 0)$ can be interpreted as follows: v and w are equivalent realisers of the formula coded by q. $P(v, w, q, 1)$ can be interpreted as: q codes a formula of the form $\mathsf{T}(s)$ where s is a class and it does not hold that v and w are equivalent realisers of $\mathsf{T}(s)$. The segment $P(\cdots, 1)$ allows us to deal with the negative occurrences of the realisation relation in the specifications for Cl and $\dot{\rightarrow}$, and to define the predicate P as least fixed point of a positive operator $\mathfrak{B}$ on words [3]. All the arguments and constructions of this section can be justified in PA^w_Ω, a system developed by Jäger in [53] of strength $\hat{\mathsf{ID}}_1$.

In the following, we often write $s = t$ meaning that these terms are equal in the standard open term model $\mathcal{TM}$, analogously for $\leq$. We use quantifiers over substitutions as abbreviations of accordingly restricted word quantifiers. Assume that $\ulcorner\mathsf{W}\urcorner, \ulcorner\mathsf{T}\urcorner, \ulcorner\wedge\urcorner, \ulcorner\vee\urcorner, \ulcorner\mathsf{Cl}\urcorner, \ulcorner\rightarrow\urcorner$ are pairwise different words and different from ϵ. We again use Clote's pairing function [20]. We also assume the unity of generators what means that terms built in a different way from the truth constants cannot be equivalent in $\mathcal{TM}$. The positive operator $\mathfrak{B}(X, v, w, q, d)$ holds by definition exactly if one of the following conditions is fulfilled.

- $q = \ulcorner\mathsf{T}(s)\urcorner \wedge s = t_0 \dot{=} t_1 \wedge t_0 = t_1 \wedge w = v = \langle\ulcorner\mathsf{T}\urcorner, \epsilon\rangle \wedge d = 0$
- $q = \ulcorner\mathsf{T}(s)\urcorner \wedge s = t_0 \dot{=} t_1 \wedge \neg(t_0 = t_1 \wedge w = v = \langle\ulcorner\mathsf{T}\urcorner, \epsilon\rangle) \wedge d = 1$
- $q = \ulcorner\mathsf{T}(s)\urcorner \wedge s = t_0 \mathbin{\dot{\wedge}} t_1 \wedge w = \langle\ulcorner\wedge\urcorner, w_2, w_3\rangle \wedge v = \langle\ulcorner\wedge\urcorner, v_2, v_3\rangle \wedge X(w_2, v_2, t_0, 0) \wedge X(w_3, v_3, t_1, 0) \wedge d = 0$
- $q = \ulcorner\mathsf{T}(s)\urcorner \wedge s = t_0 \mathbin{\dot{\wedge}} t_1 \wedge (w \neq \langle\ulcorner\wedge\urcorner, w_2, w_3\rangle \vee v \neq \langle\ulcorner\wedge\urcorner, v_2, v_3\rangle \vee X(w_2, v_2, t_0, 1) \vee X(w_3, v_3, t_1, 1)) \wedge d = 1$
- $q = \ulcorner\mathsf{T}(s)\urcorner \wedge s = t_0 \mathbin{\dot{\vee}} t_1 \wedge (w = \langle\ulcorner\vee\urcorner, 0, w_3\rangle \wedge v = \langle\ulcorner\vee\urcorner, 0, v_3\rangle \wedge X(w_3, v_3, t_0, 0)) \vee (w = \langle\ulcorner\vee\urcorner, 1, w_3\rangle \wedge v = \langle\ulcorner\vee\urcorner, 1, v_3\rangle \wedge X(w_3, v_3, t_1, 0)) \wedge d = 0$

[3] See Moschovaki's [67] for elementary results about inductive definitions.

- $q = \ulcorner \mathsf{T}(s) \urcorner \wedge s = t_0 \dot{\vee} t_1 \wedge (w \neq \langle \ulcorner \vee \urcorner, 0, w_3 \rangle \wedge w \neq \langle \ulcorner \vee \urcorner, 1, w_3 \rangle) \vee (v \neq \langle \ulcorner \vee \urcorner, 0, v_3 \rangle \wedge v \neq \langle \ulcorner \vee \urcorner, 1, v_3 \rangle) \vee w_2 \neq v_2 \vee X(w_3, v_3, t_{w_2}, 1) \wedge d = 1$
- $q = \ulcorner \mathsf{T}(s) \urcorner \wedge s = \dot{\forall} t \wedge X(w, v, tx, 0) \wedge d = 0$ for a fresh variable x
- $q = \ulcorner \mathsf{T}(s) \urcorner \wedge s = \dot{\forall} t \wedge X(w, v, tx, 1) \wedge d = 1$ for a fresh variable x
- $q = \ulcorner \mathsf{T}(s) \urcorner \wedge s = \dot{\exists} t \wedge (\exists s) X(w, v, ts, 0) \wedge d = 0$
- $q = \ulcorner \mathsf{T}(s) \urcorner \wedge s = \dot{\exists} t \wedge (\forall s) X(w, v, ts, 1) \wedge d = 1$
- $q = \ulcorner \mathsf{T}(s) \urcorner \wedge s = \dot{\mathsf{W}} t_0 t_1 \wedge t_1 \leq_{\mathsf{W}} t_0 \wedge t_0 \in \mathsf{W} \wedge w = v = \langle \ulcorner \mathsf{T} \urcorner, value(t_1) \rangle \wedge d = 0$
- $q = \ulcorner \mathsf{Cl}(s) \urcorner \wedge s = t_0 \doteq t_1 \wedge w = \langle \ulcorner \mathsf{Cl} \urcorner, \langle \mathsf{T}, \epsilon \rangle \rangle \wedge w = v \wedge d = 0$
- $$\begin{gathered} q = \ulcorner \mathsf{Cl}(s) \urcorner \wedge s = t_0 \dot{\wedge} t_1 \wedge (\exists a \in \mathbb{W})(\exists b \in \mathbb{W}) X(a, a, \ulcorner \mathsf{Cl}(t_0) \urcorner, 0) \wedge \\ X(b, b, \ulcorner \mathsf{Cl}(t_1) \urcorner, 0) \wedge a = \langle \ulcorner \mathsf{Cl} \urcorner, a_2, \cdots, a_n \rangle \wedge b = \langle \ulcorner \mathsf{Cl} \urcorner, b_2, \cdots, b_m \rangle \wedge \\ w = v = \langle \ulcorner \mathsf{Cl} \urcorner, \langle \ulcorner \wedge \urcorner, a_2, b_2 \rangle, \cdots, \langle \ulcorner \wedge \urcorner, a_n, b_2 \rangle, \cdots, \langle \ulcorner \wedge \urcorner, a_n, b_m \rangle \rangle \wedge \\ d = 0 \end{gathered}$$
- $$\begin{gathered} q = \ulcorner \mathsf{Cl}(s) \urcorner \wedge s = t_0 \dot{\vee} t_1 \wedge (\exists a \in \mathbb{W})(\exists b \in \mathbb{W}) X(a, a, \ulcorner \mathsf{Cl}(t_0) \urcorner, 0) \wedge \\ X(b, b, \ulcorner \mathsf{Cl}(t_1) \urcorner, 0) \wedge a = \langle \ulcorner \mathsf{Cl} \urcorner, a_2, \cdots, a_n \rangle \wedge b = \langle \ulcorner \mathsf{Cl} \urcorner, b_2, \cdots, b_m \rangle \wedge \\ w = \langle \ulcorner \mathsf{Cl} \urcorner, \langle \ulcorner \vee \urcorner, 0, a_2 \rangle, \cdots, \langle \ulcorner \vee \urcorner, 0, a_n \rangle, \langle \ulcorner \vee \urcorner, 1, b_2 \rangle \cdots, \langle \ulcorner \vee \urcorner, 1, b_m \rangle \rangle \\ \wedge w = v \wedge d = 0 \end{gathered}$$
- $q = \ulcorner \mathsf{Cl}(s) \urcorner \wedge s = \dot{\forall} t \wedge X(w, w, \ulcorner \mathsf{Cl}(tx) \urcorner, 0) \wedge w = v \wedge d = 0$ for a fresh variable x
- $q = \ulcorner \mathsf{Cl}(s) \urcorner \wedge s = \dot{\exists} t \wedge X(w, w, \ulcorner \mathsf{Cl}(tx) \urcorner, 0) \wedge w = v \wedge d = 0$ for a fresh variable x
- $q = \ulcorner \mathsf{T}(s) \urcorner \wedge s = t_0 \dot{\rightarrow} t_1 \wedge w = \langle \ulcorner \rightarrow \urcorner, \langle w_{2,1}, w_{2,2} \rangle, \cdots, \langle w_{n,1}, w_{n,2} \rangle \rangle \wedge$
  $v = \langle \ulcorner \rightarrow \urcorner, \langle w_{2,1}, v_{2,2} \rangle, \cdots, \langle w_{n,1}, v_{n,2} \rangle \rangle \wedge$
  $X(\langle \ulcorner \mathsf{Cl} \urcorner, w_{2,1}, \cdots, w_{n,1} \rangle, \langle \ulcorner \mathsf{Cl} \urcorner, w_{2,1}, \cdots, w_{n,1} \rangle, \ulcorner \mathsf{Cl}(t_0) \urcorner, 0) \wedge$
  $(\forall 2 \leq i \leq n) \forall [\vec{s}] (X(w_{i,1}, w_{i,1}, t_0[\vec{s}], 1) \vee X(w_{i,2}, v_{i,2}, t_1[\vec{s}], 0))$

Let P denote the least fixed point of the positive operator $\mathfrak{B}$ and let P^σ denote its σ-th stage. The following lemmas are provable by a simple induction on the ordinal stages [4].

Lemma 47 *For all words w, v, q we have*

$$\neg(P(w, v, q, 0) \wedge P(w, v, q, 1))$$

Lemma 48 *For all stages σ and all words w and all terms t we have*

$$P^\sigma(w, w, \ulcorner \mathsf{Cl}(t) \urcorner, 0) \Rightarrow (\forall v \in \mathsf{W})(\forall[\vec{s}])(P^\sigma(v, v, t[\vec{s}], 0) \vee P^\sigma(v, v, t[\vec{s}], 1)$$

We define a similar relation for $\mathcal{T}$ atoms. R is defined as least fixed point of a positive operator $\mathfrak{C}$ on words. The positive operator $\mathfrak{C}(X, v, w, q)$ holds by definition exactly if one of the following conditions is fulfilled.

- $q = \ulcorner \mathcal{T}(s) \urcorner \wedge s = t_0 \dot{=} t_1 \wedge t_0 = t_1 \wedge w = v = \langle \ulcorner \mathsf{W} \urcorner, \epsilon \rangle$
- $q = \ulcorner \mathcal{T}(s) \urcorner \wedge s = \dot{\mathsf{W}} t \wedge t \in \mathsf{W} \wedge w = v = \langle \ulcorner \mathsf{W} \urcorner, value(t) \rangle$
- $q = \ulcorner \mathcal{T}(s) \urcorner \wedge s = \dot{\mathsf{T}} t \wedge P(v, w, \ulcorner \mathsf{T}(t) \urcorner, 0)$
- $q = \ulcorner \mathcal{T}(s) \urcorner \wedge s = \dot{\mathsf{Cl}} t \wedge t_0 = t_1 \wedge P(v, w, \ulcorner \mathsf{Cl}(t) \urcorner, 0)$

[4]Induction up to stage ω is already sufficient to prove the next two lemmas since all classes are build below this level. Nevertheless, the fixed point construction closes only after ω, e.g.

$$\mathsf{T}(\mathsf{d}_\mathsf{W}(\epsilon, \epsilon, x, x) = \epsilon \dot{\rightarrow} \overbrace{\dot{\forall}\dot{\forall} \cdots \dot{\forall}}^{|x| \text{ times}} 0 \dot{=} 0)$$

becomes realised exactly at omega. Note, that for this result it is crucial that the set $\{x | \mathsf{d}_\mathsf{W}(\epsilon, \epsilon, x, x) = \epsilon\}$ describes W in $\mathcal{TM}$, and that the antecedent is always realised by the same word, which is ϵ in this case.

$$\mathsf{T}(\mathsf{d}_\mathsf{W}(\epsilon, \epsilon, x, x) = \epsilon \dot{\rightarrow} \overbrace{0 \dot{=} 0 \dot{\wedge} 0 \dot{=} 0 \dot{\wedge} \cdots \dot{\wedge} 0 \dot{=} 0}^{|x| \text{ times}})$$

never becomes realised because the clause for $t_0 \dot{\rightarrow} t_1$ assumes the existence of a function from realisers of $\mathsf{T}(t_0)[\vec{s}]$ to realisers of $\mathsf{T}(t_1)[\vec{s}]$. This is unproblematic since the theory cannot prove that $\{x | \mathsf{d}_\mathsf{W}(\epsilon, \epsilon, x, x) = \epsilon\}$ describes W.

- $q = \ulcorner \mathcal{T}(s) \urcorner \land s = t_0 \mathbin{\dot\land} t_1 \land w = \langle \ulcorner \land \urcorner, w_2, w_3 \rangle \land v = \langle \ulcorner \land \urcorner, v_2, v_3 \rangle \land X(w_2, v_2, t_0, 0) \land X(w_3, v_3, t_1, 0) \land d = 0$

- $q = \ulcorner \mathcal{T}(s) \urcorner \land s = t_0 \mathbin{\dot\lor} t_1 \land (w = \langle \ulcorner \lor \urcorner, 0, w_3 \rangle \land v = \langle \ulcorner \lor \urcorner, 0, v_3 \rangle \land X(w_3, v_3, t_0, 0)) \lor (w = \langle \ulcorner \lor \urcorner, 1, w_3 \rangle \land v = \langle \ulcorner \lor \urcorner, 1, v_3 \rangle \land X(w_3, v_3, t_1, 0)) \land d = 0$

- $q = \ulcorner \mathcal{T}(s) \urcorner \land s = \dot\forall t \land X(w, v, tx, 0) \land d = 0$ for a fresh variable x

- $q = \ulcorner \mathcal{T}(s) \urcorner \land s = \dot\exists t \land (\exists s) X(w, v, ts, 0) \land d = 0$

We define the previously mentioned relation $\equiv$ on triples of two words and one formula by induction on the complexity of the formula argument as follows.

$$
\begin{array}{lll}
w \equiv_{\mathsf{T}(t)} v & \text{iff} & P(w, v, \ulcorner \mathsf{T}(t) \urcorner, 0) \\
w \equiv_{\mathsf{Cl}(t)} v & \text{iff} & P(w, v, \ulcorner \mathsf{Cl}(t) \urcorner, 0) \\
w \equiv_{\mathcal{T}(t)} v & \text{iff} & R(w, v, \ulcorner \mathcal{T}(t) \urcorner) \\
w \equiv_{\mathsf{W}(t)} v & \text{iff} & w = v = \langle \ulcorner \mathsf{W} \urcorner, w_2 \rangle \text{ and } t = \overline{w_2}, \\
w \equiv_{t_1 = t_2} v & \text{iff} & w = v = \langle \ulcorner \mathsf{W} \urcorner, \epsilon \rangle \text{ and } t_1 = t_2 \\
w \equiv_{A \land B} v & \text{iff} & w = \langle \ulcorner \land \urcorner, w_2, w_3 \rangle \text{ and } v = \langle \ulcorner \land \urcorner, v_2, v_3 \rangle \text{ and} \\
 & & w_2 \equiv_A v_2 \text{ and } w_3 \equiv_B v_3 \\
w \equiv_{A \lor B} v & \text{iff} & (w = \langle \ulcorner \lor \urcorner, 0, w_3 \rangle \text{ and } v = \langle \ulcorner \lor \urcorner, 0, v_3 \rangle \text{ and } w_3 \equiv_A v_3) \text{ or} \\
 & & (w = \langle \ulcorner \lor \urcorner, 1, w_3 \rangle \text{ and } v = \langle \ulcorner \lor \urcorner, 1, v_3 \rangle \text{ and } w_3 \equiv_B v_3) \\
w \equiv_{(\forall x) A(x)} v & \text{iff} & w \equiv_{A(u)} v \text{ for a fresh variable } u, \\
w \equiv_{(\exists x) A(x)} v & \text{iff} & w \equiv_{A(t)} \ \text{ for some term } t.
\end{array}
$$

Lemma 49 *Let v, w be words and A a positive $\mathrm{L}_{\mathsf{T}}^{+}$ formula. We let $\vec{s} = \vec{t}$ abbreviate $s_1 = t_1 \land \cdots \land s_n = t_n$. Then the following holds for all substitutions $[\vec{s}], [\vec{t}]$.*

- $w \equiv_{A[\vec{x}]} v \Rightarrow v \equiv_{A[\vec{s}]} w$

- $\vec{s} = \vec{t} \Rightarrow (w \equiv_{A[\vec{s}]} v \Leftrightarrow w \equiv_{A[\vec{t}]} v)$

Proof. Let us prove the first claim for the atoms $\mathsf{Cl}(t)$ and $\mathsf{T}(t)$ first. We use an induction on the stages of the fixed point construction of $\equiv$ over the following property: For all lambda terms $s[\vec{x}]$ and any substitution $[\vec{t}]$, we have

- $P^\alpha(w, v, \ulcorner \mathsf{T}(s[\vec{x}])\urcorner, 0) \Rightarrow P^\alpha(w, v, \ulcorner \mathsf{T}(s[\vec{t}])\urcorner, 0)$, and
- $P^\alpha(w, v, \ulcorner \mathsf{Cl}(s[\vec{x}])\urcorner, 0) \Rightarrow P^\alpha(w, v, \ulcorner \mathsf{Cl}(s[\vec{t}])\urcorner, 0)$.

For all fixed point clauses except of the case where $s[\vec{x}] = t_0 \dot{\rightarrow} t_1[\vec{x}]$, the claim follows directly from the induction hypothesis, using that equality of lambda terms is closed under substitution. If q codes the formula $\mathsf{T}(s[\vec{x}])$ and $s[\vec{x}] = t_0 \dot{\rightarrow} t_1[\vec{x}]$ the claim follows because of the universal quantification over the substitutions $[\vec{s}]$ in the corresponding clause.

For the $\mathcal{T}$ atoms, the claim follows easily by induction on the build up of R, assuming it for T - and Cl atoms. For composed formulas the claim follows, assuming it for T -,Cl -, and $\mathcal{T}$ atoms.

The second claim easily follows because all conditions in the definition of $\equiv$ are independent of substitution by equal terms.

□

We show that $\equiv$ behaves nicely on formulas containing Cl and $\dot{\rightarrow}$, and write $w\ \mathcal{S}\ A$ for $w \equiv_A w$. $\mathcal{S}$ is an extension of the realisation relation $\mathfrak{R}$ of the previous chapter. We write $\mathfrak{R}$ instead of $\mathcal{S}$ in the following.

Lemma 50 *Assume $w, v \in \mathbb{W}$, and that s, t are terms.*

- $w \equiv_{\mathsf{Cl}(t)} v \Rightarrow (w = v = \langle \ulcorner \mathsf{Cl} \urcorner, w_2, \cdots, w_n\rangle \wedge (\forall y \in \mathbb{W})(\forall [\vec{s}])(y\ \mathcal{S}\ \mathsf{T}(t)[\vec{s}] \Rightarrow (\exists 2 \le i \le n) y = w_i))$
- *The above mentioned w_i are pairwise different.*
- $\mathsf{Cl}(t)$ *and $w \equiv_{\mathsf{T}(t)} v$ imply $w = v$.*

- *The following biconditional holds.*

$$\begin{aligned} w \equiv_{\mathsf{T}(s \dot{\rightarrow} t)} v \quad \Leftrightarrow \quad & (w = \langle \ulcorner \rightarrow \urcorner, \langle w_{2,1}, w_{2,2} \rangle, \cdots, \langle w_{n,1}, w_{n,2} \rangle \rangle \wedge \\ & v = \langle \ulcorner \rightarrow \urcorner, \langle w_{2,1}, v_{2,2} \rangle, \cdots, \langle w_{n,1}, v_{n,2} \rangle \rangle \wedge \\ & \langle \ulcorner \mathsf{Cl} \urcorner, w_{2,1}, \cdots, w_{n,1} \rangle \ \mathcal{S} \ \mathsf{Cl}(s) \wedge \\ & (\forall 2 \leq i \leq n)(\forall [\vec{t}])(w_{i,1} \ \mathcal{S} \ \mathsf{T}(s)[\vec{t}] \Rightarrow w_{i,2} \equiv_{\mathsf{T}(t)[\vec{t}]} v_{i,2}) \end{aligned}$$

Proof. The first, second, and third claim follow by an easy induction on the ordinal stages of the fixed point construction. For the fourth claim, let us prove the direction from the right to the left. So assume for an ordinal α, for a w given as $\langle \ulcorner \rightarrow \urcorner, \langle w_{1,0}, w_{,1} \rangle, \cdots, \langle w_{n,0}, w_{n,1} \rangle \rangle$, and a v given as $\langle \ulcorner \rightarrow \urcorner, \langle w_{1,0}, v_{1,1} \rangle, \cdots, \langle w_{n,0}, v_{n,1} \rangle \rangle$ the following.

- $P^{\alpha}(\langle \ulcorner \mathsf{Cl} \urcorner, w_{1,0}, \cdots, w_{n,0} \rangle, \langle \ulcorner \mathsf{Cl} \urcorner, w_{1,0}, \cdots, w_{n,0} \rangle, \mathsf{Cl}(s), 0)$
- $(\forall 2 \leq i \leq n) \forall [\vec{s}]((\exists \sigma) P^{\sigma}(w_{i,1}, w_{i,1}, s[\vec{s}], 0) \Rightarrow (\exists \tau) P^{\tau}(w_{i,2}, v_{i,2}, t[\vec{s}], 0))$

With help of lemmas 47 and 48 we derive

$$(\forall 1 \leq i \leq n) \forall [\vec{s}](P^{\alpha}(w_{i,0}, w_{i,0}, s[\vec{s}], 1) \vee (\exists \tau) P(w_{i,1}, v_{i,1}, t[\vec{s}], 0)).$$

A Σ reflection together with monotonicity deliver a τ larger than α such that

$$(\forall 1 \leq i \leq n) \forall [\vec{s}](P^{\tau}(w_{i,0}, w_{i,0}, s[\vec{s}], 1) \vee P^{\tau}(w_{i,1}, v_{i,1}, t[\vec{s}], 0)).$$

This implies together with the other assumptions

$$P^{\tau+1}(w, v, s \dot{\rightarrow} t, 0).$$

The direction from left to right follows from lemma 47. □

3.3.2 The extended formalism of addresses and pointers

For the realisation of the formulas containing Cl, $\dot{\rightarrow}$, it is convenient to enlarge the set of addresses from chapter 2.

Definition 51 *Let w, v be arbitrary words, and z a word unequal zero consisting only of zeros, and $\ulcorner\wedge\urcorner, \ulcorner\mathsf{Cl}\urcorner, \ulcorner\rightarrow\urcorner$ pairwise different words, different from ϵ. Then the following are addresses with address head w.*

- $w.\ulcorner\wedge\urcorner.0$, $w.\ulcorner\wedge\urcorner.1$
- $w.\ulcorner\mathsf{Cl}\urcorner.z$
- $w.\ulcorner\rightarrow\urcorner.z.0$, $w.\ulcorner\rightarrow\urcorner.z.1$

We let z contain only zeros for technical reasons, this restriction is not essential. $w.\ulcorner\wedge\urcorner.v_0$ behaves like $w.v_0$ in chapter 2. (Extended) CD parts are build from (extended) addresses as demonstrated there. Compared to CDs from chapter 2 the extended CDs contain an additional restriction which allows a sensible interpretation of content stored at addresses containing $\ulcorner\rightarrow\urcorner$:

Definition 52 (Extended CDs) *An extended CD ρ is a finite set of extended CD parts where the following conditions are fulfilled.*

- *Let w, v_0, v_1 be words. Then left sides of at most one of the following forms occur in ρ.*
  - w
  - $w.\ulcorner\wedge\urcorner.v_0$
  - $w.\ulcorner\mathsf{Cl}\urcorner.v_0$
  - $w.\ulcorner\rightarrow\urcorner.v_0.v_1$
- *if $w.\ulcorner\rightarrow\urcorner.v_0.v_1$ occurs as left side, then also $w.\ulcorner\rightarrow\urcorner.v_0.\overline{sg}(v_1)$.*
- *No address occurs twice as the left side of an (extended) CD part of ρ.*

We use the same notations and abbreviations for addresses and CDs as in the previous chapter. Extended addresses, extended CD parts, and extended CDs are called addresses, CD parts, and CDs from now on. In the following, we define important functions on CDs in analogy to chapter 2. We define a new construction function, we call again con.

Definition 53 (Construction function con) *The function* $\mathsf{con} : \mathbb{W} \times \mathbb{W} \to \mathbb{W}$ *is given by the following algorithm to calculate* $\mathsf{con}(\rho, \breve{v})$*:*

If ρ is not a CD or $\breve{v}$ is not an address, we stipulate $\mathsf{con}(\rho, \breve{v}) := \epsilon$, which takes the role of an error output. In all other cases, we execute the following definition by cases.

Case 1 *There is a CD part of the form $\breve{v} \to \breve{w}$:*

$$\mathsf{con}(\rho, \breve{v}) := \mathsf{con}(\rho, \breve{w}).$$

Case 2 *$\breve{v}.\ulcorner\wedge\urcorner.0$ and $\breve{v}.\ulcorner\wedge\urcorner.1$ occur as left side of a CD part:*

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner\wedge\urcorner, \mathsf{con}(\rho, \breve{v}.\ulcorner\wedge\urcorner.0), \mathsf{con}(\rho, \breve{v}.\ulcorner\wedge\urcorner.1)\rangle.$$

Case 3 *Only $\breve{v}.\ulcorner\wedge\urcorner.i$ but not $\breve{v}.\ulcorner\wedge\urcorner.j$, for $0 \leq i \neq j \leq 1$ occurs as left side of a CD part:*

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner\vee\urcorner, i, \mathsf{con}(\rho, \breve{v}.\ulcorner\wedge\urcorner.i)\rangle.$$

Case 4 *$\breve{v}.\ulcorner\mathsf{Cl}\urcorner.z_1, \cdots, \breve{v}.\ulcorner\mathsf{Cl}\urcorner.z_n$ ordered by the length of the z_i but no other addresses of this form occur as left sides of a CD part:*

$$\begin{aligned} \mathsf{con}(\rho, \breve{v}) \quad := \quad & \langle \ulcorner\mathsf{Cl}\urcorner, \mathsf{con}(\rho, \breve{v}.\ulcorner\mathsf{Cl}\urcorner.z_1), \cdots \\ & \mathsf{con}(\rho, \breve{v}.\ulcorner\mathsf{Cl}\urcorner.z_n)\rangle. \end{aligned}$$

Case 5 *$\breve{v}.\ulcorner\to\urcorner.z_1.0, \breve{v}.\ulcorner\to\urcorner.z_1.1, \cdots, \breve{v}.\ulcorner\to\urcorner.z_n.0, \breve{v}.\ulcorner\to\urcorner.z_n.1$ ordered by the length of the z_i but no other addresses of this form occur as left sides of a CD part:*

$$\begin{aligned} \mathsf{con}(\rho, \breve{v}) \quad := \quad & \langle \ulcorner\to\urcorner, \langle \mathsf{con}(\rho, \breve{v}.\ulcorner\to\urcorner.z_1.0), \mathsf{con}(\rho, \breve{v}.\ulcorner\to\urcorner.z_1.1)\rangle, \cdots, \\ & \langle \mathsf{con}(\rho, \breve{v}.\ulcorner\to\urcorner.z_n.0), \mathsf{con}(\rho, \breve{v}.\ulcorner\to\urcorner.z_n.1)\rangle\rangle. \end{aligned}$$

Case 6 *There is a CD part of the form $\breve{v} : w$:*

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner\mathsf{W}\urcorner, w\rangle.$$

Case 7 There is a CD part of the form $\breve{v} : \langle \ulcorner \mathsf{T} \urcorner, w \rangle$:

$$\mathsf{con}(\rho, \breve{v}) := \langle \ulcorner \mathsf{T} \urcorner, w \rangle.$$

Case 8 Cases 1 until 7 are not satisfied. Then $\mathsf{con}(\rho, \breve{v}) := \epsilon$.

To calculate the computational content, we additionally take the number of components of realisers of Cl-formulas into account. The related address relation R^*_ρ, used for this definition, is the one from page 31 extended to the new sorts of addresses in the obvious way.

Definition 54 ($\mathsf{con_W}^+$) *The function $\mathsf{con_W}^+ : \mathbb{W} \times \mathbb{W} \to \mathbb{W}$ is defined by the following algorithm for the calculation of $\mathsf{con_W}^+(\rho, \breve{v})$:*

Step 1: Find all addresses $\breve{w}$ for which $R^*_\rho(\breve{v}, \breve{w})$ holds. They form a set M.

Step 2: Output the maximum with respect to the lexicographic ordering over all words u such that there exists a $\breve{w}$ with either $\breve{w} \in M$ and $\breve{w} : u$ being a part of ρ or $\breve{w}.\ulcorner \mathsf{Cl} \urcorner.u \in M$. If there is no such word u, output ϵ.

$\mathsf{con_W}^+$ is a polytime function for the same reasons as $\mathsf{con_W}$ is. Since $\mathsf{con_W}^+$ behaves on addresses giving information for Cl-free formulas as $\mathsf{con_W}$, all its important properties that are used in the upper bound proof are preserved. We will write $\mathsf{con_W}$ for $\mathsf{con_W}^+$ in this chapter.

3.3.3 Realisation relation for negative formulas

Let us formulate some general considerations about the realisation approach. As we have mentioned before, the crucial difference to usual realisation approaches with functionals is that a functional realiser γ of the formula $A_0 \to A_1$ does not deliver a realiser of A_1 only from a realiser of A_0 but also uses realisers of side formulas. Still, we cannot expect the functional realiser to allow arbitrary $\mathfrak{R}$-realisers of the side formulas as input, because to make a realisation of the axiom of choice possible, the produced $\mathfrak{R}$-realiser has to be a function only dependent on the $\mathfrak{R}$-realiser of A_0 in a certain sense. So, we will allow only inputs ρ for γ giving a (up to a certain extend) fixed

$\mathfrak{R}$ -realisation information for side formulas. Nevertheless, a total fixation of the $\mathfrak{R}$ -realisation information is not possible since the realisation functionals we will produce do not operate in a functional way on $\mathfrak{R}$ -realisation information. This is the reason why the relation $\equiv$ was introduced.

We deal with the missing functionality along the following lines. In parallel to the address pointer realisation formalism, we employ a standard realisation approach with functionals that work on $\mathfrak{R}$ -realisers instead of CDs. This formalism on $\mathfrak{R}$ -realisers can be seen as a projection of the address pointer realisation formalism since it abstracts away the particular form in which $\mathfrak{R}$ -realisation information is stored. This projection controls at the same time the set of permitted $\mathfrak{R}$ -realisers of the side formulas when a negative formula is realised, and guarantees that $\mathfrak{R}$ -realisers stored in the output of γ depend only on the $\mathfrak{R}$ -realisers stored in the input. The reason we cannot drop the address-pointer formalism completely in favour of the above described formalism on $\mathfrak{R}$ -realisers is of course its inefficiency. It would not deliver a feasible upper bound for our theory.

Let us now specify our strategy, and define a suitable realisation relation for sequents consisting of formulas of $\mathfrak{A}$. First, we show how to realise sequences of positive formulas. This is done by a realisation relation $\mathfrak{r}_b^+$ defined similarly as $\mathfrak{r}_b$ for $\mathsf{T_{PT}}$ but containing a tuple of words as additional arguments. They help to define a realisation relation $\mathfrak{r}'$ for (single) formulas of the form $A_0 \to A_1$ where A_0, A_1 are positive. As expected, we use functionals as realisers, whose input is a CD with $\mathfrak{R}$ -realisation information for A_0 and the side formulas Γ.

Definition 55 ($\mathfrak{r}_b^+$) *The four relata of $\mathfrak{r}_b^+$ are a word ρ, a recursive address finder* [5] *b, a sequence $A_1, \cdots, A_n$ of positive formulas, and a tuple of words $< c_1, \cdots, c_n >$. The relation holds on this relata, written as $\rho\ \mathfrak{r}_b^+\ A_1, \cdots, A_n < c_1, \cdots, c_n >$, iff for $1 \leq i \leq n$, we have*

$$\mathsf{con}(\rho, b(\rho, i)) \equiv_{A_i} c_i$$

[5] In contrast to the last chapter, address finders are not assumed to be feasible in this chapter. Later, they will just be arguments of type $\mathbb{W}^2 \to \mathbb{W}$ of feasible realisation functionals.

In such a context, we call $< c_1, \cdots, c_n >$ a projection of ρ.

In the following, we write $\mathfrak{r}_b$ instead of $\mathfrak{r}_b^+$ for any address finder b.

Definition 56 *Let b be an address finder with n relevant inputs. Let w be an arbitrary word. Then the address finder b_w is defined as follows* [6]*.*

$$b_w(\rho, i) = \begin{cases} b(w,i), & \textit{if } 1 \leq i \leq n \\ \mathsf{MA}(\rho), & \textit{if } i = n+1 \end{cases}$$

Definition 57 *For two CDs α, ρ we say that α extends ρ exactly if the following conditions are fulfilled.*

- $\rho \subseteq \alpha$
- *$\alpha \setminus \rho$ does not contain an address head ρ contains.*

Definition 58 ($\mathfrak{r}'$) *The realisation relation $\mathfrak{r}'$ has five relata. We give the intended meaning of the relata in the brackets. The first is of type*

$$\mathbb{W}^{\mathbb{W} \times \mathbb{W}^{\mathbb{W}}} \times (\mathbb{W}^{\mathbb{W}})^3$$

given as $< \gamma_1, \cdots, \gamma_4 >$ (the functional realiser together with bounds), the second is a formula of $\mathfrak{A}$ of the form $A_0 \to A_1$ for positive A_0, A_1 (the formula to be realised), the third is a positive sequence Γ (the side formulas), the fourth is a tuple of words $< c_1, \cdots, c_n >$ (restrictions for the allowed $\mathfrak{R}$ -realisers of the side formulas) and the fifth a function $\tilde{\gamma}$ of type $\mathbb{W}^{\mathbb{W}}$ (a projection of the functional realiser). The relation $\mathfrak{r}'$ holds of its five relata, written as $< \gamma_1, \cdots, \gamma_4 > \mathfrak{r}' A$ relative to $\Gamma, < c_1, \cdots, c_n, \tilde{\gamma} >$, exactly if for any address finder b, any substitution $[\vec{s}]$, any word v, and any α with α $\mathfrak{r}_b$ $A_0[\vec{s}], \Gamma[\vec{s}]$ $< v, c_1, \cdots, c_n >$ the following conditions hold.

- *$\gamma_1, \cdots, \gamma_4$ are recursive.*
- *$\gamma_1(\alpha, b)$ extends α*

[6]We extend the function MA from chapter 2 to the larger set of CDs in the obvious way.

- $\gamma_1(\alpha, b)\ \mathfrak{r}_{b_\alpha}\ A_0[\vec{s}], \Gamma[\vec{s}], A_1[\vec{s}] < v, c_1, \cdots, c_n, \tilde{\gamma}(v, c_1, \cdots, c_n) >$,
- $\mathsf{MA}(\gamma_1(\alpha, b)) \leq \mathsf{MA}(\alpha) + \gamma_2(\mathsf{W}_b(\alpha))$
- $\gamma_1(\alpha, b) \leq \alpha + \gamma_3(\mathsf{W}_b(\alpha), \mathsf{MA}(\alpha))$ [7]
- $\mathsf{W}_{b_\alpha}(\gamma_1(\alpha, b)) \leq \gamma_4(\mathsf{W}_b(\alpha))$

We define a new realisation relation for sequents consisting of formulas of $\mathfrak{A}$ by fusing the realisation relations $\mathfrak{r}_b$ and $\mathfrak{r}'$. Since this relation is always relative to an address finder b, we can call it $\mathfrak{R}_b$ without danger of confusion with the already defined $\mathfrak{R}$. In the following, we give some intuition how the general realisation relation works and set some notational conventions to prepare its definition.

The extended realisation relation holds (amongst other relata) between a word ρ and a positive sequence Γ, or between objects of type

$$\mathbb{W} \times \overbrace{\mathbb{W}^{\mathbb{W} \times \mathbb{W}^{\mathbb{W}}} \times \cdots \times \mathbb{W}^{\mathbb{W} \times \mathbb{W}^{\mathbb{W}}}}^{n \text{ times}} \times (\mathbb{W}^{\mathbb{W}})^3$$

and sequences containing $n > 0$ negative formulas. For the case $n = 0$ we just use the realisation relation $\mathfrak{r}_b$ for positive formulas from definition 55. We concentrate now on the case where a sequence Γ with $n > 0$ negative formulas is realised, and give the intended meaning of the components of the realiser in this case. The first component is a realiser of the positive formulas of Γ in the sense of definition 55. The next n components are realisers of the negative formulas. The last three components are bounds for the realisers of the negative formulas that work like $\gamma_2, \gamma_3, \gamma_4$ in definition 58 but simultaneously for all functional realisers of negative formulas. Now a bit of notation.

- $\wp$, $< \rho, \gamma_1, \cdots, \gamma_n, P_1, P_2, P_3 >$, or similar notations always denote arbitrary objects of type

$$\mathbb{W} \times \overbrace{\mathbb{W}^{\mathbb{W} \times \mathbb{W}^{\mathbb{W}}} \times \cdots \times \mathbb{W}^{\mathbb{W} \times \mathbb{W}^{\mathbb{W}}}}^{n \text{ times}} \times (\mathbb{W}^{\mathbb{W}})^3,$$

[7]We consider γ_3 as a function from $\mathbb{W}$ to $\mathbb{W}$. Its intended input is a pair. We make similar assumptions in other places. W_b is defined analogously as in chapter 2.

where for $\wp$ n is determined by the context. We call these objects generalised CDs.

- For $\wp$ given as $< \rho, \gamma_1, \cdots, \gamma_n, P_1, P_2, P_3 >$ we denote its first component ρ by $\wp^+$. The other components are denoted by $\wp^-$, so we have $\wp =< \wp^+, \wp^- >$.
- We write $\mathsf{MA}(\wp)$ for $\mathsf{MA}(\wp^+)$ and analogously for W_b.
- $\vec{\gamma}$ abbreviates $\gamma_1, \cdots, \gamma_n$ and $\vec{P}$ abbreviates P_1, P_2, P_3. The arity of $\vec{\gamma}$ is determined by the context.
- For a sequence containing n negative formulas we write NF_i for $1 \leq i \leq n$ for the i-th negative formula.
- We write Γ^+ for the sequence Γ with all negative formulas deleted.
- We write $\Gamma^+, NF_1, \cdots, NF_n$ for any sequence which contains exactly the displayed formulas with the same multiplicity, whose positive formulas are ordered as the ones in Γ^+, and whose negative formulas are ordered as the NF_i. Our realisation approach does not discriminate any of these sequencies.

Now, we can define a realisation relation for sequences of formulas of $\mathfrak{A}$.

Definition 59 ($\mathfrak{R}_b$) *The realisation relation $\mathfrak{R}_b$ has four relata. The first is a sequence Γ with m positive and n negative formulas (the sequence that has to be realised). If n equals 0, the second relata is a word (CD for the positive sequence), the third is a tuple of words $< c_1, \cdots, c_m >$ (restrictions for the $\mathfrak{R}$ -realisers of Γ^+), the fourth is an empty tuple. If $n > 0$, the second relata is of type*

$$\mathbb{W} \times \overbrace{\mathbb{W}^{\mathbb{W} \times \mathbb{W}^{\mathbb{W}}} \times \cdots \times \mathbb{W}^{\mathbb{W} \times \mathbb{W}^{\mathbb{W}}}}^{n\ times} \times (\mathbb{W}^{\mathbb{W}})^3$$

(realiser of the mixed sequence), the third is a tuple of words $< c_1, \cdots, c_m >$ (restrictions for the realisers of Γ^+), the fourth is a tuple $\vec{\tilde{\gamma}}$ of n functions of type $\mathbb{W}^{\mathbb{W}}$ (projections for the functional realisers of the negative formulas). We write $\wp$ $\mathfrak{R}_b$ $\Gamma < c_1, \cdots, c_m, \vec{\tilde{\gamma}} >$ iff the relation $\mathfrak{R}_b$ holds between the

above mentioned relata. We drop $\vec{\tilde{\gamma}}$ if it is the empty tuple. We define $\mathfrak{R}_b$ by the following definition by cases.

Assume $n = 0$**.**

$$\wp\ \mathfrak{R}_b\ \Gamma < c_1, \cdots, c_m >:\Leftrightarrow \wp\ \mathfrak{r}_b\ \Gamma < c_1, \cdots, c_m >$$

Assume $n > 0$**.** *We define $\mathfrak{R}_b$ inductively on the sum of quantifiers having $\rightarrow$ in their scope.*

- *First, assume that all negative formulas NF_i are of the form $\ell NF_i \rightarrow rNF_i$ for $\ell NF_i, rNF_i$ positive. Then*

  $$\wp\ \mathfrak{R}_b\ \Gamma < c_1, \cdots, c_m, \vec{\tilde{\gamma}} >$$

  iff

  - $\wp^+\ \mathfrak{r}_b\ \Gamma^+ < c_1, \cdots, c_m >$*, and*
  - *for all $1 \leq i \leq n$, we have*

    $$< \gamma_i, \vec{P} > \mathfrak{r}' NF_i \text{ relative to } \Gamma^+, < c_1, \cdots, c_m, \tilde{\gamma}_i >$$

- *Assume that NF_i is of the form $(\forall x)A[x]$. Then we have*

  $$\wp\ \mathfrak{R}_b\ \Gamma^+, NF_1, \cdots, (\forall x)A[x], \cdots, NF_n < c_1, \cdots, c_m, \vec{\tilde{\gamma}} >:\Leftrightarrow$$
  $$\wp\ \mathfrak{R}_b\ \Gamma^+, NF_1, \cdots, A[u], \cdots, NF_n < c_1, \cdots, c_m, \vec{\tilde{\gamma}} >$$

  for a fresh variable u.

- *Assume that NF_i is of the form $(\exists x)A[x]$. Then we have*

  $$\wp\ \mathfrak{R}_b\ \Gamma^+, NF_1, \cdots, (\exists x)A[x], \cdots, NF_n < c_1, \cdots, c_m, \vec{\tilde{\gamma}} >:\Leftrightarrow$$
  $$\wp\ \mathfrak{R}_b\ \Gamma^+, NF_1, \cdots, A[t], \cdots, NF_n < c_1, \cdots, c_m, \vec{\tilde{\gamma}} >$$

  for some term t.

If $\wp\ \mathfrak{R}_b\ \Gamma^+, NF_1, \cdots, NF_n < c_1, \cdots, c_m, \vec{\tilde{\gamma}} >$, we call $< c_1, \cdots, c_m, \vec{\tilde{\gamma}} >$ the projection of $\wp$. Next, we show that the realisation relation $\mathfrak{R}_b$ is sensible, by proving some standard properties.

Lemma 60 *Let Γ a sequence of formulas. We use the abbreviation $\vec{s} = \vec{t}$ for $s_1 = t_1 \wedge \cdots \wedge s_n = t_n$. Then the following holds for all substitutions $[\vec{s}]$, $[\vec{t}]$ and all address finders b.*

- $\wp \ \mathfrak{R}_b \ \Gamma[\vec{x}] < c_1, \cdots, c_n, \vec{\bar{\gamma}} > \Rightarrow \wp \ \mathfrak{R}_b \ \Gamma[\vec{s}] < c_1, \cdots, c_n, \vec{\bar{\gamma}} >$.
- $\vec{s} = \vec{t} \Rightarrow (\wp \ \mathfrak{R}_b \ \Gamma[\vec{s}] < c_1, \cdots, c_n, \vec{\bar{\gamma}} > \Leftrightarrow \wp \ \mathfrak{R}_b \ \Gamma[\vec{t}] < c_1, \cdots, c_n, \vec{\bar{\gamma}} >)$
- $\wp \ \mathfrak{R}_b \ \Gamma, (\exists x)A[x] < c_1, \cdots, c_n, \vec{\bar{\gamma}} > \Rightarrow \wp \ \mathfrak{R}_b \ \Gamma, A[t] < c_1, \cdots, c_n, \vec{\bar{\gamma}} >$, *for some term t.*
- *Assume that A is a positive formula. Then we have*

  $$\begin{aligned} \wp \ \mathfrak{R}_b \ \Gamma, (\forall x)A[x] \quad & < c_1, \cdots, c_n, \vec{\bar{\gamma}} > \Rightarrow \\ & \wp \ \mathfrak{R}_{b'} \ \Gamma, (\forall x)A[x], A[t] < c_1, \cdots, c_n, c_n, \vec{\bar{\gamma}} > \end{aligned}$$

  for any term t, where b' finds the last address twice.
- *Assume that A is a negative formula. Then we have*

  $$\wp \ \mathfrak{R}_b \ \Gamma, (\forall x)A[x] < c_1, \cdots, c_n, \vec{\bar{\gamma}} > \Rightarrow \wp \ \mathfrak{R}_b \ \Gamma, A[t] < c_1, \cdots, c_n, \vec{\bar{\gamma}} >$$

  for any term t.

Proof.
We prove the first assertion: Assume $\wp \ \mathfrak{R}_b \ \Gamma[\vec{x}] < \vec{c}, \vec{\bar{\gamma}} >$. That $\wp^+$ works correctly for $\Gamma^+(\vec{s})$ follows from the lemma 49. Let us prove the correctness of $\wp^-$ by induction on the sum of quantifiers binding $\rightarrow$. For a formula A of the form $A_0 \rightarrow A_1$ for A_0, A_1 positive the claim follows from the closedness of the relation $\mathfrak{r}'$ under substitution. The induction step is trivial.

The second claim of the lemma holds because the relation $\equiv$ is closed under substitutions of the terms of its formula argument by equal terms.

Let us prove that the claim for the existential quantifier holds. Assume $\wp \ \mathfrak{R}_b \ \Gamma, (\exists x)A[x]$. If A is a negative formula the claim trivially holds. So, assume that A is positive. $\wp^+$ clearly works for a substitution by a suitable closed term t. Let us argue that the functional components of $\wp^-$ do their job with respect to t. Let γ_i receive an input ρ with

$$\rho \ \ \mathfrak{r}_{\ b} \ \ell NF_i, \Gamma^+, A[t][\vec{s}] < v, \vec{c} > .$$

We have that $w_0 \equiv_{A(t)[\vec{s}]} w_1$ implies $w_0 \equiv_{(\exists x)A[x][\vec{s}]} w_1$ for all $w_0, w_1 \in \mathbb{W}$ and all $[\vec{s}]$ which implies

$$\rho \ \ \mathfrak{r}_{\ b} \ \ell NF_i, \Gamma^+, (\exists x)A[x][\vec{s}] < v, \vec{c} > .$$

The assumption yields

$$\gamma_i(\rho, b) \ \ \mathfrak{r}_{\ b_\rho} \ \ell NF_i, \Gamma^+, (\exists x)A[x], rNF_i[\vec{s}] < v, \vec{c}, \tilde{\gamma}_i(v) > .$$

γ_i extends ρ which implies

$$\gamma_i(\rho, b) \ \ \mathfrak{r}_{\ b_\rho} \ \ell NF_i, \Gamma^+, A[t], rNF_i[\vec{s}] < v, \vec{c}, \tilde{\gamma}_i(v) > .$$

Let us prove that the claim for the universal quantifier holds. Assume $\wp \ \mathfrak{R}_b \ \Gamma, (\forall x)A[x]$ for A positive. $\wp^+$ clearly works for any term t. Let us argue that the functional components of $\wp^-$ do their job with respect to an arbitrary t. Let γ_i receive an input ρ with

$$\rho \ \ \mathfrak{r}_{\ b} \ \ell NF_i, \Gamma^+, (\forall x)A[x], A[t][\vec{s}] < v, c_1, \cdots, c_n, c_n > .$$

We have

$$\gamma_i(\rho, b) \ \ \mathfrak{r}_{\ b_\rho} \ \ell NF_i, \Gamma^+, (\forall x)A[x], rNF_i[\vec{s}] < v, c_1, \cdots, c_n, \tilde{\gamma}_i(v) > .$$

Since γ_i extends its input, we derive

$$\gamma_i(\rho, b) \ \ \mathfrak{r}_{\ b_\rho} \ \ell NF_i, \Gamma^+, (\forall x)A[x], A[t], rNF_i[\vec{s}] < v, c_1, \cdots, c_n, c_n, \tilde{\gamma}_i(v) >$$

as required. The claim for A being a negative formula trivially holds.
$\square$

3.4 Application of the formalism to T^+

In the following, we work with two realisation functionals, the first working on CDs and the second on their projections. Their interplay allows the realisation of the axiom of choice. Note that the second functional does not have to be feasible.

3.4.1 Stating the main claim

We formulate a main theorem, which is very similar as the one in chapter 2.

Theorem 61 *Assume* $\mathsf{T}^+ \vdash \Gamma \Rightarrow D$ *quasi cut free, where* Γ *contains* m *positive and* n *negative formulas. Then there exists a feasible functional* p *(realisation functional), feasible functionals* κ, δ, γ *(bounds for the realisation functional) and a recursive functional* $\tilde{p}$ *(projection of the realisation functional) such that the following properties hold for all address finders* b*, all substitutions* $[\vec{s}]$*, all* $\vec{c} \in \mathbb{W}^m$*, all* $\vec{\tilde{\gamma}} \in (\mathbb{W}^{\mathbb{W}})^n$*, all* $\wp$ *such that* $\wp \; \mathfrak{R}_b \; \Gamma[\vec{s}] < \vec{c}, \vec{\tilde{\gamma}} >$.

- $p(\wp, b) \; \mathfrak{R}_{b_{\wp^+}} \; \Gamma, D[\vec{s}] \; \tilde{p}(\vec{c}, \vec{\tilde{\gamma}})$

-
  - $\mathsf{MA}(p(\wp, b)) \leq \mathsf{MA}(\wp) + \kappa(\mathsf{W}_b(\wp), P_1, P_3)$
  - $p(\wp, b)^+ \leq \wp^+ + \delta(\mathsf{W}_b(\wp), \mathsf{MA}(\wp), \vec{P})$.
  - $\mathsf{W}_{b_{\wp^+}}(p(\wp, b)) \leq \gamma(\mathsf{W}_b(\wp), P_3)$

-
  - $p(\wp, b)_{P_1}$ *is a feasible functional of* P_1 *and* P_3.
  - $p(\wp, b)_{P_2}$ *is a feasible functional of* P_1, P_2 *and* P_3.
  - $p(\wp, b)_{P_3}$ *is a feasible functional of* P_3.

Note that the new address for the $|\Gamma^+| + 1$-th positive formula is irrelevant if D is a negative formula.

There would be the possibility to use alternatively the following realisation approach which does not use projections: We extend the relation $\equiv$ to an equivalence relation $\equiv^*$ relative to its word arguments with the formula fixed, using its transitive closure [8] . The functions $\vec{\tilde{\gamma}}$ and the realisation relation

[8] In the following, we show that transitivity relative to the word arguments for a fixed formula fails for $\equiv$. We let $sg : \mathbb{W} \to \mathbb{W}$ be the following function.

$$sg(w) = \begin{cases} 0, & \text{if } w = 0 \\ 1, & \text{else} \end{cases}$$

We let t_{sg} denote the corresponding term on word inputs. Now, a counterexample can be build for the formula $(\exists x)(\mathsf{d_W}(x, x, 0, 0) = 0 \wedge t_{sg}x = x \wedge \mathsf{T}(x \doteq 0 \; \dot{\vee} \; x \doteq 1 \dot{\rightarrow} x \leq_{\mathsf{W}} 1)) := (\exists x)A[x]$ and the following three realisers:

p are assumed to respect these equivalence classes. We do not choose this approach because the property that

$$s_0 \equiv^*_{\exists x A[x]} s_1 \Rightarrow s_0 \equiv^*_{A[t]} s_1 \text{ for a term } t$$

would fail which makes the realisation of the existential quantifier left rule awkward. During the proof of the main theorem, we tacitly assume that $p(\wp, b)^+$ extends $\wp^+$. We also tacitly assume that $\lambda x.\kappa(x, P_1, P_3)$ always bounds the functions P_1, P_3. We make analogous assumptions about δ, γ.
We have to show that the realisation functional works for all possible substitutions $[\vec{s}]$. In unproblematic cases, we will ignore this to increase readability.

3.4.2 Realisation functions for the axioms

Since weakening is restricted, we have to realise axioms with positive side formulas.

The axioms of $\mathsf{T_{PT}}$ only contain positive formulas, therefore they can be realised as in chapter 2. We only have to add suitable projections of the realisation functions, which can be found easily. Also the $\mathcal{T}$ reflection axioms, $(\mathcal{T} - \mathsf{CD})$, $(\mathcal{T} - \mathsf{UP})$, can be realised easily using pointers.

$$\langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle,$$
$$\langle \ulcorner \rightarrow \urcorner, \langle\langle 0, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \mathsf{W} \urcorner, 0\rangle\rangle, \langle\langle 1, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \mathsf{W} \urcorner, 0\rangle\rangle\rangle\rangle\rangle := \rho_1$$
$$\langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle,$$
$$\langle \ulcorner \rightarrow \urcorner, \langle\langle 0, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \mathsf{W} \urcorner, 0\rangle\rangle, \langle\langle 1, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \mathsf{W} \urcorner, 1\rangle\rangle\rangle\rangle\rangle := \rho_2$$
$$\langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle,$$
$$\langle \ulcorner \rightarrow \urcorner, \langle\langle 0, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \mathsf{W} \urcorner, 1\rangle\rangle, \langle\langle 1, \epsilon\rangle, \langle \ulcorner \wedge \urcorner, \langle \ulcorner \mathsf{W} \urcorner, \epsilon\rangle, \langle \ulcorner \mathsf{W} \urcorner, 1\rangle\rangle\rangle\rangle\rangle := \rho_3$$

The formula is satisfied in the standard model exactly for the witnesses 0 and 1, since the equation $\mathsf{d_W}(x, x, 0, 0) = 0$ is exactly satisfied for words, and the signum function, which is correctly simulated by t_{sg} on words, has the only fixed points 0 and 1. We have

$$\rho_1 \equiv_{A[0]} \rho_2 \text{ and } \rho_2 \equiv_{A[1]} \rho_3,$$

so if $\equiv_{(\exists x)A[x]}$ is transitive, $\rho_1 \equiv_{(\exists x)A[x]} \rho_3$ has to hold. Nevertheless, it fails because $\rho_1 \mathfrak{R} A[1]$ fails, as well as $\rho_3 \mathfrak{R} A[0]$. So, there is no common witness for which both, ρ_1 and ρ_3, realise $(\exists x)A[x]$.

Class generation axioms

Let us realise the axiom

$$\Gamma, \mathsf{Cl}(s), \mathsf{Cl}(t) \Rightarrow \mathsf{Cl}(s \mathbin{\dot\wedge} t),$$

where Γ is a positive sequence. Assume that $\alpha\ \mathfrak{r}_b\ \Gamma, \mathsf{Cl}(s), \mathsf{Cl}(t)$. (We suppress the projection of α for readability.) We have to construct all possible realisers of $\mathsf{T}(s \dot\wedge t)$ which are just all the pairs of a possible realiser of $\mathsf{T}(s)$ and a possible realiser of $\mathsf{T}(t)$. These realisers are stored under the extensions of $b(\alpha, |\Gamma| + 1)\!\downarrow\!.\mathsf{Cl}$ or $b(\alpha, |\Gamma| + 2)\!\downarrow\!.\mathsf{Cl}$, respectively. For any CD ρ, let the function $\rho_{\mathsf{Cl}(s),v}$ give as output the $|v|$-th left side of ρ which is an extension of $b(\alpha, |\Gamma| + 1)\!\downarrow\!.\mathsf{Cl}$ ordered by the length of the $v \in \mathsf{W}$ extending it. We give out ϵ if this does not make sense. We use number notation for the input v. We assume that $\mathsf{T}(s)$ has n and $\mathsf{T}(t)$ m possible realisers. A function yielding a realiser for $\Gamma, \mathsf{Cl}(s), \mathsf{Cl}(t), \mathsf{Cl}(s \mathbin{\dot\wedge} t)$ can be defined as follows, where M is a sufficiently large address head.

$$\begin{aligned}
f(\rho, b) \quad := \quad & \rho / M.\ulcorner\mathsf{Cl}\urcorner.0 \rightarrow \mathsf{MA}(\rho) + 1/ \\
& \mathsf{MA}(\rho) + 1.0 \rightarrow \rho_{\mathsf{Cl}(s),1} / \mathsf{MA}(\rho) + 1.1 \rightarrow \rho_{\mathsf{Cl}(t),1} / \cdots / \\
& M.\ulcorner\mathsf{Cl}\urcorner.\overbrace{00\cdots0}^{m \text{ times}} \rightarrow \mathsf{MA}(\rho) + m/ \\
& \mathsf{MA}(\rho) + m.0 \rightarrow \rho_{\mathsf{Cl}(s),1} / \mathsf{MA}(\rho) + m.1 \rightarrow \rho_{\mathsf{Cl}(t),m} / \cdots / \\
& M.\ulcorner\mathsf{Cl}\urcorner.\overbrace{00\cdots0}^{n\cdot m \text{ times}} \rightarrow \mathsf{MA}(\rho) + n \cdot m/ \\
& \mathsf{MA}(\rho) + n \cdot m.0 \rightarrow \rho_{\mathsf{Cl}(s),n} / \mathsf{MA}(\rho) + n \cdot m.1 \rightarrow \rho_{\mathsf{Cl}(t),m}
\end{aligned}$$

This function is clearly feasible and fulfils property 1. The until now suppressed projection $\tilde{p}$ of the realisation function can be defined analogously. The bounding property 2.1 is fulfilled because the necessary new maximal address M can be bounded by the old one plus a bound for $|\mathsf{W}_b(\rho) \times \mathsf{W}_b(\rho)|$. The bounding property 2.2 is fulfilled because we add at most $|\mathsf{W}_b(\rho) \times \mathsf{W}_b(\rho)|$ CD parts that can be bounded polynomially in $\mathsf{MA}(\rho)$ plus $\mathsf{W}_b(\rho) \times \mathsf{W}_b(\rho)$. 2.3 is fulfilled for the bounding polynomial $\mathsf{W}_b(\rho) \times \mathsf{W}_b(\rho)$. Clearly, we need to admit that class atoms have computational content in order to achieve the bounding properties 2.1, 2.2. Property 3 is trivially fulfilled. This finishes the proof for this class axiom.

We briefly switch to the other class axioms. The axiom $\Gamma \Rightarrow \mathsf{Cl}(x \dot{=} y)$ can be realised trivially. For $\Gamma, \mathsf{Cl}(s), \mathsf{Cl}(t) \Rightarrow \mathsf{Cl}(s \mathbin{\dot{\vee}} t)$, we use a similar realisation function as displayed above. The class axioms for quantifiers can be realised trivially because of the definition of the realisation relation $\mathfrak{R}$.

Here, it becomes clear that our approach cannot deal realise the sequent

$$\mathsf{Cl}(s), \mathsf{Cl}(t) \Rightarrow \mathsf{Cl}(s \rightarrow t)$$

since the number of possible realisers of $\mathsf{T}(s \rightarrow t)$ cannot be bounded polynomially in the number of possible realisers of $\mathsf{T}(s), \mathsf{T}(t)$ even if we allow only finite functions whose image is within the set of possible realisers of $\mathsf{T}(t)$.

$\dot{\rightarrow}$ axiom right

Let the axiom have the following form.

$$\Gamma, \mathsf{Cl}(s), \mathsf{T}(s) \rightarrow \mathsf{T}(t) \Rightarrow \mathsf{T}(s \dot{\rightarrow} t)$$

We produce the realiser of $\mathsf{T}(s \dot{\rightarrow} t)$ as follows. We can read of the possible realisers of $\mathsf{T}(s)$ from the realiser of $\mathsf{Cl}(s)$. Then, we apply the functional realiser of $\mathsf{T}(s) \rightarrow \mathsf{T}(t)$ consecutively to the possible realisers. If a possible realiser is an actual realiser for a substitution $[\vec{t}]$ this will yield an actual realiser of $\mathsf{T}(t)[\vec{t}]$. But if the possible realiser is not an actual one, we do not know whether the application of the functional realiser produces a reasonable result. Especially, we do not know whether the bounds are respected. Therefore, we have to enforce them using bounding functions that keep the maximal address, the computational content and the length under control. In addition, we can assume that the functional realisers applied to non actual realisers still have an output that is a CD, and extends their input since the functional realisers can be easily manipulated to do so.

Definition 62 *The function* $\mathsf{MA}bound : \mathbb{W}^2 \rightarrow \mathbb{W}$ *is defined as follows:* $\mathsf{MA}bound(\rho, w)$ *outputs* ρ *with all CD parts with address heads larger than* w *deleted.*

Definition 63 *The function* $\mathsf{W}bound : \mathbb{W}^2 \to \mathbb{W}$ *is defined as follows for* $\rho_0 \subseteq \rho_1$*:*

$$\mathsf{W}bound(\rho_0, \rho_1) = \begin{cases} \rho_1, & \textit{if } \mathsf{con}_{\mathsf{W}}(\rho_1, \mathsf{MA}(\rho_1)) = \epsilon \\ \rho_0, & \textit{else} \end{cases}$$

If $\rho_0 \subseteq \rho_1$ *does not hold, we give output* ϵ*.*

We write $\rho|_{\mathsf{MA}}w$ instead of $\mathsf{MA}bound(\rho, w)$ and write $\rho_1|_{\mathsf{W}}\rho_0$ instead of $\mathsf{W}bound(\rho_0, \rho_1)$. Assume that the polytime function $g : \mathbb{W} \to \mathbb{W}$ applied to $w \in \mathbb{W}$ gives a bound for the CD parts not build from words longer than w (excluding $\ulcorner\wedge\urcorner, \ulcorner\mathsf{Cl}\urcorner, \ulcorner\to\urcorner$).

We define the following bounding polynomial to keep the length of the CDs under control by definition of cases.

$$\begin{aligned} B(\rho, \epsilon, \vec{P}) &:= \rho \\ B(\rho, \mathsf{s}_i w, \vec{P}) &:= \rho + P_2(\mathsf{W}_b(\rho), \mathsf{MA}(\rho) + P_1(\mathsf{W}_b(\rho)) \times w + \mathsf{s}_i w) \times \mathsf{s}_i w + \\ &\quad\; g(\mathsf{W}_b(\rho) + \mathsf{MA}(\rho) + P_1(\mathsf{W}_b(\rho)) \times w + \mathsf{s}_i w) \times \mathsf{s}_i w \end{aligned}$$

Definition 64 *For an address finder* b *with* $n+1$ *relevant inputs, we define* $b\ddagger$ *as follows.*

$$b\ddagger(\rho, i) = \begin{cases} b(\rho, n+1), & \textit{if } i = 1 \\ b(\rho, i-1), & \textit{if } 1 < i \leq n+1 \end{cases}$$

Again, for any CD ρ, let the function $\rho_{\mathsf{Cl}(s),v}$ give as output the $|v|$-th left side of ρ which is an extension of $b(\rho, |\Gamma|+1)\downarrow.\mathsf{Cl}$ ordered by the length of the $v \in \mathsf{W}$ extending it. We give out ϵ if this does not make sense. We use number notation for the input v. We define an auxiliary function h such that $h(\rho, \gamma, \vec{P}, w, b)$ contains all components of a realiser of $\mathsf{T}(s \dot{\to} t)$ under the assumption that $< \rho, \gamma, \vec{P} > \ \mathfrak{R}_b\ \Gamma, \mathsf{Cl}(s), \mathsf{T}(s) \to \mathsf{T}(t) < \vec{c}, \tilde{\gamma} >$, and the $\mathfrak{R}$ -realiser of $\mathsf{Cl}(s)$ having $|w|$ components not counting its first component $\ulcorner\mathsf{Cl}\urcorner$ [9].

[9] We assume that the cut function $|$, that is used to cut with the bound B, works such that for two words ρ, w, if ρ is a CD then $\rho|w$ is a CD. This is guaranteed by occasionally cutting away more than necessary.

$$\begin{array}{rcl} h(\rho,\gamma,\vec{P},\epsilon,b) & := & \rho \\ h(\rho,\gamma,\vec{P},\mathsf{s}_i v,b) & := & \gamma(h(\rho,\gamma,v,b)/\mathsf{MA}(h(\rho,\gamma,v,b))+1 \to \rho_{\mathsf{Cl}(s),\mathsf{s}_i v} \\ & & |_{\mathsf{MA}}\mathsf{MA}(\rho)+P_1(\mathsf{W}_b(\rho))\times \mathsf{s}_i v+\mathsf{s}_i v|_{\mathsf{W}} h(\rho,\gamma,v,b) \\ & & |B(\rho,\mathsf{s}_i v,\vec{P}) \end{array}$$

$h(\rho,\gamma,\vec{P},v,b)$ is feasible because it is defined by bounded recursion. Let us check that indeed all components of a realiser of $\mathsf{T}(s \dot{\to} t)$ are given under certain addresses in $h(\alpha,\gamma,\vec{P},w,b)$ if we have

$$< \alpha,\gamma,\vec{P} > \ \mathfrak{R}_b\ \Gamma,\mathsf{Cl}(s),\mathsf{T}(s)\to\mathsf{T}(t) < \vec{c},\langle \ulcorner\mathsf{Cl}\urcorner,a_1,\cdots,a_{|w|}\rangle,\tilde{\gamma} >$$

The $\mathfrak{R}$ -realisation content that has to be delivered has the following form, where the b_i will be defined yet.

$$\langle \ulcorner\to\urcorner,\langle a_1,b_1\rangle,\cdots,\langle a_{|w|},b_{|w|}\rangle\rangle.$$

The realisation information for the a_i is already given at the addresses $b(\alpha,|\Gamma|+1)\!\downarrow\!.\mathsf{Cl}.v$ for $v\subseteq w$. Let us find the addresses at which the corresponding information for the b_i is given. Assume $v\subseteq w$. We do a case distinction depending on whether $a_{|v|}$ is a realiser of $\mathsf{T}(s)[\vec{t}]$ for some substitution $[\vec{t}]$. First, assume that this is the case. Assume $\mathsf{s}_i v'=v$. We abbreviate $h(\alpha,\gamma,\vec{P},v',b)/\mathsf{MA}(h(\alpha,\gamma,v',b))+1\to\alpha_{\mathsf{Cl}(s),v}$ as d. This means

$$d\ \ \mathfrak{r}_{\ b_\alpha \ddagger}\ \mathsf{T}(s),\Gamma,\mathsf{Cl}(s)[\vec{t}] < a_{|v|},\vec{c} > .$$

We abbreviate h without the bounds as h' and infer because of the properties of γ the following.

$$\mathsf{con}(h'(\alpha,\gamma,\vec{P},\mathsf{s}_i v',b),\mathsf{MA}(h'(\alpha,\gamma,\vec{P},\mathsf{s}_i v',b)))\equiv_{\mathsf{T}(t)[\vec{t}]}\tilde{\gamma}(a_{|v|},\vec{c})$$

This means that in this case $b_{|v|}$ has to be defined as $\tilde{\gamma}(a_{|v|},\vec{c})$. It has to be checked yet that the bounds in the definition of h are not active in this case which means that $h'(\alpha,\gamma,\vec{P},\mathsf{s}_i v',b)=h(\alpha,\gamma,\mathsf{s}_i v',b)$. This follows from $\mathsf{W}_{b_\alpha\ddagger}(h(\alpha,\gamma,\vec{P},v,b))=\mathsf{W}_b(\alpha)$ for all $v\in\mathbb{W}$ and the correctness of $\gamma,\vec{P}$.

As second case, we now assume that $a_{|v|}$ does not realise $\mathsf{T}(s)[\vec{t}]$ for any substitution $[\vec{t}]$. Since in this case we do not have to produce a correct realiser of $\mathsf{T}(t)[\vec{t}]$ the maximal address of $h(\alpha, \gamma, \vec{P}, v, b)$ works for an arbitrary defined $b_{|v|}$.

After having applied h, we have to add a new address under which $\mathsf{T}(s\dot{\to}t)$ will be realised. Its extensions will point at the relevant addresses that are all produced by h as we proved before. Accordingly, the realisation function $p(\wp, b)$ for $\wp :=< \rho, \gamma, \vec{P} >$ is calculated as follows:

- Produce a word w with $|w|$ being the number of left sides of the form $b(\rho, |\Gamma| + 1)\downarrow.\mathsf{Cl}.v$ in ρ.
- Calculate $h(\rho, \gamma, \vec{P}, w, b)$ and remember the relevant addresses $\breve{a}_1, \breve{b}_1, \cdots, \breve{a}_{|w|}, \breve{b}_{|w|}$ where $\breve{a}_i$ gives information for a_i and $\breve{b}_i$ for b_i.
- We add the following CD parts, where M is a new address:

$$M.\ulcorner\to\urcorner.0.0 \to \breve{a}_1/ \quad M.\ulcorner\to\urcorner.0.1 \to \breve{b}_1/ \cdots /M.\ulcorner\to\urcorner.n.0 \to \breve{a}_n/$$
$$M.\ulcorner\to\urcorner.n.1 \to \breve{b}_n$$

- We get the full realiser by adding the unchanged functional.

The projection $\tilde{p}$ is defined as follows where we assume that Γ contains m (positive) formulas.

$$\tilde{p}(\vec{c}, c_{m+1}, \tilde{\gamma}) := \begin{cases} \langle \vec{c}, c_{m+1}, \langle \ulcorner\to\urcorner, \langle a_0, \tilde{\gamma}(a_0)\rangle, \langle a_1, \tilde{\gamma}(a_1)\rangle, \cdots \rangle, \tilde{\gamma}\rangle, & \text{if } c_{m+1} \text{ is of the form} \\ & \langle \ulcorner\mathsf{Cl}\urcorner, a_0, a_1, \cdots \rangle \\ \langle \vec{c}, c_{m+1}, \epsilon, \tilde{\gamma}\rangle, & \text{else} \end{cases}$$

The arguments before imply that the first property is fulfilled for p,$\tilde{p}$. Property 2 is fulfilled because of the bounded definition of f. Property 3 is fulfilled for $p(\wp, b)_{P_i} = P_i$.

Note that because we cannot control the behaviour of the bounds in the case that possible realisers of $\mathsf{T}(s)$ are not actual, the realisation function and its projection produce $\mathfrak{R}$ -realisers that are only equal modulo $\equiv_{T(s\dot{\to}t)}$.

$\dot{\to}$ axiom left

Let the axiom have the following form.

$$\Gamma, \mathsf{Cl}(s), \mathsf{T}(s \dot{\to} t), \mathsf{T}(s) \Rightarrow \mathsf{T}(t)$$

From $\alpha\ \mathfrak{r}_b\ \Gamma, \mathsf{Cl}(s), \mathsf{T}(s \dot{\to} t), \mathsf{T}(s)$, we produce the required realiser as follows. We compare the coded classical realiser for $\mathsf{T}(s)$ with all possible realisers of $\mathsf{T}(s)$, given within the realisation information for $\mathsf{T}(s \dot{\to} t)$ (or $\mathsf{Cl}(s)$). There is a unique fitting address $b(\alpha, |\Gamma| + 2){\downarrow}.\ulcorner{\to}\urcorner.v.0$ because of lemma 50. Unlike the construction of $\mathfrak{R}$ -realisers their comparison is feasible as we will prove in lemma 65. Then add the CD part $\mathsf{MA}(\alpha) + 1 \to b(\alpha, |\Gamma| + 2){\downarrow}.\ulcorner{\to}\urcorner.v.1$. (For arbitrary inputs ρ, if no v can be found or ρ has not the intended form, just output ϵ.)

Stipulated that $\mathfrak{R}$ -realiser comparison is feasible, the above sketched functional is feasible too. It fulfils the properties 2,3 trivially. $\tilde{p}$ can be defined analogously, the correctness of p and $\tilde{p}$ is implied by lemma 50. The feasibility of p follows from the next lemma.

Lemma 65 *There is a feasible comparing function* $\mathsf{compare} : \mathbb{W}^4 \to \{0, 1\}$ *such that*

$$\mathsf{compare}(\rho_0, \rho_1, \breve{a}_0, \breve{a}_1) = 1 \Leftrightarrow \mathsf{con}(\rho_0, \breve{a}_0) = \mathsf{con}(\rho_1, \breve{a}_1).$$

Proof. We first describe the general strategy to compute compare, and enhance the efficiency of the computation in a second step to obtain feasibility. If any of the inputs is not of the intended form output 0. Else, to prove of disprove the equality of the realisers stored at addresses $\breve{a}_0$ and $\breve{a}_1$, we recursively compare realisers at smaller addresses $\breve{v}, \breve{w}$ such that $\mathsf{con}(\rho_0, \breve{v}) = \mathsf{con}(\rho_1, \breve{w})$ is implied by $\mathsf{con}(\rho_0, \breve{a}_0) = \mathsf{con}(\rho_1, \breve{a}_1)$. For two addresses $\breve{v}$ and $\breve{w}$ which have to be compared in one of these sub computations, we execute a case distinction on the case that is used first when calculating $\mathsf{con}(\rho_0, \breve{v})$ or $\mathsf{con}(\rho_0, \breve{v})$, respectively, relative to the definition by cases of con on page 71. If e.g. ρ_0 equals $1 : 0/2 \to 1$, to calculate $\mathsf{con}(\rho_0, 2)$ we first use case 1, for $\mathsf{con}(\rho_0, 1)$ we use case 6.

- The computation of $\mathsf{con}(\rho_0, \breve{v})$ and $\mathsf{con}(\rho_1, \breve{w})$ fall under different cases in the definition by cases of the function con and none of them falls under case 1: We end the whole computation with output 0.

- The computation of $\mathsf{con}(\rho_0, \breve{v})$ and $\mathsf{con}(\rho_1, \breve{w})$ both fall under cases 2,3,4,5 or one of them falls under case 1: We find new addresses, we have to compare. If e.g. both have $\ulcorner\wedge\urcorner$ extensions $\breve{v}.\ulcorner\wedge\urcorner.0$ and $\breve{v}.\ulcorner\wedge\urcorner.1$ or $\breve{w}.\ulcorner\wedge\urcorner.0$ and $\breve{w}.\ulcorner\wedge\urcorner.1$ respectively. We compare $\breve{v}.\ulcorner\wedge\urcorner.0$ with $\breve{w}.\ulcorner\wedge\urcorner.0$ and $\breve{v}.\ulcorner\wedge\urcorner.1$ with $\breve{w}.\ulcorner\wedge\urcorner.1$. It is also possible that e.g. $\breve{v}$ points at an address $\breve{v'}$ which means that this $\breve{v'}$ and $\breve{w}$ have to be compared.

- The computation of $\mathsf{con}(\rho, \breve{v})$ and $\mathsf{con}(\rho, \breve{w})$ both fall under case 6 or both fall under case 7: If both are of the form $\breve{l} : r$ or $\breve{l} : \langle\ulcorner\mathsf{T}\urcorner, r\rangle$ respectively, for equal r, we end this sub computation. If not, we end the whole computation with output 0.

- The computation of $\mathsf{con}(\rho, \breve{v})$ or $\mathsf{con}(\rho, \breve{w})$ falls under case 8: We end the whole computation with output 0.

Therefore by recursively using the comparing function, we will finally conclude $\mathsf{con}(\rho_0, \breve{a}_0) = \mathsf{con}(\rho_1, \breve{a}_1)$ and give output 1, if the computation terminates without giving output 0. The algorithm may execute the necessary sub computations in any order. To make the computation efficient enough, we have to remember which sub computations have been executed already. Assume that l_{ρ_0} is a left side of ρ_0 and l_{ρ_1} is a left side of ρ_1. We store all pairs $< l_{\rho_0}, l_{\rho_1} >$ for which the above displayed instructions have already been executed. The storage of these pairs allows to go through the above mentioned instructions only once for each pair $< l_{\rho_0}, l_{\rho_1} >$. Since these instructions and the required manipulations of the storage can be executed in polynomial time in ρ_0 and ρ_1 this establishes the claim.

$\square$

Axiom of choice

This axiom is given as follows for A positive.

$$\Gamma, (\forall x)(x \in \mathsf{W} \to (\exists y)(y \in \mathsf{W} \land \mathcal{T}(axy)) \Rightarrow$$
$$(\exists f)(\forall x)(x \in \mathsf{W} \to fx \in \mathsf{W} \land \mathcal{T}(ax(fx)))$$

As usual, we can realise the axiom of choice using twice the same functional realiser. Accordingly, we define the realisation functional as follows.

$$p(<\rho, \gamma, \vec{P}>, b) =<\rho, \gamma, \gamma, \vec{P}>$$

We define the projection $\tilde{p}$ as follows.

$$\tilde{p}(<\vec{c}, \tilde{\gamma}>) :=<\vec{c}, \tilde{\gamma}, \tilde{\gamma}>$$

We will give a suitable witness f for the existential quantifier in the succedent and prove the correctness of $p, \tilde{p}$ relative to it. Assume

$$\wp \ \mathfrak{R}_b \ \Gamma, (\forall x)(x \in \mathsf{W} \to (\exists y)(y \in \mathsf{W} \land \mathcal{T}(axy))) <\vec{c}, \tilde{\gamma}>$$

From now on, we write MA instead of $\mathsf{MA}(\rho)$, if MA occurs in the second argument of con or con_W with ρ as first argument. We define f as the closed term which corresponds to the following function from $\mathbb{W}$ to $\mathbb{W}$

$$\lambda x.\mathsf{con}_\mathsf{W}(\gamma(\wp^+/\mathsf{MA}(\wp) + 1 : x, b_{\wp^+}\ddagger), \mathsf{MA}\downarrow.0).$$

We find such a term because γ, b are recursive. According to the logical structure of the main claim the dependence of the witness on $\wp, b$ is unproblematic. Now, we prove

$$<\gamma, \vec{P}> \mathfrak{r}'(\exists f)(\forall x)(x \in \mathsf{W} \to fx \in \mathsf{W} \land \mathcal{T}(ax(fx))),$$

relative to $\Gamma, <\vec{c}, \tilde{\gamma}>$ for the witness given above.
Assume for a $\rho \in \mathbb{W}$, an address finder b', and a fresh variable u

$$\rho \ r_{b'} \ u \in \mathsf{W}, \Gamma[\vec{s}] < w, \vec{c}> .$$

$w_0 \equiv_{s \in \mathsf{W}} w_1$ implies $w_0 = w_1$ for any $w_0, w_1 \in \mathbb{W}$ and any term s. This implies that $value(u[\vec{s}]) = w \in \mathbb{W}$ [10]. The properties of γ, $\tilde{\gamma}$ imply

[10] For readability, we ignore in the following the difference between w and $\langle \ulcorner \mathsf{W} \urcorner, w \rangle$.

$$\mathsf{con}(\gamma(\rho, b'), \mathsf{MA}) \equiv_{(\exists y)(y \in \mathsf{W} \wedge \mathcal{T}(au[\vec{s}]y))[\vec{s}]} \tilde{\gamma}(w)$$

We use the properties of $\equiv$ for existential quantification to derive

(A) $$\mathsf{con}(\gamma(\rho, b'), \mathsf{MA}) \equiv_{t \in \mathsf{W} \wedge \mathcal{T}(au[\vec{s}]t))[\vec{s}]} \tilde{\gamma}(w)$$

for some term t. We use the properties of $\equiv$ for $\wedge$ and of $\downarrow$ to derive

$$\mathsf{con}(\gamma(\rho, b'), \mathsf{MA}{\downarrow}.0) \equiv_{t \in \mathsf{W}[\vec{s}]} \tilde{\gamma}(w)_1$$

for the same term t. This implies

$$\mathsf{con}_{\mathsf{W}}(\gamma(\rho, b'), \mathsf{MA}{\downarrow}.0) = \tilde{\gamma}(w)_{1,1}$$

In particular, if we choose ρ as $\wp^+/\mathsf{MA}(\wp) + 1 : w$ and b' as $b_{\wp^+}\ddagger$, we derive

$$value(fu[\vec{s}]) = \tilde{\gamma}(w)_{1,1}$$

So, because of the properties of the realisation relation, we have that the above mentioned t equals $fu[\vec{s}]$ in the standard model, which yields because of (A) the desired result. It follows immediately that properties 1,2,3 are fulfilled.

Note that in order to show that the witness chosen for $(\exists y)(y \in \mathsf{W} \wedge \mathcal{T}(au[\vec{s}]y))$ does not depend on the particular representation of the $\mathfrak{R}$ - realisation information in the address-pointer formalism but only on the value of $u[\vec{s}]$, we make crucial use of the projection formalism.

3.4.3 Rules of T^+

First, we introduce some for our approach specific problems we will encounter in the realisation of the rules and give rough instructions how to deal with them. In the following, we look at rules with premisses Δ_i and conclusion Σ. We abbreviate the positive formulas of the antecedent of the sequent Σ as Σ^+, analogously for the Δ_i.

Problem 1: If a Δ_i^+ is logically weaker than Σ^+, we cannot apply the realisation function of this premise to the given realiser, because its functional components are not strong enough. We avoid this problem by using a special formulation of the rules and by avoiding weakening.

Problem 2: If a Δ_i^+ is stronger than Σ^+, the functional realisers produced by the realisation function of this premise are not strong enough to be functional realisers of the conclusion. This is the case for the rules $\vee$-left, cut, the $\rightarrow$-rules, existential quantifier left. This problem will be solved usually by combining the weak functional realisers the realisation function of one of the premisses produces with functions that deliver a realiser of one of the Δ_i^+ from a realiser of Σ^+.

Problem 3: For the structural rules Δ^+ and Σ^+ are logically equivalent, nevertheless, similar difficulties as described above occur because the functional realisers for Δ^+ expect information for a certain formula at an other address than the ones of Σ^+. The problem can be solved by address finder modifications.

Problem 2 only occurs if the succedent is a negative formula. Else, the functional realisers for Σ^+ can just be preserved and the weakened ones, produced by the premise realisation function, ignored. Let us first treat rules where problems 2 and 3 do not occur.

$\rightarrow$ rule right

Let the applied rule have the following form.

$$\frac{\Gamma, A \Rightarrow D}{\Gamma \Rightarrow A \rightarrow D}$$

We assume that Γ has m positive formulas. We can assume that A, D are positive. By induction hypothesis we have a realisation functional p for the premise, which almost gives the functional realiser of $A \rightarrow D$, we just have to modify the address finder.

Definition 66 *For an address finder b with $n+1$ relevant inputs, we define $b\dagger$ as follows.*

$$b\dagger(\rho, i) = \begin{cases} b(\rho, i+1), & \textit{if } 1 \leq i \leq n \\ b(\rho, 1), & \textit{if } i = n+1 \end{cases}$$

We define the realisation function f as follows.

$$\begin{aligned} f(\wp, b) &= < \wp^+, \vec{\gamma}, \lambda w.\lambda c.p(< w, \vec{\gamma}, \vec{P} >, c\dagger)^+, \\ &\qquad \lambda w.\kappa_p(w, P_1, P_3), \lambda w.\lambda v.\delta_p(w, v, \vec{P}), \lambda w.\gamma_p(w, P_3) > \end{aligned}$$

We define $\tilde{f}$ as follows.

$$\tilde{f}(\vec{c}, \vec{\tilde{\gamma}}) = < \vec{c}, \vec{\tilde{\gamma}}, \lambda v.\tilde{p}_D(\vec{c}, v, \vec{\tilde{\gamma}}) >$$

To prove that properties 1,2,3 are fulfilled for f and $\tilde{f}$ it suffices to prove the correctness of the added functions. We assume $\wp\ \mathfrak{R}_b\ \Gamma[\vec{s}] < \vec{c}, \vec{\tilde{\gamma}} >$.

Take a $\rho \in \mathbb{W}$ with $\rho\ \mathfrak{r}_{b'}\ A, \Gamma^+[\vec{s}][\vec{t}] < v, \vec{c} >$. This implies $\rho\ \mathfrak{r}_{b'\dagger}\ \Gamma^+, A[\vec{s}][\vec{t}] < \vec{c}, v >$. Because the functions $\vec{\gamma}$ will ignore the information for the formula $A[\vec{s}][\vec{t}]$, we have

$$< \rho, \wp^- > \quad \mathfrak{r}_{b'\dagger}\ \Gamma, A[\vec{s}][\vec{t}] < \vec{c}, v, \vec{\tilde{\gamma}} >$$

For each formula A, we write $\tilde{p}_A$ for the projection of $\tilde{p}$ producing an $\mathfrak{R}$-realiser of A. This yields

$$p(< \rho, \wp^- >, b'\dagger)^+ \quad \mathfrak{r}_{b'_\rho}\ A, \Gamma^+, D[\vec{s}][\vec{t}] < v, \vec{c}, \tilde{p}_D(< \vec{c}, v, \vec{\tilde{\gamma}} >) >$$

So the added functional realiser is correct. The new bounds work for the functional realisers of $\wp$ because of the monotonicity of κ, δ, γ. They additionally work for the added functional because the functional parameters occurring in $\lambda w.\lambda c.p(< w, \vec{\gamma}, \vec{P} >, c\dagger)^+$ extend the input to a realiser of $\Gamma, A[\vec{s}][\vec{t}]$ and for such inputs the bounds hold due to the induction hypothesis.

Induction

We have to realise the extended induction rule of the following form.

$$\frac{\Gamma, t \in \mathsf{W} \Rightarrow \mathsf{T}(r\epsilon) \vee C \qquad \Gamma, t \in \mathsf{W}, \mathsf{T}(rx) \vee C, x \in \mathsf{W} \Rightarrow \mathsf{T}(r(\mathsf{s}_i x)) \vee C}{\Gamma, t \in \mathsf{W} \Rightarrow \mathsf{T}(rt) \vee C},$$

where C is a positive formula. By induction hypothesis we have realisation functionals p, q_0 and q_1 for the premisses. We can find the realisation functional f for the conclusion in a very similar way as in chapter 2. This is so, because the functional realisers, possibly present in the given realiser of $\Gamma, t \in \mathsf{W}[\vec{s}]$, have not to be changed during the recursion, they only behave as parameters. So, we can concentrate on a realiser for the positive part. The use of functionals even allows are more elegant definition than in the mentioned chapter: We include the address finder b as third argument, the first two arguments of f will behave as for the original f. f is given by a recursion that we bound using the functions $\vec{P}$. The bound is inactive if the inputs are as intended. Now, we define bounding functions for the addresses and the length of the realiser.

$$h(\wp, w, b) \;:=\; \kappa_p(\mathsf{W}_b(\wp), P_1, P_3) + \kappa_q(\mathsf{W}_b(\wp), P_1, P_3) \times w + w + 1,$$

where κ_q is a bound for $\kappa_{q_0}, \kappa_{q_1}$. Again, we define $g : \mathbb{W} \to \mathbb{W}$ as the feasible function which applied to $w \in \mathbb{W}$ gives a bound for the CD parts not build from words longer than w (excluding $\ulcorner\mathsf{Cl}\urcorner, \ulcorner\wedge\urcorner, \cdots$).

Definition 67 *The bounding function B is defined by case distinction as follows.*

$$\begin{aligned}
B(\wp, \epsilon, b) &:= \wp^+ + \delta_p(\mathsf{W}_b(\wp), \mathsf{MA}(\wp), \vec{P}) + g(\mathsf{MA}(\wp) + h(\wp, \epsilon, b)) \\
B(\wp, \mathsf{s}_i w, b) &:= \wp^+ + \delta_p(\mathsf{W}_b(\wp), \mathsf{MA}(\wp), \vec{P}) + \\
&\quad \big[\delta_q(\mathsf{W}_b(\wp), \mathsf{MA}(\wp) + h(\wp, w, b), \vec{P}) + \\
&\quad g(\mathsf{W}_b(\rho)) + \mathsf{MA}(\wp) + h(\wp, \mathsf{s}_i w, b)\big] \times \mathsf{s}_i w
\end{aligned}$$

Definition 68 *The function $\tilde{b}_w$ is given by the following definition of cases.*

$$\tilde{b}_w(\rho, i) = \begin{cases} b(w, i), & \text{if } 1 \le i \le |\Gamma| + 1 \\ \mathsf{MA}(\rho) - 1, & \text{if } i = |\Gamma| + 2 \\ \mathsf{MA}(\rho), & \text{if } i = |\Gamma| + 3 \end{cases}$$

Definition 69 *The function f' is defined by bounded recursion as follows. We abbreviate $q_i(< f(\wp, w, b), \vec{\gamma}, \vec{P} >, \tilde{b}_{\wp^+})^+$ as $c(\wp, w, b)$. We abbreviate $\mathsf{MA}(f(\wp, w, b)) - 1\!\downarrow$ as v.*

$$f'(\wp, \epsilon, b) \;:=\; p(\wp, b)^+/\mathsf{MA}(p(\wp, b)^+) + 1 : \epsilon | B(\wp, \epsilon, b)$$

$$f'(\wp, \mathsf{s}_i w, b) \;:=\; \begin{cases} f'(\wp, w, b), & \text{if } R_{\wp^+}(v, v.1) \\ c(\wp, w, b)/\mathsf{MA}\Big(c(\wp, w, b)\Big) + 1 : \mathsf{s}_i w | B(\wp, \mathsf{s}_i w, b), & \text{else} \end{cases}$$

The projection $\tilde{f}'(\vec{c}, w, \vec{\tilde{\gamma}})$ of f' is defined analogously but by unbounded recursion.

The correctness of the functions $f', \tilde{f}'$ can be proved very similarly as in chapter 2 for T_{PT}: We assume $\wp \ \mathfrak{R}_b \ \Gamma, t \in \mathsf{W}[\vec{s}] < \vec{c}, \vec{\tilde{\gamma}} >$. We denote by v the minimal word $v' \subseteq value(t[\vec{s}])$ such that

$$R(\mathsf{MA}(f(\wp, v, b)) - 1\downarrow, \mathsf{MA}(f(\wp, v, b)) - 1\downarrow.1),$$

if such a word exists, and else $value(t[\vec{s}]) * 0$.
Statements analogous to lemmas 26,27 in the previous chapter can be proved for a function f defined as f' but without bound B. For the analogon of lemma 26, we prove by induction on w for all $w \subset v$

$$f(\wp, w, b) \ \mathfrak{r}_{\tilde{b}_{\wp^+}} \Gamma^+, t \in W, \mathsf{T}(r\overline{w}), \overline{w} \in \mathsf{W}[\vec{s}]\tilde{f}(\vec{c}, \vec{\tilde{\gamma}}, w)$$

For the induction step we use that the $\mathfrak{r}_{\tilde{b}_{\wp^+}}$ realiser produced by f can be completed to a $\mathfrak{R}_{\tilde{b}_{\wp^+}}$ realiser adding $\wp^-$. Lemma 27 holds in a restricted form exactly for the same reasons as before: For each $w \subset v$, we have

$$\mathsf{W}_{\tilde{b}}(f_b(\wp, w, b)) \leq \mathsf{W}_b(\wp).$$

Now, it can be proved easily that the bounding function B in the definition of f' is inactive under the assumption that $\wp$ is a realiser relative to b, which implies their correctness. The realisation function and its projection can be defined easily from f', $\tilde{f}'$. We can add the unmodified functional realisers. Property 1 follows from the arguments given before. 2.1 follows because of the analogon of lemma 27, 2.2 follows because of the bounded definition of f'. 2.3 follows for a feasible bound γ of γ_p, γ_q because we increase the computable content at most once: in the step we first realise the disjunct C. Property 3 follows with unchanged $\vec{P}$. Note that compared with the treatment of the induction rule in chapter 2, we do not need a reverse of the auxiliary realisation function. This is because we are working with realisation functionals with the address finder as argument. Therefore, we are allowed to let the address finder depend on $\wp^+$ for the realiser $\wp$ of $\Gamma, t \in \mathsf{W}[\vec{s}]$.

$\rightarrow$ rule left

Let the applied rule have the following form.

$$\frac{\Gamma \Rightarrow A \qquad\qquad\qquad\qquad \Gamma, B \Rightarrow D}{\Gamma, A \rightarrow B \Rightarrow D}$$

We can assume that A, D are positive. By induction hypothesis we have the realisation functionals p and q for the premisses. We have to apply both of them and additionally the realiser of $A \rightarrow B$.

Definition 70 *For $\wp =:< \rho, \gamma_1, \cdots, \gamma_n, \vec{P} >$ we write $\wp^{\boxminus}$ for $< \rho, \gamma_1, \cdots, \gamma_{n-1}, \vec{P} >$.*

We define an approximation f' of the realisation functional as follows.

$$f'(\wp, b) = q(< \gamma_{A\rightarrow B}[p(\wp^{\boxminus}, b)^+, b_{\wp^+}\ddagger], (\wp^{\boxminus})^- >, b_{\wp^+})$$

We show that f' delivers realisation information for the positive formulas, and explain afterwards how it has to be modified to treat the negative formulas. So, let us assume

$$\wp \ \mathfrak{R}_b \ \Gamma, A \rightarrow B[\vec{s}] < \vec{c}, \vec{\tilde{\gamma}}, \tilde{\gamma}_{A\rightarrow B} > .$$

This implies

$$\wp^{\boxminus} \ \mathfrak{R}_b \ \Gamma[\vec{s}] < \vec{c}, \vec{\tilde{\gamma}} > .$$

The induction hypothesis for p delivers

$$p(\wp^{\boxminus}, b)^+ \ \ \mathfrak{r}_{\ b_{\wp^+}} \ \Gamma^+, A[\vec{s}] < \vec{c}, \tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}}) > .$$

(Remember that $\tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}})$ gives the component of the realisation information given in $\tilde{p}(\vec{c}, \vec{\tilde{\gamma}})$ that is responsible for the formula A.) This implies

$$p(\wp^{\boxminus}, b)^+ \ \ \mathfrak{r}_{\ b_{\wp^+}\ddagger} \ A, \Gamma^+[\vec{s}] < \tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}}), \vec{c} > .$$

We apply $\gamma_{A\rightarrow B}$, forget the realiser for A and get the following.

$$c := \gamma_{A\rightarrow B}[p(\wp^{\boxminus}, b)^+, b_{\wp^+}\ddagger] \ \ \mathfrak{r}_{\ b_{\wp^+}} \ \Gamma^+, B[\vec{s}] < \vec{c}, \tilde{\gamma}_{A\rightarrow B}(\tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}})) > .$$

The induction hypothesis for q implies that

$$\text{(A)} \qquad q(\gamma_{A\to B}[p(\wp^{\boxminus}, b)^+, b_{\wp^+}\ddagger], (\wp^{\boxminus})^-), b_{\wp^+}) \ \mathfrak{R}\ {}_{b_{\wp^+},c} \\ \Gamma, B, D[\vec{s}]\ \tilde{q}(\vec{c}, \tilde{\gamma}_{A\to B}(\tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}})), \vec{\tilde{\gamma}})$$

Now, by forgetting the information for $B[\vec{s}]$ we get an $\mathfrak{r}$ -realiser of $(\Gamma, D)^+$ relative to a projection as above described but with the information for B is erased. The only problem is that we have possibly weakened the functional realisers as they expect an input containing an $\mathfrak{R}$ -realiser w with $w \equiv_{B[\vec{s}]} \tilde{\gamma}_{A\to B}(\tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}}))$. First, let us assume that the negative formula NF_i that is to be realised is of the form $\ell NF_i \to rNF_i$ for $\ell NF_i, rNF_i$ positive.

The searched functional realiser for this formula can be described in the following way: We take the input which is a realiser of $\ell NF_i, \Gamma[\vec{s}][\vec{t}]$ (in the intended case) and produce from it, similarly as demonstrated before, a realiser of $\ell NF_i, \Gamma, B[\vec{s}][\vec{t}]$. Then, we apply the functional realiser $f'(\wp, b)_{NF_i}$ to this modified input. We change the bounds accordingly. For more complicated NF_i, the same strategy works. Accordingly, f is given by the following algorithm to compute $f(\wp, b)$.

- Compute $f'(\wp, b)$.
- Replace the functional components $f'(\wp, b)_{NF_i}$ by

$$\lambda x.\lambda y.f'(\wp, b)_{NF_i}(\gamma_{A\to B}[p(< x, (\wp^{\boxminus})^- >, y\dagger)^+, y\dagger_x\ddagger], y_x)$$

- Replace the bound $f'(\wp, b)_{P_1}$ by

$$\lambda x.f'(\wp, b)_{P_1}[\wp_{P_3}(\gamma_p(x, \wp_{P_3}))] + \wp_{P_1}(\gamma_p(x, \wp_{P_3})) + \kappa_p(x, \wp_{P_1}, \wp_{P_3})$$

- Replace the bound $f'(\wp, b)_{P_2}$ by

$$\lambda x.\lambda y.f'(\wp, b)_{P_2}[\wp_{P_3}(\gamma_p(x, \wp_{P_3})), y + \wp_{P_1}(\gamma_p(x, \wp_{P_3})) + \\ \kappa_p(x, \wp_{P_1}, \wp_{P_3})] + \wp_{P_2}(\gamma_p(x, \wp_{P_3}), y + \kappa_p(x, \wp_{P_1}, \wp_{P_3})) + \delta_p(x, y, \wp_{\vec{P}})$$

- Replace the bound $f'(\wp, b)_{P_3}$ by

$$\lambda x.f'(\wp, b)_{P_3}(\wp_{P_3}(\gamma_p(x, \wp_{P_3})))$$

- Add the removed functional realiser $\gamma_{A \to B}$.

To get the right projections of the functional components, we replace the functional components $\tilde{f}'(\vec{c}, \vec{\tilde{\gamma}})_{NF_i}$ of the projection
$\tilde{f}'(\vec{c}, \vec{\tilde{\gamma}}) := \tilde{q}(\vec{c}, \tilde{\gamma}_{A \to B}(\tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}})))$ by

$$\lambda v. \tilde{f}'(\vec{c}, \vec{\tilde{\gamma}})_{NF_i}(v, \vec{c}, \gamma_{A \tilde{\to} B}(\tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}})))$$

After these modifications, we delete the content for B and add the functional component for $A \to B$. This delivers the projection $\tilde{f}$ of the realisation function f.

Now, we will prove the correctness of the realisation function and its projection. Let us prove property 1 first. Claim (A) and the arguments above imply

$$f(\wp, b)^+ \ \mathfrak{r}_{\ b_{\wp^+}} \ (\Gamma, D)^+ \tilde{f}(\vec{c}, \vec{\tilde{\gamma}})^+.$$

So, we only have to show that the functional components of the realiser are correct. We use induction on the number of quantifiers having $\to$ in their scope. Assume for a $\rho \in \mathbb{W}$ and a formula $A_0 \to A_1$ with A_0, A_1 positive

$$\rho \ \ \mathfrak{r}_{\ b'} \ A_0, \Gamma^+[\vec{s}][\vec{t}] < v, \vec{c} > .$$

This implies

$$< \rho, (\wp^{\ominus})^- > \ \ \mathfrak{r}_{\ b' \dagger} \ \Gamma[\vec{s}][\vec{t}] < \vec{c}, \vec{\tilde{\gamma}} >,$$

where we ignore the information for $A_0[\vec{s}][\vec{t}]$. In the same way as before, we derive

$$\gamma_{A \to B}(p(< \rho, (\wp^{\ominus})^- >, b'\dagger)^+, b'\dagger_\rho \ddagger) \ \mathfrak{r}_{\ b'_\rho} \ A_0, \Gamma^+, B[\vec{s}][\vec{t}] < v, \vec{c}, \tilde{\gamma}_{A \to B}(\tilde{p}_A(\vec{c}, \vec{\tilde{\gamma}})) >$$

But on this CD, $\tilde{f}'(\wp, b)_{A_0 \to A_1}$ works correctly, which implies $f(\wp, b)_{\tilde{\gamma}_i}$

$$f(\wp, b)_{\tilde{\gamma}_i}(\rho, b') \ \ \mathfrak{r}_{\ b'_\rho} \ A_0, \Gamma^+, A_1[\vec{s}][\vec{t}] < v, \vec{c}, \tilde{f}(\wp, b)_{\tilde{\gamma}_i}(v, \vec{c}) >$$

as required. The correctness of the bounds follows easily form the induction properties for the involved functionals. The induction step for negative formulas of higher complexity is trivial.

The bounds for property 2 are produced by replacing in the bounds for the functional realisers the terms $f'(\wp, b)_{P_i}$ by $\kappa_q, \delta_q, \gamma_q$, respectively. Property 3 is clearly fulfilled.

Existential quantifier rule left

Let this rule be formulated as follows where the usual variable condition applies.

$$\frac{\Gamma, A[u] \Rightarrow D}{\Gamma, (\exists x)A[x] \Rightarrow D}$$

Take the realisation function p of the premise if A is negative, so assume that A is positive. Against expectation the realisation is not trivial. We assume $\wp \ \mathfrak{R}\ _b\Gamma, (\exists x)A[x][\vec{s}] < c_1, \cdots, c_{|\Gamma^+|+1}, \vec{\tilde{\gamma}} >$. Let p be the realisation function of the premise. Its application to $\wp$ yields correct positive realisers because we can apply the induction hypothesis for a suitable substitution of u by a closed term t. But we produce functional realisers which are very weak because they require inputs ρ such that for an address finder b'

$$\mathsf{con}(\rho, b'(\rho, |\Gamma^+| + 2)) \equiv_{A(t)[\vec{s}]} c_{|\Gamma^+|+1},$$

where t is a *specific* substitution of u determined by $\wp^+$. Although, we need a functional realiser that works for inputs ρ with

$$\mathsf{con}(\rho, b'(\rho, |\Gamma^+| + 2)) \equiv_{(\exists x)A(x)[\vec{s}]} c_{|\Gamma^+|+1}.$$

Therefore, we have to replace $p(\wp, b)_{NF_i}$ by γ'_i defined as follows.

$$\gamma'_i(x, y) := p(< x, \wp^- >, y\dagger)_{NF_i}(x, y)$$

The $p(\wp, b)_{P_i}$ do not have to be modified, since p is applied to a realiser with the same negative part as $\wp$.
We replace the projections of $\tilde{p}(\wp, b)_{NF_i}$ analogously:

$$\tilde{\gamma}_i{}'(v) := \tilde{p}(< \vec{c}, \vec{\tilde{\gamma}} >)_{NF_i}(v)$$

Let us prove the correctness of the modified functional realisers γ_i, where we still assume $\wp \ \mathfrak{R}\ _b\Gamma, (\exists x)A[x][\vec{s}] < c_1, \cdots, c_{|\Gamma^+|+1}, \vec{\tilde{\gamma}} >$. We let NF_i be of the form $\ell NF_i \to rNF_i$ for $\ell NF_i, rNF_i$ positive. The other cases can again be treated by an easy induction. So assume

$$\rho \ \ \mathfrak{r}\ _{b'}\ \ell NF_i, \Gamma^+, (\exists x)A[x][\vec{s}][\vec{t}] < v, \vec{c} >$$

This implies

$$< \rho, \wp^- > \ \mathfrak{R}_{b'\dagger} \ \Gamma, (\exists x) A[x][\vec{s}][\vec{t}] < \vec{c}, \vec{\tilde{\gamma}} >$$

Because of lemma 60

$$< \rho, \wp^- > \ \mathfrak{R}_{b'\dagger} \ \Gamma, A[t][\vec{s}][\vec{t}] < \vec{c}, \vec{\tilde{\gamma}} >$$

for a specific closed term t depending on the realiser ρ. This means that $p(< \rho, \wp^- >, b'\dagger)$ produces functional components that are correct on inputs x with address finder y with

$$\mathsf{con}(x, y(x, 1)) \ \mathfrak{R} \ \ell NF_i[\vec{s}][\vec{t}][\vec{u}]$$

and

$$\mathsf{con}(x, y(x, i+1)) \equiv_{\Gamma_i^+[\vec{s}][\vec{t}][\vec{u}]} c_i$$

for $1 \leq i \leq |\Gamma|$ and

$$\mathsf{con}(x, y(x, |\Gamma^+| + 1)) \equiv_{A[t][\vec{s}][\vec{t}][\vec{u}]} c_{|\Gamma|+1}.$$

For $[\vec{u}]$ being the identity substitution, we derive

$$\gamma_i'(\rho, b') \ \ \mathfrak{r}_{\ b'_\rho} \ \ell NF_i[\vec{s}][\vec{t}], \Gamma^+[\vec{s}][\vec{t}], A[t][\vec{s}][\vec{t}], rNF_i[\vec{s}][\vec{t}] < v, \vec{c}, \tilde{\gamma}_i{}'(v) > .$$

We derive

$$\gamma_i(\rho, b') \ \ \mathfrak{r}_{\ b'_\rho} \ \ell NF_i[\vec{s}][\vec{t}], \Gamma^+[\vec{s}][\vec{t}], (\exists x) A[x][\vec{s}][\vec{t}], rNF_i[\vec{s}][\vec{t}] < v, \vec{c}, \tilde{\gamma}_i(v) > .$$

So, we produce correct realisers for the negative formulas. For the realisation of the positive formulas, we can just use the premise realisation function p. It is easy to see that properties 1,2, and 3 are fulfilled.

For the realisation of the $\vee$ left rule, one applies very similar ideas.

Structural rules

If the particular structural rule has a negative main formula, it can be realised very easily. Additionally, weakening with positive formulas was excluded. Therefore, we will just demonstrate contraction and commutation for positive main formulas. These rules require us to modify the address finders being the second argument of functional realisers.

Let the applied contraction rule have the following form.

$$\frac{\Gamma, A, A \Rightarrow D}{\Gamma, A \Rightarrow D}$$

Let p be the realisation function of the premise. Define an approximation f' of the realisation function as follows.

$$f'(\wp, b) := p(\wp, b'),$$

where

$$b'(\rho, i) = \begin{cases} b(\rho, i), & \text{if } 1 \leq i \leq |\Gamma^+| + 1 \\ b(\rho, |\Gamma^+| + 1), & \text{if } i = |\Gamma^+| + 2 \end{cases}$$

This produces almost the right realiser, we just have to adjust the arity of the functional realisers. So the realisation function $f(\rho, b)$ is defined as $shrink(f'(\rho, b))$ were $shrink$ replaces each $f'(\wp, b)_{NF_i}$ by

$$\lambda x.\lambda y.f'(\wp, b)_{NF_i}(x, \hat{y}),$$

with $\hat{y}$ defined as follows.

$$\hat{y}(\rho, i) = \begin{cases} y(\rho, i), & \text{if } 1 \leq i \leq |\Gamma^+| + 2 \\ y(\rho, |\Gamma^+| + 2), & \text{if } i = |\Gamma^+| + 3 \end{cases}$$

We define $\tilde{f}'$ as follows.

$$\tilde{f}'(c_1, \cdots, c_{|\Gamma^+|+1}, \vec{\tilde{\gamma}}) := \tilde{p}(c_1, \cdots, c_{|\Gamma^+|+1}, c_{|\Gamma^+|+1}, \vec{\tilde{\gamma}})$$

Let the applied commutation rule have the following form.

$$\frac{\Gamma, A, B, \Delta \Rightarrow D}{\Gamma, B, A, \Delta \Rightarrow D}$$

Define the function f' as follows.

$$f'(\wp, b) := p(< \rho, \lambda x.\lambda y.\gamma_1(x, \hat{y}), \cdots, \lambda x.\lambda y.\gamma_n(x, \hat{y}), \vec{P} >, b'),$$

where $\hat{}$ switches the values of b at positions $|\Gamma^+|+2$ and $|\Gamma^+|+3$ and $'$ switches the values of b at positions $|\Gamma^+| + 1$ and $|\Gamma^+| + 2$. This almost produces the required realiser only the function components have to be adjusted. So the realisation function $f(\rho, b)$ is defined as $switch(f'(\rho, b))$ were $switch$ replaces each $f'(\wp, b)_{NF_i}$ by

$$\lambda x.\lambda y.f'(\wp, b)_{NF_i}(x, \hat{y}).$$

We define the projection $\tilde{f}$ as follows.

$$\tilde{f}(c_1, \cdots, c_{|\Gamma^+|+1}, c_{|\Gamma^+|+2}, \vec{\tilde{\gamma}}) := \tilde{p}(c_1, \cdots, c_{|\Gamma^+|}, c_{|\Gamma^+|+2}, c_{|\Gamma^+|+1}, \vec{\tilde{\gamma}})$$

Universal quantifier rule left

Let us look at the universal quantifier rule.

$$\frac{\Gamma, (\forall x)A[x], A[t] \Rightarrow D}{\Gamma, (\forall x)A[x] \Rightarrow D}$$

The realisation function can be found trivially if A is negative. So assume that A is positive. Let p be the realisation function of the premise. Because of lemma 60, the following realisation function yields correct *positive* realisers.

$$f(\wp, b) := p(\wp, b'),$$

where b' finds the $|\Gamma|+1$-th realiser relative to b twice. The functional realisers have to be modified as for contraction. We define $\tilde{f}$ as for contraction.

Cut

Let the applied rule have the following form.

$$\frac{\Gamma \Rightarrow A \qquad\qquad\qquad\qquad \Gamma, A \Rightarrow D}{\Gamma \Rightarrow D}$$

Assume first that the cut formula A is positive. Then we apply both premise realisation functions consecutively as usual for cut. We have to modify the weakened functional components analogously as for the $\rightarrow$ left rule. The projection can be defined analogously. In case that A is negative, we define the realisation functional as expected and easily prove that property 1 holds. To prove that the property 2 is fulfilled for the realisation function f, we use the induction hypothesis for property 3. Property 3 holds because of the closedness of feasible functionals under composition.

The rules, we did not treat here, can be realised using similar ideas. This finishes the proof of the main lemma.

Because the feasible functionals from $\mathbb{W}$ to $\mathbb{W}$ are exactly the polytime functions, we deduce the following corollary as in chapter 2. Note that for the construction of the polytime function corresponding to a provable total function, we do not rely on the projection formalism, which guarantees the feasibility.

Corollary 71 (of theorem 61 and lemma 43) *The provably total functions of the following theories are exactly the polytime functions.*

- $\mathsf{T}_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP}$
- $\mathsf{T}^{i}_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP} + \mathsf{RN}$

3.5 Induction over negative formulas for weak applicative theories

As we claimed at the beginning of this chapter, the technique of treating negative induction formulas by realisers coding them as finite functions can easily be adapted to further weak applicative theories. In the following, we sketch how intuitionistic versions of Strahm's theories PT, PTLS, PS, and LS, introduced in [79], can be conservatively extended by induction over negative formulas.

Definition 72 *Let* T *be one of the theories* PT^i, PTLS^i, PS^i, *or* LS^i. *Then, the theory* T' *extends* T *by the following induction scheme, where in* A, *no* W *occurs in the antecedent of any implication.*

$$u \in \mathsf{W} \rightarrow (A^u[\epsilon] \wedge (\forall x \in \mathsf{W})(A^u[x] \rightarrow A^u[\mathsf{s}_i x]) \rightarrow (\forall x \in \mathsf{W}) A^u[x])$$

The upper bound of the extended theories T' is found by a realisation approach for quasi-cut-free proofs, formulated in sequent calculus. We have to explain how to realise arbitrary formulas without occurrence of W in the antecedent of any implication. Such formulas are called normal from now on. For W free formulas, we define the set of possible realisers as for classes of the theory T^+. Let A be a W free formula, and B a normal formula, then the word ρ is a realiser of $A \rightarrow B$ exactly if the following conditions are fulfilled.

- ρ codes a finite function f whose inputs are the possible realisers of A.
- For any substitution $[\vec{s}]$ and any word ρ' we have

$$\rho' \ \mathfrak{R} \ A[\vec{s}] \Rightarrow f(\rho') \ \mathfrak{R} \ B[\vec{s}]$$

If we choose a sensible coding of the finite functions, the realisers of the induction formulas A^t can still be bounded linearly in $value(t)$ which allows the realisation of induction by bounded recursion. The $\rightarrow$ rule right is realised by applying the realisation function of the premise to all possible realisers of the antecedent of the main formula. The $\rightarrow$ rule right is realised by applying consecutively the premise realisation functions and the coded finite function.

Also the theory PETJ^i, introduced and analysed by Spescha and Strahm in [74], can be conservatively extended by allowing induction over negative formulas. (PETJ^i is presented in detail in the next chapter.) This is done using a second name predicate $\mathfrak{S}$ corresponding to classes of T^+, and an additional constant imp fulfilling the following axioms.

$$\mathfrak{S}(a) \wedge \mathfrak{R}(b) \rightarrow \mathfrak{R}(\mathsf{imp}(a,b)) \wedge (\forall x)(x \in \mathsf{imp}(a,b) \leftrightarrow x \in a \rightarrow x \in b)$$

An upper bound by an embedding into T^+ can be found by standard techniques along the lines of the next chapter. Note that this does not deliver a feasible theory of explicit mathematics that reflects all *normal* formulas but only such formulas where all antecedents are positive and W free. For the realisation of the more general theory of explicit mathematics the problem described for T^+ on page 83 occurs.

3.6 Open questions

Cantini's reduction technique of classical - to the corresponding intuitionistic theories developed in [18] clearly does not work for an axiom of choice for negative formulas A. Nevertheless, in the cited paper he succeeds to realise an intuitionistic theory with full axiom of choice, and proves that this axiom does not strengthen his theory with respect to its provably total functions. We strongly assume that our system $\mathsf{T}^i_{\mathsf{PT}} + \mathsf{Pos} - \mathsf{AC} + \mathsf{UP} + \mathsf{RN}$ extended by the full axiom of choice is still feasible. We also strongly assume that the natural axiom

$$\mathsf{Cl}(s), \mathsf{Cl}(t) \Rightarrow \mathsf{Cl}(s \dot{\to} t)$$

does not increase the strength of the system T^+ or any of its mentioned extensions. It would be interesting to see if one could prove these assumptions using a generalisation of the proposed realisation approach.

Chapter 4

Embeddings between weak theories of truth and explicit mathematics

The contents of this chapter are joint work with Thomas Strahm, and have been published as [32].

4.1 Introduction

In this chapter, we study the relationship between weak applicative theories of truth and corresponding theories of explicit mathematics. In particular, we consider two truth theories: $\mathsf{T_{PR}}$ of primitive recursive strength, and the previously introduced theory $\mathsf{T_{PT}}$. The theory $\mathsf{T_{PR}}$ is just $\mathsf{T_{PT}}$ with

$$a \in \mathsf{W} \to (\mathsf{T}(\dot{\mathsf{W}}ab) \leftrightarrow b \leq_{\mathsf{W}} a)$$

replaced by

$$a \in \mathsf{W} \leftrightarrow \mathsf{T}(\dot{\mathsf{W}}a).$$

Clearly, $\mathsf{T_{PR}}$ proves the Tarski biconditionals exactly for positive L_{T} formulas which immediately implies its lower bound. The upper bound is achieved by an embedding into PR plus Σ^0_1 induction by using the formalized term model construction used in the proof of Theorem 9 in Cantini [19]. Alternatively, one can also use Cantini's realisation approach [18] to prove the upper bound.

We will see that the truth theories can interpret corresponding systems of explicit mathematics very directly, whereas reverse embeddings of truth theories into explicit mathematics are more elaborate and require additional assumptions.

The chapter is structured as follows: In section 2, we present two natural systems of explicit mathematics of polynomial and primitive recursive strength, respectively: the system PETJ of Spescha and Strahm [73, 72, 74] and the system EPCJ; both of these frameworks are direct subsystems of Feferman's EM_0 plus the join principle (cf. [33, 34]). For the embedding of truth theories into explicit mathematics, further principles will be needed, for example, the existence of universes, and Cantini's uniformity principle. Section 3 is devoted to mutual embeddings of weak truth theories and systems of explicit mathematics. Firstly, we will see that PETJ and EPCJ are very directly contained in T_{PT} and T_{PR}, respectively. The reverse embeddings are more difficult: (i) for the direct embedding of T_{PR} into EPCJ we assume the existence of a universe and the uniformity principle; (ii) the reduction of T_{PT} to PETJ proceeds via an intermediate levelled truth theory, which in turn can be directly modelled in an extension of PETJ by universes. In Section 4, we discuss the proof-theory of weak systems of explicit mathematics, including the mentioned ones.

4.2 Explicit mathematics

Types in explicit mathematics are collections of operations and must be thought of as being generated successively from preceding ones. They are represented by operations via a suitable *naming relation* $\Re$. Types are extensional and have (explicit) names which are intensional. The formalization of explicit mathematics using a naming relation $\Re$ is due to Jäger [52].

We will present the two weak theories of explicit mathematics EPCJ and PETJ and some extensions thereof. We will describe the two theories simultaneously since their axioms differ only slightly.

4.2.1 The language $\mathbb{L}$ of explicit mathematics

The language $\mathbb{L}$ is a two-sorted language extending L by

- type variables $U, V, W, X, Y, Z, \dots$
- binary relation symbols $\Re$ (naming) and $\in$ (elementhood)
- new (individual) constants w (sets of words), id (identity), un (union), int (intersection), dom (domain), all (forall), inv (inverse image), and j (join)

The *formulas* $(A, B, C, \dots)$ of $\mathbb{L}$ are built from the atomic formulas of L as well as from formulas of the form

$$(s \in X), \quad \Re(s, X), \quad (X = Y)$$

by closing under the propositional connectives and quantification in both sorts of variables. The formula $\Re(s, X)$ reads as "the individual s is a name of (or represents) the type X".

We use the following abbreviations:

$$\begin{aligned} \Re(s) &:= (\exists X)\Re(s, X), \\ s \in t &:= (\exists X)(\Re(t, X) \land s \in X). \end{aligned}$$

4.2.2 Two theories of explicit mathematics

In the following we spell out the axioms of the system EPCJ whose characteristic axioms are elementary positive comprehension and join. The applicative basis of EPCJ is B^+ as for all theories of explicit mathematics studied in this chapter. Hence their underlying logic is ordinary two-sorted classical predicate logic.

The following axioms state that each type has a name, that there are no homonyms and that equality of types is extensional.

Ontological axioms:

(O1) $\quad (\exists x)\Re(x, X)$

(O2) $\quad \Re(a, X) \land \Re(a, Y) \rightarrow X = Y$

(O3) $\quad (\forall z)(z \in X \leftrightarrow z \in Y) \rightarrow X = Y$

The following axioms provide a finite axiomatisation of the schema of positive elementary comprehension and join.

Type existence axioms:

(w$_{\mathsf{PR}}$) $\Re(\mathsf{w}) \wedge (\forall x)(x \in \mathsf{w} \leftrightarrow x \in \mathsf{W})$

(id) $\Re(\mathsf{id}) \wedge (\forall x)(x \in \mathsf{id} \leftrightarrow x = \langle \mathsf{p}_0 x, \mathsf{p}_1 x\rangle \wedge \mathsf{p}_0 x = \mathsf{p}_1 x)$

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

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

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

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

(all) $\Re(a) \to \Re(\mathsf{all}(a)) \wedge (\forall x)(x \in \mathsf{all}(a) \leftrightarrow (\forall y)(\langle x,y\rangle \in a))$

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

(j.2) $\Re(a) \wedge (\forall x \in a)\Re(fx) \to (\forall x)(x \in \mathsf{j}(a,f) \leftrightarrow \Sigma(f,a,x))$

where $\Sigma(f,a,x)$ is the formula

$$(\exists y)(\exists z)(x = \langle y,z\rangle \wedge y \in a \wedge z \in fy)$$

The only difference between EPCJ and PETJ is that for PETJ we replace the axiom (w$_{\mathsf{PR}}$) by (w$_{\mathsf{PT}}$).

(w$_{\mathsf{PT}}$) $\quad a \in \mathsf{W} \;\to\; \Re(\mathsf{w}(a)) \wedge (\forall x)(x \in \mathsf{w}(a) \leftrightarrow x \leq_{\mathsf{W}} a)$

In contrast to the comprehension schema available in EPCJ, in PETJ it is not claimed that the collection of binary words forms a type, but merely that for each word a, the collection $\{x \in \mathsf{W} : x \leq a\}$ forms a type, uniformly in a.

Finally, both theories include the principle of type induction along W.

Type induction on W:

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

The finite axiomatisations of elementary comprehension in EPCJ and PETJ immediately imply corresponding schemes of elementary comprehension with respect to the characteristic formula classes of EPCJ and PETJ, respectively.

Lemma 73 (Positive Comprehension) *Let $A[a, \vec{x}, \vec{X}]$ be a positive $\mathbb{L}$ formula with exactly the free variables displayed which does neither contain the predicate $\Re$ nor second order quantifiers. Then there exists a term $t_A[\vec{x}, \vec{z}]$ with exactly the free variables displayed such that* EPCJ *proves*

$$\Re(\vec{z}, \vec{X}) \;\rightarrow\; \Re(t_A[\vec{x}, \vec{z}]) \wedge (\forall a)\big(a \in t_A[\vec{x}, \vec{z}] \leftrightarrow A[a, \vec{x}, \vec{X}]\big)$$

Definition 74 (Simple $\mathbb{L}$ formulas) *Let A be a positive $\mathbb{L}$ formula which does neither contain the predicate $\Re$ nor second order quantifiers. Then the formula A^u which is obtained by replacing each subformula of the form $t \in \mathsf{W}$ of A by $t \leq_{\mathsf{W}} u$ is called simple.*

Note, that simple $\mathbb{L}$ formulas are defined in analogy to simple L_{T} formulas 5.

Lemma 75 (Simple Comprehension) *Let $A^u[u, a, \vec{x}, \vec{X}]$ be a simple $\mathbb{L}$ formula with exactly the free variables displayed. Then there exists a term $t_A[u, \vec{x}, \vec{z}]$ with exactly the free variables displayed such that* PETJ *proves*

$$u \in \mathsf{W} \wedge \Re(\vec{z}, \vec{X}) \;\rightarrow\; \Re(t_A[u, \vec{x}, \vec{z}]) \wedge (\forall a)\big(a \in t_A[u, \vec{x}, \vec{z}] \leftrightarrow A^u[u, a, \vec{x}, \vec{X}]\big)$$

Lemma 73 and Lemma 75 allow us to use set notation. We will sometimes write $\{x : A[x]\}$ instead of t_A where t_A is defined as above.

4.2.3 Extensions

In standard models of explicit mathematics, the elementhood and naming relation are constructed in stages; the same applies to standard models of truth theories, see Feferman [33] and Cantini [13]. Beginning with sentences which can be immediately seen to be true, we establish the truth of more complex statements. The truth predicate can then be conceived as joining the truth stage predicates of at least ω many stages. To interpret this object in explicit mathematics, we will define recursively types corresponding to particular stages. However the admissibility of an iterative definition of types presupposes induction on W for the naming predicate to prove that the type constructors work as intended. In the following, we will expand our theories of explicit mathematics such that name induction is possible at least in a restricted way. But instead of expanding these theories by a type reflecting the name predicate, we vote for the weaker and more usual extension by universes. This additional expressive power makes it possible to interpret the truth theories T_{PR} and T_{PT} respectively.

A universe is a type U such that:

- U is closed under (positive) elementary comprehension and join;
- All elements of U are names.

To introduce universes precisely we define a closure condition in the following way: $C_{\mathsf{EPCJ}}(z, a)$ holds iff one of the following conditions is satisfied:

- $a = \mathsf{w}$
- $a = \mathsf{id}$
- $(\exists x)(\exists f)(a = \mathsf{inv}(f, x) \wedge x \in z)$
- $(\exists x)(\exists y)(a = \mathsf{un}(x, y) \wedge x \in z \wedge y \in z)$
- $(\exists x)(\exists y)(a = \mathsf{int}(x, y) \wedge x \in z \wedge y \in z)$
- $(\exists x)(a = \mathsf{dom}(x) \wedge x \in z)$
- $(\exists x)(a = \mathsf{all}(x) \wedge x \in z)$
- $(\exists x)(\exists f)\big[a = \mathsf{j}(x, f) \wedge x \in z \wedge (\forall y \in x)(fy \in z)\big]$

For PETJ we have to adapt the first condition: we replace $a = \mathsf{w}$ by the formula $(\exists u \in \mathsf{W})(a = \mathsf{w}u)$. We call the modified formula $C_{\mathsf{PETJ}}(z, a)$.

Assuming that z is a name, the formula $(\forall x)(C_{\mathsf{EPCJ}}(z, x) \rightarrow x \in z)$ expresses that z names a type that is closed under the type constructors of the theory EPCJ; analogously for PETJ. We abbreviate the formula

$$(\forall x)(C_{\mathsf{EPCJ}}(z, x) \rightarrow x \in z) \wedge (\forall x)(x \in z \rightarrow \Re(x)) \wedge \Re(z)$$

by $\mathsf{U}_{\mathsf{EPCJ}}(z)$; the formula $\mathsf{U}_{\mathsf{PETJ}}(z)$ is defined analogously.

Next assume that the language $\mathbb{L}$ contains two additional constants ℓ_{EPCJ} and ℓ_{PETJ}. The following two axioms state that ℓ_{EPCJ} and ℓ_{PETJ} create an EPCJ or PETJ universe respectively, if applied to a name.

($\mathsf{U}_{\mathsf{EPCJ}}$) $\qquad \Re(a) \to \mathsf{U}_{\mathsf{EPCJ}}(\ell_{\mathsf{EPCJ}}(a)) \wedge a \in \ell_{\mathsf{EPCJ}}(a)$

($\mathsf{U}_{\mathsf{PETJ}}$) $\qquad \Re(a) \to \mathsf{U}_{\mathsf{PETJ}}(\ell_{\mathsf{PETJ}}(a)) \wedge a \in \ell_{\mathsf{PETJ}}(a)$

In order to keep the notation simple, we write $\mathsf{EPCJ} + \mathsf{U}$ instead of $\mathsf{EPCJ} + \mathsf{U}_{\mathsf{EPCJ}}$ and analogously $\mathsf{PETJ} + \mathsf{U}$ instead of $\mathsf{PETJ} + \mathsf{U}_{\mathsf{PETJ}}$. Similarly, we drop the subscript of ℓ_{EPCJ} and ℓ_{PETJ} if it is clear from the context.

Using the universe it is possible to code the elementhood relation of its types by using a suitable join.

Lemma 76 *There exists a closed term* e *such that for* $\mathsf{Th} = \mathsf{PETJ} + \mathsf{U}$ *or* $\mathsf{EPCJ} + \mathsf{U}$ *we have that* Th *proves*

$$\Re(a) \to \Re(\mathsf{e}(a)) \wedge (\forall x)\big(x \in \mathsf{e}(a) \leftrightarrow (\exists y)(\exists z)(x = \langle y, z\rangle \wedge z \in \ell(a) \wedge y \in z\big)$$

Below, Cantini's uniformity principle (cf. [18]) is needed for the embedding of T_{PR}. It claims for each positive $\mathbb{L}$ formula A

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

This concludes the description of the relevant extensions of explicit mathematics.

4.3 Embeddings

We will embed the theories of explicit mathematics with universes into the theories of truth and vice versa. The embedding of the theories of explicit mathematics with universes is straightforward. The reverse embeddings are more difficult and work in a different way for both theories of truth: it seems to be impossible to embed T_{PT} into $\mathsf{PETJ} + \mathsf{U}$ directly. Instead we embed a levelled theory of truth to which T_{PT} is reducible by an asymmetric interpretation argument.

In this section we assume an equivalent first order formulation of EPCJ and PETJ.

The first order formulations postulate the type-theoretic axioms directly via a unary naming predicate $\Re$ and a binary elementhood relation $\in$ between *individuals.* The first and the second order versions can be mutually embedded for both theories of explicit mathematics. For details about the embedding, see e.g. Spescha [72] or Spescha and Strahm [74].

4.3.1 Embedding weak theories of explicit mathematics into weak truth theories

For both weak truth theories introduced in the chapter, the embedding works completely analogously. We take the embedding of $\mathsf{EPCJ} + \mathsf{U}$ into $\mathsf{T_{PR}}$ as example. The main idea is to interpret the elementhood relation by using the truth predicate and to trivialize the universes. The translation * of a formula $s \in t$ will be

$$\mathsf{T}(t^* s^*).$$

To make this translation work, we have to interpret the type constructors in the right way. The idea is to translate them by predicates, which embody their membership condition.

Definition 77 (Translation of terms) *For each term t of $\mathbb{L}$, its translation t^* into L_T is defined recursively on the complexity of t in the following way.*

- *All applicative constants are left untouched.*
- $\mathsf{id}^* \equiv \lambda z.z \mathbin{\dot{=}} \langle \mathsf{p}_0 z, \mathsf{p}_1 z\rangle \mathbin{\dot{\wedge}} \mathsf{p}_0 z \mathbin{\dot{=}} \mathsf{p}_1 z$
- $\mathsf{w}^* \equiv \lambda z.\ \dot{\mathsf{W}} z$
- $\mathsf{int}^* \equiv \lambda a.\lambda b.\lambda z.\ az \mathbin{\dot{\wedge}} bz$
- $\mathsf{un}^* \equiv \lambda a.\lambda b.\lambda z.\ az \mathbin{\dot{\vee}} bz$
- $\mathsf{inv}^* \equiv \lambda f.\lambda a.\lambda z.\ a(fz)$

- $\mathsf{dom}^* \equiv \lambda a.\lambda z.\ \dot{\exists}\lambda y.a\langle z, y\rangle$
- $\mathsf{all}^* \equiv \lambda a.\lambda z.\ \dot{\forall}\lambda y.a\langle z, y\rangle$
- $\mathsf{j}^* \equiv \lambda f.\lambda a.\lambda z.\ \dot{\exists}\lambda x.\dot{\exists}\lambda y.z \doteq \langle x, y\rangle \mathbin{\dot{\wedge}} ax \mathbin{\dot{\wedge}} (fx)y$
- ${\ell_{\mathsf{EPCJ}}}^* \equiv \lambda a.\lambda z.0 \doteq 0$
- $st^* \equiv s^* t^*$

Formulas are translated in the following way: atomic formulas commute with * except for the formulas of the shape $s \in t$ whose translation is $\mathsf{T}(t^* s^*)$ and the formulas of the shape $\Re(s)$ whose translation is $0 = 0$. The translation commutes with implication, propositional connectives and quantifiers.

For this translation, the embedding theorem below is proved without difficulties. Since the name predicate is interpreted trivially, the translations of the universe axioms hold in T_{PR}. Moreover, the translation can be modified in the obvious way in order to provide an embedding of $\mathsf{PETJ} + \mathsf{U}$ into T_{PT}.

Theorem 78 $\mathsf{EPCJ}+\mathsf{U}$ *and* $\mathsf{PETJ}+\mathsf{U}$ *are contained in* T_{PR} *and* T_{PT} *via the* * *translation or a slight modification thereof, respectively.*

Let us mention that this embedding theorem also holds for expansions of explicit mathematics by positive uniformity if the truth theories are expanded analogously.

4.3.2 Embedding of T_{PR} into $\mathsf{EPCJ} + \mathsf{U} + \mathsf{UP}$

First, we define types corresponding to levels of truth. We construct a truth-level type τw for each word w. Using join we can then collect all these truth-level types. Because of UP the resulting type satisfies the translated truth axioms.

The types τw for the truth levels all consist of tuples of three elements. The first element contains a code for a logical symbol of L_{T}. All these codes are assumed to be different words. The second and the third element stand for the terms the logical constant is applied to. The third element is sometimes only a place holder. Let us define the bottom truth level $\tau\epsilon$ as

$$\{\langle a,b,c\rangle \mid (a = \ulcorner \dot{=} \urcorner \wedge b = c) \vee (a = \ulcorner \dot{\mathsf{W}} \urcorner \wedge \mathsf{W}(b) \wedge c = \epsilon)\}.$$

The types for the higher truth levels are defined recursively (using the fixed point theorem of B) in the following way:

$$\begin{aligned} \tau(\mathsf{s}_i w) := \tau w \cup \{\langle a,b,c\rangle \mid \ & \big(a = \ulcorner \dot{\wedge} \urcorner \wedge b \in \tau w \wedge c \in \tau w\big) && \vee \\ & \big(a = \ulcorner \dot{\vee} \urcorner \wedge [b \in \tau w \vee c \in \tau w]\big) && \vee \\ & \big(a = \ulcorner \dot{\exists} \urcorner \wedge (\exists x)(bx \in \tau w) \wedge c = \epsilon\big) && \vee \\ & \big(a = \ulcorner \dot{\forall} \urcorner \wedge (\forall x)(bx \in \tau w) \wedge c = \epsilon\big) && \} \end{aligned}$$

To justify the type notation, we have to show that the above given terms τw are names for each $w \in \mathsf{W}$. Only then, the type constructors work in the intended way and indeed name the above displayed types. Since $\mathfrak{R}$-induction is not available, we use type induction with the universe $\ell_{\mathsf{EPCJ}}(\mathsf{id})$ for this purpose. It is easy to see that $\tau\epsilon \in \ell_{\mathsf{EPCJ}}(\mathsf{id})$ and

$$(\forall w \in \mathsf{W})(\tau w \in \ell_{\mathsf{EPCJ}}(\mathsf{id}) \rightarrow \tau(\mathsf{s}_i w) \in \ell_{\mathsf{EPCJ}}(\mathsf{id}))$$

hold. We apply type induction and use the fact $(\forall x)\big(x \in \ell_{\mathsf{EPCJ}}(\mathsf{id}) \rightarrow \mathfrak{R}(x)\big)$ to get the desired result.

Now the stage is set to define a translation $^\circ$ of $\mathsf{T_{PR}}$ into $\mathsf{EPCJ} + \mathsf{U} + \mathsf{UP}$. In particular, we translate the truth predicate using the above defined hierarchy of types.

Definition 79 (Translation of terms) *For each term t of L_{T}, its translation t° is defined inductively on its complexity in the following way.*

- *All applicative constants are left untouched.*
- $\dot{=}^\circ \equiv \lambda x.\lambda y.\langle \ulcorner \dot{=} \urcorner, x, y\rangle$
- $\dot{\mathsf{W}}^\circ \equiv \lambda x.\langle \ulcorner \dot{\mathsf{W}} \urcorner, x, \epsilon\rangle$
- $\dot{\wedge}^\circ \equiv \lambda x.\lambda y.\langle \ulcorner \dot{\wedge} \urcorner, x, y\rangle$

- $\dot{\vee}^\circ \equiv \lambda x.\lambda y.\langle \ulcorner \dot{\vee} \urcorner, x, y\rangle$
- $\dot{\exists}^\circ \equiv \lambda x.\langle \ulcorner \dot{\exists} \urcorner, x, \epsilon\rangle$
- $\dot{\forall}^\circ \equiv \lambda x.\langle \ulcorner \dot{\forall} \urcorner, x, \epsilon\rangle$
- $(st)^\circ \equiv s^\circ t^\circ$

Definition 80 (Translation of formulas) *For each formula A of L_{T}, its translation A° is defined inductively in the following way.*

- $(s = t)^\circ \equiv s^\circ = t^\circ$
- $(s \in \mathsf{W})^\circ \equiv s^\circ \in \mathsf{W}$
- $\mathsf{T}(t)^\circ \equiv t^\circ \in \mathsf{dom}(\mathsf{inv}(\lambda x.\langle \mathsf{p}_1 x, \mathsf{p}_0 x\rangle, \mathsf{j}(\mathsf{W}, \tau)))$
- *The translation commutes with implication, propositional connectives and quantifiers.*

Note that by the type axioms of $\mathsf{EPCJ} + \mathsf{U}$, we have

$$t \in \mathsf{dom}(\mathsf{inv}(\lambda x.\langle \mathsf{p}_1 x, \mathsf{p}_0 x\rangle, \mathsf{j}(\mathsf{W}, \tau))) \leftrightarrow (\exists w \in \mathsf{W})(t \in \tau(w))$$

We are now ready to state the embedding of T_{PR} into $\mathsf{EPCJ} + \mathsf{U} + \mathsf{UP}$.

Theorem 81 T_{PR} *is contained in* $\mathsf{EPCJ} + \mathsf{U} + \mathsf{UP}$ *via the* $^\circ$ *translation.*

Proof. It is clear that the translations of the applicative axioms hold in $\mathsf{EPCJ} + \mathsf{U}$. Further, we can show that the translation of truth induction holds in $\mathsf{EPCJ} + \mathsf{U}$ using inv. So let us check the translations of the truth axioms. The direction from right to left is always trivially fulfilled except for $(\dot{\forall})$. Its translation is in $\mathsf{EPCJ} + \mathsf{U}$ equivalent to

$$(\forall x)(\exists w \in \mathsf{W})(fx \in \tau w) \rightarrow (\exists w \in \mathsf{W})(\langle \ulcorner \dot{\forall} \urcorner, f, \epsilon\rangle \in \tau w).$$

Using UP, from the antecedens we can derive the existence of a $w \in \mathsf{W}$ such that $(\forall x)(fx \in \tau w)$. This implies that $\langle \ulcorner \dot{\forall} \urcorner, f, \epsilon\rangle$ is in the successor truth level type $\tau(\mathsf{s}_i w)$.

The direction from left to right is always proved in the same way. We sketch the proof for the $\dot{\wedge}$-axiom. Let $\dot{-}$ be defined in the following way.

- $x \dot{-} \epsilon := x$
- $x \dot{-} \mathsf{s}_i w := \mathsf{p}_{\mathsf{W}}(x \dot{-} w)$

Let us assume the left-hand side of the $\dot{\wedge}$-axiom. Its translation implies in $\mathsf{EPCJ} + \mathsf{U}$

$$\langle \ulcorner \dot{\wedge} \urcorner, a, b \rangle \in \tau w$$

for a $w \in \mathsf{W}$ unequal ϵ. We define the formula $A[x]$ as[1]

$$\langle \ulcorner \dot{\wedge} \urcorner, a, b \rangle \in \tau(w \dot{-} x) \vee (\exists y \in \mathsf{W})(y \subset w \wedge a \in \tau y \wedge b \in \tau y).$$

Clearly, this formula is progressive in W due to the construction of the truth level types. By type induction we get

$$\langle \ulcorner \dot{\wedge} \urcorner, a, b \rangle \in \tau \epsilon \vee (\exists y \in \mathsf{W})(y \subset w \wedge a \in \tau y \wedge b \in \tau y).$$

Since the bottom level of truth does not contain tuples of the form $\langle \ulcorner \dot{\wedge} \urcorner, a, b \rangle$, the second disjunct has to be true. This immediately implies the left-hand side of the $\dot{\wedge}$-axiom. $\square$

Note that we needed only the existence of one single universe to prove this embedding. In addition, similarly as described in the next paragraph for T_{PT}, it is also possible to reduce T_{PR} via an intermediate levelled truth theory to $\mathsf{EPCJ} + \mathsf{U}$. This results in a reduction of T_{PR} to $\mathsf{EPCJ} + \mathsf{U}$ which does not depend on the uniformity principle.

4.3.3 Reduction of T_{PT} to $\mathsf{PETJ} + \mathsf{U}$

Unfortunately an embedding similar to the one in the previous subsection does not seem to be possible in this case. This is because we cannot collect the truth levels for all words, since join can have only initial segments of the type of words as index type.

The levelled truth theory $\mathsf{T}^{\ell}_{\mathsf{PT}}$

Because of the above mentioned reasons, we have to reduce the truth theory T_{PT} to a levelled truth theory $\mathsf{T}^{\ell}_{\mathsf{PT}}$. This means that the predicate T in

[1]We use $s \subset t$ as abbreviation for $\mathsf{d}_{\mathsf{W}}(0, 1, s, t) = 1 \wedge \mathsf{c}_{\subseteq} st = 0$

$\mathsf{T}^\ell_{\mathsf{PT}}$ is a binary predicate, whose first argument is written as superscript and interpreted as truth level. This superscript displays the maximal complexity of formulas the corresponding unary truth predicate can reflect. The logical axioms and rules of $\mathsf{T}^\ell_{\mathsf{PT}}$ are the usual ones. $\mathsf{T}^\ell_{\mathsf{PT}}$ has the following non-logical axioms.

- $a \in \mathsf{W} \to (\mathsf{T}^a(x \doteq y) \leftrightarrow x = y)$
- $a, b \in \mathsf{W} \to (x \leq_{\mathsf{W}} b \leftrightarrow \mathsf{T}^a(\dot{\mathsf{W}}bx))$
- $a \in \mathsf{W} \to \big(\mathsf{T}^{\mathsf{s}_i a}(x \mathbin{\dot\vee} y) \leftrightarrow \mathsf{T}^a(x) \vee \mathsf{T}^a(y)\big)$
- $a \in \mathsf{W} \to \big(\mathsf{T}^{\mathsf{s}_i a}(x \mathbin{\dot\wedge} y) \leftrightarrow \mathsf{T}^a(x) \wedge \mathsf{T}^a(y)\big)$
- $a \in \mathsf{W} \to \big(\mathsf{T}^{\mathsf{s}_i a}(\dot\exists f) \leftrightarrow (\exists z)\mathsf{T}^a(fz)\big)$
- $a \in \mathsf{W} \to \big(\mathsf{T}^{\mathsf{s}_i a}(\dot\forall f) \leftrightarrow (\forall z)\mathsf{T}^a(fz)\big)$
- $a_0, a_1 \in \mathsf{W} \wedge a_0 \leq a_1 \wedge \mathsf{T}^{a_0}(x) \to \mathsf{T}^{a_1}(x)$

Additionally, we have truth induction in the following form:

$$\mathsf{T}^{p\epsilon}(f\epsilon) \wedge (\forall x \in \mathsf{W})\big[\mathsf{T}^{px}(fx) \to \mathsf{T}^{p(\mathsf{s}_0 x)}(f(\mathsf{s}_0 x)) \wedge \mathsf{T}^{p(\mathsf{s}_1 x)}(f(\mathsf{s}_1 x))\big]$$
$$\to (\forall x \in \mathsf{W})(\mathsf{T}^{px}(fx))$$

In the following, p will always be a polynomial.

For the asymmetric interpretation of T_{PT} in $\mathsf{T}^\ell_{\mathsf{PT}}$, we bound the truth level and the W predicate simultaneously. We work as usual with sequent style formulations of T_{PT} and $\mathsf{T}^\ell_{\mathsf{PT}}$ which we call T_{PT} and $\mathsf{T}^\ell_{\mathsf{PT}}$ as well. We assume that in these calculi all axioms are formulated for terms to guarantee a sufficient cut elimination.

Definition 82 (Asymmetric interpretation) *Let A be a positive L_{T} formula and let a, b be variables. The formula $A^{a,b}$ is defined in the following way.*

- $(t \in \mathsf{W})^{a,b} \equiv t \leq_{\mathsf{W}} a$
- $\mathsf{T}(t)^{a,b} \equiv \mathsf{T}^b(t)$

Other atomic formulas are untouched by the asymmetric interpretation. The asymmetric interpretation commutes with propositional connectives and quantifiers.

Next we state the crucial asymmetric interpretation lemma of T_{PT} into $\mathsf{T}^{\ell}_{\mathsf{PT}}$. An immediate consequence of the lemma is that the provably total functions of T_{PT} are contained in the provably total functions of $\mathsf{T}^{\ell}_{\mathsf{PT}}$.

Lemma 83 *Let $\Gamma \Rightarrow \Delta$ be a positive sequent which has a proof of depth k in T_{PT} containing only positive formulas. Then there exists a polynomial p of degree $2^{(2^k)}$ such that*[2]

$$\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,b} \Rightarrow \Delta^{pa,pa*b}$$

Proof. We show the lemma by induction on the depth of the positive proof. The only difficult case is induction. In this case we have by induction hypothesis polynomials p, q_0, q_1 of degree $2^{(2^k)}$ with the following properties.

$$\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,b} \Rightarrow \mathsf{T}^{pa*b}(r\epsilon), \Delta^{pa,pa*b} \tag{4.1}$$

$$\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,b}, \mathsf{T}^{b}(rx), x \leq_{\mathsf{W}} a \Rightarrow \mathsf{T}^{q_i a*b}(r(\mathsf{s}_i x)), \Delta^{q_i a, q_i a*b} \tag{4.2}$$

Let q be a polynomial that bounds q_0 and q_1. We define the polynomial g as

$$g(x,y) := p(x) * (q(x) \times \mathsf{s}_0 y).$$

Because of monotonicity of the asymmetric interpretation the following hold:

$$\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,b} \Rightarrow \mathsf{T}^{ga\epsilon*b}(r\epsilon), \Delta^{ga\epsilon,ga\epsilon*b} \tag{4.3}$$

$$\begin{aligned}\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,gax*b}, \mathsf{T}^{gax*b}(rx), x \leq_{\mathsf{W}} a \Rightarrow \\ \mathsf{T}^{ga(\mathsf{s}_i x)*b}(r(\mathsf{s}_i x)), \Delta^{qa,ga(\mathsf{s}_i x)*b}\end{aligned} \tag{4.4}$$

We use again monotonicity of the asymmetric interpretation to unify the side formulas and thus get:

$$\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,b} \Rightarrow \mathsf{T}^{ga\epsilon*b}(r\epsilon), \Delta^{gaa,gaa*b} \tag{4.5}$$

$$\begin{aligned}\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,b}, \mathsf{T}^{gax*b}(rx), x \leq_{\mathsf{W}} a \Rightarrow \\ \mathsf{T}^{ga(\mathsf{s}_i x)*b}(r(\mathsf{s}_i x)), \Delta^{gaa,gaa*b}\end{aligned} \tag{4.6}$$

[2]We use the notation $pa * b$ to denote the term $*(pa, b)$. Analogously for similar notations.

These are the premises for an induction over an initial segment of W which is proved to be admissible as usual. After using induction, monotonicity delivers the following.

(4.7) $$\mathsf{T}^{\ell}_{\mathsf{PT}} \vdash a, b \in \mathsf{W}, \Gamma^{a,b}, x \leq_{\mathsf{W}} a \Rightarrow \mathsf{T}^{gaa*b}(rx), \Delta^{gaa,gaa*b}$$

Since gaa is a polynomial of degree $2^{(2^k)} + 1$ in a this is the desired result. Note that the degree $2^{(2^{k+1})}$ for the bounding polynomial is needed for the cut rule. $\square$

Embedding of $\mathsf{T}^{\ell}_{\mathsf{PT}}$ into $\mathsf{PETJ} + \mathsf{U}$

In analogy to the previous embedding, we construct a closed term τ such that for all $w \in \mathsf{W}$ the type $\tau(w)$ corresponds to the truth level w. We set

$$\tau(\epsilon) := \{\langle a, b, c\rangle | a = \ulcorner \dot{=} \urcorner \wedge b = c \vee a = \ulcorner \dot{\mathsf{W}} \urcorner \wedge \langle c, \mathsf{w}(b)\rangle \in \mathsf{e}(\mathsf{id})\}.$$

The types for the higher truth levels are defined recursively as before:

$$\begin{aligned} \tau(\mathsf{s}_i w) := \tau w \cup \{\langle a, b, c\rangle | \quad & \big(a = \ulcorner \dot{\wedge} \urcorner \wedge b \in \tau w \wedge c \in \tau w\big) & \vee \\ & \big(a = \ulcorner \dot{\vee} \urcorner \wedge [b \in \tau w \vee c \in \tau w]\big) & \vee \\ & \big(a = \ulcorner \dot{\exists} \urcorner \wedge (\exists x)(bx \in \tau w) \wedge c = \epsilon\big) & \vee \\ & \big(a = \ulcorner \dot{\forall} \urcorner \wedge (\forall x)(bx \in \tau w) \wedge c = \epsilon\big) & \} \end{aligned}$$

As above, we show that these levels are all names by type induction with the universe $\ell(\mathsf{e}(\mathsf{id}))$.

We are now ready to modify the translation $^\circ$ from the last subsection in order to provide a translation from $\mathsf{T}^{\ell}_{\mathsf{PT}}$ into $\mathsf{PETJ} + \mathsf{U}$.

Definition 84 (Translation of terms) *For each term t of L_{T} its translation t° into $\mathbb{L}$ is defined in the same way as above except that we put*

$$\dot{\mathsf{W}}^\circ \equiv \lambda x.\lambda y.\langle \ulcorner \dot{\mathsf{W}} \urcorner, x, y\rangle$$

Definition 85 (Translation of formulas) *For each formula A of L_{T} its translation A° is defined inductively in the following way.*

- $(s = t)^\circ \equiv s^\circ = t^\circ$

- $(s \in \mathsf{W})^\circ \equiv s^\circ \in \mathsf{W}$
- $\mathsf{T}^s(t)^\circ \equiv \langle t^\circ, \tau(0 \times s^\circ)\rangle \in \mathsf{e}(\mathsf{e}(\mathsf{id}))$
- *The translation commutes with implication, propositional connectives and quantifiers.*

Similarly to the previous embedding theorem, we now obtain the following result.

Theorem 86 $\mathsf{T}^{\ell}_{\mathsf{PT}}$ *is contained in* $\mathsf{PETJ} + \mathsf{U}$ *via the* $^\circ$ *translation.*

Proof. The translations of the truth axioms are proved as before; note that the induction formula is equivalent to a simple formula. Because of the levelling the uniformity principle is not needed. An easy induction establishes $v, w \in \mathsf{W} \wedge v \subset w \wedge \mathsf{T}^v(x) \to \mathsf{T}^w(x)$, which implies the translation of monotonicity. $\square$

Note that the previous lemma and theorem readily imply that the provably total functions of T_{PT} are contained in those of $\mathsf{PETJ} + \mathsf{U}$.[3]

4.4 Proof-theoretic analysis

In this section we give an overview of the literature regarding the proof theory of weak systems of explicit mathematics, including the ones discussed in this chapter. For an overview about the proof theory of weak systems of truth, we refer to section 2.1.

Let us start by briefly reviewing some previous proof-theoretic work regarding systems of explicit mathematics of strength PR. Early systems of flexible typing of this strength are extensively studied by Feferman. In [36, 37] he

[3]The theory T_{PT} can also be reduced to an extension of polynomial strength of PETJ by a *single* universe and an additional type constructor to deal with sharply bounded universal quantification. In this case the intermediate reduction theory is a twice levelled theory of truth whose second level designates the maximal value of s to which formulas of the form $t \leq_{\mathsf{W}} s$ are reflected. The corresponding truth levels can be interpreted in this extension of PETJ without using $\mathsf{e}(\mathsf{id})$.

proposes a program to use explicit mathematics to analyse properties of functional programs. Theories that are strongly related to the system EPCJ are considered in Krähenbühl [63]. Let us note that all upper bound computations for the systems mentioned above proceed via embeddings into suitable subsystems of Peano arithmetic of strength PR.

Let us now turn to the discussion of systems of explicit mathematics of polynomial strength which were first introduced and studied in Spescha and Strahm [73]. There the system PET of types and names[4] has been proposed whose provably total functions are exactly the polytime functions. The PhD thesis of Spescha [72] gives a uniform treatment of various weak systems of explicit mathematics in the spirit of PET, possibly augmented by the axiom of join, see also the article Spescha and Strahm [74]. The very recent work of Probst [69] solves the delicate and difficult problem of showing that the provably total operations of the system PET with the join axiom and *classical* logic are still the polynomial time computable ones. Strahm's article [81] surveys most of these results.

The upper bounds of the theories of polynomial strength mentioned above are nearly exclusively determined by realisation approaches in the style of Strahm's [79]. An exception is Probst's [69] which employs model theoretic methods to reduce a system of explicit mathematics to the underlying applicative system.

Let us switch now to the systems introduced in this chapter. The proof that T_{PR} is of primitive recursive strength is sketched in the introduction, the feasible upper bound proof of T_{PT} is contained in chapter 2. Both systems can be extended by UP without increasing their strength since this axiom can be realised trivially. The embeddings presented in the last section deliver upper bound proofs for the systems EPCJ and PETJ, possibly extended by universes and UP. Alternatively, upper bounds for these theories are obtained using realisation techniques. For both theories of explicit mathematics, U is a consequence of $\forall\mathfrak{R}$, since under this assumption, ℓa can be interpreted as $\{x : x = x\}$ for any a. Then, we employ realisation approaches that trivialize the name predicate. For EPCJ and its extensions we use an approach in the

[4]In the notation of this chapter, PET is PETJ without the join axioms.

style of Cantini's [18]. The realisation approach presented in the last chapter works for PETJ and its extensions.

Let us conclude this section by summarizing the results about the proof-theoretic strength of the theories of truth and explicit mathematics considered in this article.

Theorem 87 (Systems of primitive recursive strength) *The provably total functions of the following theories are the primitive recursive ones:*

1. EPCJ*, possibly augmented by* UP *and* U*;*
2. T_{PR}*, possibly augmented by* UP*.*

Theorem 88 (Systems of polynomial strength) *The provably total functions of the following theories are the polynomial time computable ones:*

1. PETJ*, possibly augmented by* UP *and* U*;*
2. T_{PT}*, possibly augmented by* UP*.*

4.5 Concluding remarks

We have studied two natural truth-theoretic frameworks over combinatory logic and their relationship to weak systems of explicit mathematics. We have seen that the embedding of explicit mathematics into truth theories is very straightforward, whereas the *direct* reverse embeddings require further (natural) extensions of explicit mathematics and sometimes even intermediate reduction steps. The corresponding extensions of explicit mathematics do not increase the proof-theoretic strength of the underlying systems.

Chapter 5

Unfolding feasible arithmetic

The contents of this chapter are joint work with Thomas Strahm, and have been accepted for publication as [31].

5.1 Introduction

In this chapter we continue Feferman's unfolding program initiated in [38] which uses the concept of the unfolding $\mathcal{U}(\mathsf{S})$ of a schematic system S in order to describe those operations, predicates and principles concerning them, which are implicit in the acceptance of S. The program has been carried through for a schematic system of non-finitist arithmetic NFA in Feferman and Strahm [43] and for a system FA (with and without Bar rule) in Feferman and Strahm [44]. The present contribution elucidates the concept of unfolding for a basic schematic system FEA of feasible arithmetic. Apart from the operational unfolding $\mathcal{U}_0(\mathsf{FEA})$ of FEA, we study two full unfolding notions, namely the predicate unfolding $\mathcal{U}(\mathsf{FEA})$ and a more general truth unfolding $\mathcal{U}_{\mathsf{T}}(\mathsf{FEA})$ of FEA, the latter making use of a truth predicate added to the language of the operational unfolding. The main results obtained are that the provably total functions on binary words for all three unfolding systems are precisely those being computable in polynomial time. The upper bound computations make essential use of the theory of truth T_{PT} introduced in chapter 2.

The notion of *unfolding a schematic formal system* was introduced in Feferman [38] in order to answer the following question:

> *Given a schematic system* S*, which operations and predicates, and which principles concerning them, ought to be accepted if one has accepted* S*?*

A paradigmatic example of a schematic system S is the basic system NFA of non-finitist arithmetic. In Feferman and Strahm [43], three unfolding systems for NFA of increasing strength have been analysed and characterized in more familiar proof-theoretic terms; in particular, it was shown that the full unfolding of NFA, $\mathcal{U}(\mathsf{NFA})$, is proof-theoretically equivalent to predicative analysis. For more information on the path to the unfolding program, especially with regard to predicativity and the implicitness program, see Feferman [39].

More recently, the unfolding notions for a basic schematic system of finitist arithmetic, FA, and for an extension of that by a form BR of the so-called bar rule have been worked out in Feferman and Strahm [44]. It is shown that $\mathcal{U}(\mathsf{FA})$ and $\mathcal{U}(\mathsf{FA}+\mathsf{BR})$ are proof-theoretically equivalent, respectively, to primitive recursive arithmetic, PRA, and to Peano arithmetic, PA.

The aim of the present contribution is to elucidate the concept of unfolding in the context of a natural schematic system FEA for *feasible arithmetic*. We will sketch various unfoldings of FEA using a typing discipline or a partial truth predicate, respectively.

The basic schematic system FEA of feasible arithmetic is based on a language for binary words generated from the empty word by the two binary successors S_0 and S_1; in addition, it includes some natural basic operations on the binary words like, for example, word concatenation and multiplication. The logical operations of FEA are conjunction ($\wedge$), disjunction ($\vee$), and the bounded existential quantifier ($\exists^{\leq}$). FEA is formulated as a system of sequents in this language: apart from the defining axioms for basic operations on words, its heart is a schematically formulated, i.e. open-ended induction rule along the binary words, using a free predicate letter P.

The operational unfolding $\mathcal{U}_0(\mathsf{FEA})$ of FEA extends FEA by a general background theory of combinatory algebra and tells us which operations on words are implicit in the acceptance of FEA. It further includes the generalized substitution rule from Feferman and Strahm [44], which allows arbitrary formulas to be substituted for free predicates in derivable rules of inference such as, for example, the induction rule. We will see that $\mathcal{U}_0(\mathsf{FEA})$ derives the totality of precisely the polynomial time computable functions.

The full predicate unfolding $\mathcal{U}(\mathsf{FEA})$ of FEA tells us, in addition, which predicates and operations on them ought to be accepted if one accepts FEA. It presupposes each logical operation of FEA as an operation on predicates. Predicates themselves are just represented as special operations equipped with an elementhood relation on them. We may further accept the formation of the disjoint union of a (bounded with respect to $\leq$) sequence of predicates given by a corresponding operation. It will turn out that the provably convergent functions of $\mathcal{U}(\mathsf{FEA})$ are still the polynomial time computable ones.

We will also describe an alternative way to define the full unfolding of FEA which makes use of a truth predicate T which mimics the logical operations of FEA in a natural way and makes explicit the requirement that implicit in the acceptance of FEA is the ability to reason about truth in FEA. The truth unfolding $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ is thus obtained by extending the combinatory algebra by a unary truth predicate. Indeed, $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ contains the predicate unfolding $\mathcal{U}(\mathsf{FEA})$ in a natural way, including the disjoint union operator for predicates. Moreover, the truth unfolding is proof-theoretically equivalent to the predicate unfolding in the sense that its provably convergent functions on the binary words are precisely the polytime functions.

The upper bound computations for both $\mathcal{U}(\mathsf{FEA})$ and $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ will be obtained via straightforward embeddings into T_PT. Some special care and additional considerations are needed in order to treat their generalized substitution rules.

5.2 The basic schematic system FEA

In this section we introduce the basic schematic system FEA of feasible arithmetic. Its intended universe of discourse is the set of finite binary words and its basic operations and relations include the binary successors S_0 and S_1, the predecessor Pd, the initial subword relation $\subseteq$, word concatenation $\circledast$ as well as word multiplication $\boxtimes$.[1] The logical operations of FEA are conjunction ($\wedge$), disjunction ($\vee$), and bounded existential quantification ($\exists^{\leq}$). As in the case of finitist arithmetic FA, the statements proved in FEA are sequents of formulas in the given language, i.e. implication is allowed at the outermost level.

5.2.1 The language of FEA

The language $\mathcal{L}_0$ of FEA contains a countably infinite supply of variables $\alpha, \beta, \gamma, \ldots$ (possibly with subscripts). These variables are interpreted as ranging over the set of binary words $\mathbb{W}$. $\mathcal{L}_0$ includes a constant ϵ for the empty word, three unary function symbols $\mathsf{S}_0, \mathsf{S}_1, \mathsf{Pd}$ and three binary function symbols $\circledast$, $\boxtimes$, $\subseteq$.[2] Terms of $\mathcal{L}_0$ are defined as usual and are denoted by $\sigma, \tau, \ldots$. Further, $\mathcal{L}_0$ contains the binary predicate symbol $=$ for equality, and an infinite supply $P_0, P_1, \ldots$ of free predicate letters.

The atomic formulas of $\mathcal{L}_0$ are of the form $(\sigma = \tau)$ and $P_i(\sigma_1, \ldots, \sigma_n)$ for $i \in \mathbb{N}$. The formulas are closed under $\wedge$ and $\vee$ as well as under bounded existential quantification. In particular, if A is an $\mathcal{L}_0$ formula, then $(\exists \alpha \leq \tau)A$ is an $\mathcal{L}_0$ formula as well, where τ is not allowed to contain α. In analogy to the previous chapters, we use $\sigma \leq \tau$ as an abbreviation for $1 \boxtimes \sigma \subseteq 1 \boxtimes \tau$, thus expressing that the length of σ is less than or equal to the length of τ.

5.2.2 Axioms and rules of FEA

FEA is formulated as a Gentzen-style sequent calculus with sequents of the form $\Gamma \rightarrow A$. We write $\rightarrow$ instead of $\Rightarrow$ to separate left and right sides of sequents for notational reasons. We assume the usual axioms and rules, in particular, the bounded existential quantifier is governed by the following

[1] Given two words w_1 and w_2, the word $w_1 \boxtimes w_2$ denotes the length of w_2 fold concatenation of w_1 with itself.

[2] We assume that $\subseteq$ defines the characteristic function of the initial subword relation. Further, we employ infix notation for these binary function symbols.

rules, where the usual variable conditions apply:

(E1) $$\frac{\Gamma \rightarrow \sigma \leq \tau \wedge A[\sigma]}{\Gamma \rightarrow (\exists \beta \leq \tau) A[\beta]}$$

(E2) $$\frac{\Gamma, \alpha \leq \tau, A[\alpha] \rightarrow B}{\Gamma, (\exists \beta \leq \tau) A[\beta] \rightarrow B}$$

Further, in our restricted logical setting, we adopt the following rule of term substitution:

(S0) $$\frac{\Gamma[\alpha] \rightarrow A[\alpha]}{\Gamma[\tau] \rightarrow A[\tau]}$$

The non-logical axioms of FEA state the usual defining equations for the function symbols of the language $\mathcal{L}_0$, see, e.g., Ferreira [45] for similar axioms. Finally, we have the schematic induction rule formulated for a free predicate P as follows:

(Ind) $$\frac{\Gamma \rightarrow P(\epsilon) \qquad \Gamma, P(\alpha) \rightarrow P(\mathsf{S}_i(\alpha)) \quad (i = 0, 1)}{\Gamma \rightarrow P(\alpha)}$$

In the various unfolding systems of FEA introduced below, we will be able to substitute an arbitrary formula for the free predicate letter P.

5.3 The operational unfolding $\mathcal{U}_0(\mathsf{FEA})$

In this section we are going to introduce the *operational unfolding* $\mathcal{U}_0(\mathsf{FEA})$ of FEA. It tells us which operations from and to individuals, and which principles concerning them, ought to be accepted if one has accepted FEA.

In the operational unfolding, we make these commitments explicit by extending FEA by a partial combinatory algebra. Since it represents any recursion principle and thus any recursive function by suitable terms, it is expressive enough to reflect any ontological commitment we want to reason about. Using the notion of *provable totality*, we single out those functions and recursion principles we are actually committed to by accepting FEA.

5.3.1 The language $\mathcal{L}_1$

The language $\mathcal{L}_1$ is an expansion of the language $\mathcal{L}_0$ including new constants k, s, p, p_0, p_1, d, tt, ff, e, ϵ, s_0, s_1, p_W, $\mathsf{c}_\subseteq$, $*$, $\times$, and an additional countably infinite set of variables $x_0, x_1, \ldots$.[3] The new variables are supposed to range over the universe of operations and are usually denoted by $a, b, c, x, y, z, \ldots$. The $\mathcal{L}_1$ terms $(r, s, t, \ldots)$ are inductively generated from variables and constants of $\mathcal{L}_0$ and $\mathcal{L}_1$ by means of the function symbols of FEA and the application operator $\cdot$. We use the usual abbreviations for applicative terms, see page 10. We have $(s = t)$, $s{\downarrow}$ and $P_i(\vec{s})$ for $i \in \mathbb{N}$ as atoms of $\mathcal{L}_1$. The formulas $(A, B, C, \ldots)$ are built from the atoms as before using $\vee, \wedge$ and the bounded existential quantifier, where as above the bounding term is a term of $\mathcal{L}_0$ not containing the bound variable. For s a term of $\mathcal{L}_1 \setminus \mathcal{L}_0$ we write $s \leq \tau$ for $(\exists \beta \leq \tau)(s = \beta)$.

Let us compare $\mathcal{L}_1$ to our applicative base language L. $\mathcal{L}_1$ has two sorts of variables, and extends L by the function constants of $\mathcal{L}_0$. $\mathcal{L}_1$ contains a free predicate P but no unbounded quantifiers. The applicative constants of both languages correspond to each other with exception of the constants responsible for case distinction (d, tt, ff, or $\mathsf{d_W}$, respectively). The constants of $\mathcal{L}_1$ allow to divide case distinctions into checking equality and switching between cases.

5.3.2 Axioms and rules of $\mathcal{U}_0(\mathsf{FEA})$

The operational unfolding $\mathcal{U}_0(\mathsf{FEA})$ is formulated as a sequent calculus. $\emptyset \to A$ will just be displayed as A. Apart from the axioms for FEA, $\mathcal{U}_0(\mathsf{FEA})$ comprises the following axioms and rules of inference.

I. Applicative counterpart of the initial functions.

(1) $\mathsf{s}_i\alpha = \mathsf{S}_i(\alpha)$, $\quad \mathsf{p_W}\alpha = \mathsf{Pd}(\alpha)$,

(2) $*\alpha\beta = \alpha \circledast \beta$, $\quad \times\alpha\beta = \alpha \boxtimes \beta$, $\quad \mathsf{c}_{\subseteq}\alpha\beta = \alpha \subseteq \beta$.

II. Partial combinatory algebra, pairing, definition by cases.

(3) $\mathsf{k}ab = a$,

(4) $\mathsf{s}ab{\downarrow}$, $\quad \mathsf{s}abc \simeq ac(bc)$,

[3]These variables are syntactically different from the $\mathcal{L}_0$ variables $\alpha_0, \alpha_1, \ldots$.

(5) $\mathsf{p}_0\langle a, b\rangle = a, \quad \mathsf{p}_1\langle a, b\rangle = b,$

(6) $\mathsf{d}ab\,\mathsf{tt} = a, \quad \mathsf{d}ab\,\mathsf{ff} = b.$

III. Equality on the binary words.

(7) $\mathsf{e}\alpha\beta = \mathsf{tt} \vee \mathsf{e}\alpha\beta = \mathsf{ff},$

(8) $\mathsf{e}\alpha\beta = \mathsf{tt} \leftrightarrow \alpha = \beta.$[4]

The operational unfolding of FEA includes the rules of inference of FEA (extended to the new language). In addition, in analogy to the rule ($\mathsf{S0}$), we have the following new substitution rule for terms of $\mathcal{L}_1$:

($\mathsf{S1}$)
$$\frac{\Gamma[u] \rightarrow A[u]}{\Gamma[t], t{\downarrow} \rightarrow A[t]}$$

The next useful substitution rule ($\mathsf{S2}$) can be derived easily from the other axioms and rules. It tells us that bounded terms can be substituted for word variables:[5]

($\mathsf{S2}$)
$$\frac{\Gamma[\alpha] \rightarrow A[\alpha] \quad \Gamma[t] \rightarrow t \leq \tau}{\Gamma[t] \rightarrow A[t]}$$

Finally, $\mathcal{U}_0(\mathsf{FEA})$ includes the generalized substitution rule for derived rules of inference as it is developed in Feferman and Strahm [44]. Towards a more compact notation, let use write $\Sigma_1, \Sigma_2, \dots, \Sigma_n \Rightarrow \Sigma$ to denote a rule of inference with premises $\Sigma_1, \dots, \Sigma_n$ and conclusion Σ. We let $A[\vec{B}/\vec{P}]$ denote the formula $A[\vec{P}]$ with each subformula $P_i(\vec{t})$ replaced by $\vec{t}{\downarrow} \wedge B_i[\vec{t}]$, where the length of $\vec{t}$ equals the arity of P_i. The generalized substitution rule ($\mathsf{S3}$) can now be described as follows: Assume that the rule of inference $\Sigma_1, \Sigma_2, \dots, \Sigma_n \Rightarrow \Sigma$ is derivable from the axioms and rules at hand. Then we can adjoin an arbitrary substitution instance

($\mathsf{S3}$)
$$\Sigma_1[\vec{B}/\vec{P}], \dots, \Sigma_n[\vec{B}/\vec{P}] \Rightarrow \Sigma[\vec{B}/\vec{P}]$$

[4]To be precise, this equivalence is a shorthand for the two sequents $\mathsf{e}\alpha\beta = \mathsf{tt} \rightarrow \alpha = \beta$ and $\alpha = \beta \rightarrow \mathsf{e}\alpha\beta = \mathsf{tt}$.

[5]Note that for an $A[\alpha]$ with α occurring in a bound and a term $t \in \mathcal{L}_1 \setminus \mathcal{L}_0$, the rule ($\mathsf{S2}$) cannot be derived because then $A[t]$ is not a formula.

as new rule of inference to our system. Here $\vec{P}$ and $\vec{B}$ are finite sequences of free predicates and $\mathcal{L}_1$ formulas, respectively. Note that the notion of *derivability of a rule of inference* is dynamic as one unfolds a given system. Clearly, using the generalized substitution rule, the induction rule in its usual form can be derived for an arbitrary $A \in \mathcal{L}_1$:

$$\frac{\Gamma \to A[\epsilon] \qquad \Gamma, A[\alpha] \to A[\mathsf{S}_i(\alpha)] \quad (i = 0, 1)}{\Gamma \to A[\alpha]}$$

Moreover, the usual substitution rule for sequents, $\Sigma[\vec{P}] \Rightarrow \Sigma[\vec{B}/\vec{P}]$ can be obtained as an admissible rule of inference. This ends the description of the operational unfolding $\mathcal{U}_0(\mathsf{FEA})$ of FEA. Next we want to show that the polynomial time computable functions can be proved to be total in $\mathcal{U}_0(\mathsf{FEA})$.

Lemma 89 *The polynomial time computable functions are provably total in the operational unfolding* $\mathcal{U}_0(\mathsf{FEA})$.

Proof. We use Cobham's characterization of the polynomial time computable functions (cf. [23, 20]). First of all, the projections represented using lambda abstraction and the other initial functions of Cobham's algebra are obviously provably total. Closure of the provably total functions under composition is established by making use of the substitution rules ($\mathsf{S1}$) and ($\mathsf{S2}$) as well as the fact that the $\mathcal{L}_0$ functions are provably monotone. In order to show closure under bounded recursion, assume that F is defined by bounded recursion with initial function G and step function H, where τ is the corresponding bounding polynomial.[6] By the induction hypothesis, G and H are provably total via suitable $\mathcal{L}_1$ terms t_G and t_H. Using the recursion or fixed point theorem of the partial combinatory algebra, we find an $\mathcal{L}_1$ term t_F which provably in $\mathcal{U}_0(\mathsf{FEA})$ satisfies the following recursion equations for $i = 0, 1$:

$$\begin{aligned} t_F(\vec{\alpha}, \epsilon) &\simeq t_G(\vec{\alpha}) \mid \tau[\vec{\alpha}, \epsilon], \\ t_F(\vec{\alpha}, \mathsf{s}_i(\beta)) &\simeq t_H(t_F(\vec{\alpha}, \beta), \vec{\alpha}, \beta) \mid \tau[\vec{\alpha}, \mathsf{s}_i(\beta)] \end{aligned}$$

[6]We can assume that only functions built from concatenation and multiplication are permissible bounds for the recursion.

Here $|$ is the usual truncation operation such that $\alpha|\beta$ is α if $\alpha \leq \beta$ and β otherwise. Now fix $\vec{\alpha}$ and let $A[\beta]$ be the formula $t_F(\vec{\alpha}, \beta) \leq \tau[\vec{\alpha}, \beta]$ [7] and simply show $A[\beta]$ by induction on β. Thus F is provably total in $\mathcal{U}_0(\mathsf{FEA})$ which concludes the proof of the lower bound lemma. $\square$

5.4 The full predicate unfolding $\mathcal{U}(\mathsf{FEA})$

In this section we define the full predicate unfolding $\mathcal{U}(\mathsf{FEA})$ of FEA. It tells us, in addition, which predicates and operations on predicates ought to be accepted if one has accepted FEA. By accepting $\mathcal{U}_0(\mathsf{FEA})$ one implicitly accepts an equality predicate and operations on predicates corresponding to the logical operations of $\mathcal{U}_0(\mathsf{FEA})$. Finally, we may accept the principle of forming the predicate for the disjoint union of a (bounded) sequence of predicates given by an operation.

As before the equality predicate and the above-mentioned operations will be given as elements of an underlying combinatory algebra which is extended by a binary relation $\in$ for elementship, so predicates are represented via classifications in the sense of explicit mathematics. We additionally use a relation $\Re$ to single out the operations representing predicates one is committed to by accepting FEA.

The language $\mathcal{L}_2$ of $\mathcal{U}(\mathsf{FEA})$ is an extension of $\mathcal{L}_1$ by new individual constants id (identity), inv (inverse image), int (intersection), un (union), leq (bounded existential quantifier), and j (bounded disjoint unions); further new constants are $\mathsf{p}_0, \mathsf{p}_1, \ldots$ which are combinatorial representations of free predicates. Finally, $\mathcal{L}_2$ has a new unary relation symbol $\Re$, and a binary relation symbol $\in$. The terms of $\mathcal{L}_2$ are generated as before but now taking into account the new constants. The formulas of $\mathcal{L}_2$ extend the formulas of $\mathcal{L}_1$ by allowing new atomic formulas of the form $\Re(t)$ and $s \in t$.

The language $\mathcal{L}_2$ corresponds to the language of the first order formulation of explicit mathematics introduced on page 112. Note that $\mathcal{L}_2$ does not

[7] Recall that by expanding the definition of the $\leq$ relation, the formula $A[\beta]$ stands for the assertion $(\exists \gamma \leq \tau[\vec{\alpha}, \beta])(t_F(\vec{\alpha}, \beta) = \gamma)$.

contain type constructors corresponding to unbounded quantifiers (all, dom). A further difference are the constructors for the free predicates present in $\mathcal{L}_2$.

$\mathcal{U}(\mathsf{FEA})$ is formulated as a sequent calculus, and it extends $\mathcal{U}_0(\mathsf{FEA})$ by the following axioms about predicates. For the constructors $\mathsf{id}, \mathsf{inv}, \mathsf{int}, \mathsf{un}$ these are just sequent style reformulations of the axioms presented on page 108.

I. Identity predicate

(1) $\Re(\mathsf{id})$,

(2) $x \in \mathsf{id} \to \mathsf{p}_0 x = \mathsf{p}_1 x \wedge x = \langle \mathsf{p}_0 x, \mathsf{p}_1 x \rangle$,

(3) $\mathsf{p}_0 x = \mathsf{p}_1 x, x = \langle \mathsf{p}_0 x, \mathsf{p}_1 x \rangle \to x \in \mathsf{id}$.

II. Inverse image predicates

(4) $\Re(a) \to \Re(\mathsf{inv}(f, a))$,

(5) $\Re(a), x \in \mathsf{inv}(f, a) \to fx \in a$,

(6) $\Re(a), fx \in a \to x \in \mathsf{inv}(f, a)$.

III. Intersection and union

(7) $\Re(a), \Re(b) \to \Re(\mathsf{int}(a, b))$,

(8) $\Re(a), \Re(b), x \in \mathsf{int}(a, b) \to x \in a \wedge x \in b$,

(9) $\Re(a), \Re(b), x \in a, x \in b \to x \in \mathsf{int}(a, b)$,

(10) $\Re(a), \Re(b) \to \Re(\mathsf{un}(a, b))$,

(11) $\Re(a), \Re(b), x \in \mathsf{un}(a, b) \to x \in a \vee x \in b$,

(12) $\Re(a), \Re(b), x \in a \vee x \in b \to x \in \mathsf{un}(a, b)$.

IV. Bounded existential quantification

(13) $\Re(a) \to \Re(\mathsf{leq} a)$,

(14) $\Re(a), \langle y, \alpha \rangle \in \mathsf{leq}(a) \to (\exists \beta \leq \alpha)(\langle y, \beta \rangle \in a)$,

(15) $\Re(a), (\exists \beta \leq \alpha)(\langle y, \beta \rangle \in a) \to \langle y, \alpha \rangle \in \mathsf{leq}(a)$.

V. Free predicates [8]

(16) $\Re(\mathsf{p}_i)$,

(17) $\vec{x} \in \mathsf{p}_i \to P_i(\vec{x}), \quad P_i(\vec{x}) \to (\vec{x}) \in \mathsf{p}_i$.

Further, the full unfolding $\mathcal{U}(\mathsf{FEA})$ includes axioms stating that a bounded sequence of predicates determines the predicate of the disjoint union of this sequence. We use the following three rules to axiomatise the join predicates in our restricted logical setting.

VI. Join rules [9]

$$\frac{\Gamma, \beta \leq \alpha \to \Re(f\beta)}{\Gamma \to \Re(\mathsf{j}(f, \alpha))} \tag{18}$$

$$\frac{\Gamma, \beta \leq \alpha \to \Re(f\beta)}{\Gamma, x \in \mathsf{j}(f, \alpha) \to x = \langle \mathsf{p_0}x, \mathsf{p_1}x\rangle \wedge \mathsf{p_0}x \leq \alpha \wedge \mathsf{p_1}x \in f(\mathsf{p_0}x)} \tag{19}$$

$$\frac{\Gamma, \beta \leq \alpha \to \Re(f\beta)}{\Gamma, x = \langle \mathsf{p_0}x, \mathsf{p_1}x\rangle, \mathsf{p_0}x \leq \alpha, \mathsf{p_1}x \in f(\mathsf{p_0}x) \to x \in \mathsf{j}(f, \alpha)} \tag{20}$$

The rules of inference of $\mathcal{U}_0(\mathsf{FEA})$ are also available in $\mathcal{U}(\mathsf{FEA})$. In particular, $\mathcal{U}(\mathsf{FEA})$ contains the generalized substitution rule (S3): the formulas $\vec{B}$ to be substituted for $\vec{P}$ are now in the language of $\mathcal{L}_2$; the rule in the premise of (S3), however, is required to be in the language $\mathcal{L}_1$[10]. This concludes the description of the predicate unfolding $\mathcal{U}(\mathsf{FEA})$ of FEA.

5.5 The truth unfolding $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$

In this section we describe an alternative way to define the full unfolding of FEA. The truth unfolding $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ of FEA makes use of a truth predicate T which reflects the logical operations of FEA in a natural and direct way. We will see that the full predicate unfolding $\mathcal{U}(\mathsf{FEA})$ is directly contained in $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$.[11]

[8] We write $\vec{x} \in \mathsf{p}_i$ for $\langle x_1, \cdots, x_n\rangle \in \mathsf{p}_i$.

[9] In the formulation of these rules, it is assumed that β does not occur in Γ.

[10] This last restriction is imposed since predicates may depend on $\vec{P}$.

[11] We note that in Feferman's original definition of unfolding in [38], a truth predicate is used in order to describe the full unfolding of a schematic system.

As in the last section, we want to make the commitment to the logical operations of FEA explicit. This is done by introducing a truth predicate in the style of the previous chapters. The language $\mathcal{L}_\mathsf{T}$ of $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ extends $\mathcal{L}_1$ by new individual constants $\dot{=}$, $\dot{\wedge}$, $\dot{\vee}$, $\dot{\exists}$, as well as constants $\mathsf{p}_0, \mathsf{p}_1, \ldots$. In addition, $\mathcal{L}_\mathsf{T}$ includes a new unary relation symbol T. The terms and formulas of $\mathcal{L}_\mathsf{T}$ are defined in the expected manner.

$\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ is formulated as a sequent calculus, and extends $\mathcal{U}_0(\mathsf{FEA})$ by the following axioms about the truth predicate T:

$(\dot{=})$	$\mathsf{T}(x \mathbin{\dot{=}} y)$	$\leftrightarrow$	$x = y$
$(\dot{\wedge})$	$\mathsf{T}(x \mathbin{\dot{\wedge}} y)$	$\leftrightarrow$	$\mathsf{T}(x) \wedge \mathsf{T}(y)$
$(\dot{\vee})$	$\mathsf{T}(x \mathbin{\dot{\vee}} y)$	$\leftrightarrow$	$\mathsf{T}(x) \vee \mathsf{T}(y)$
$(\dot{\exists})$	$\mathsf{T}(\dot{\exists}\alpha x)$	$\leftrightarrow$	$(\exists \beta \leq \alpha)\mathsf{T}(x\beta)$
(p_i)	$\mathsf{T}(\mathsf{p}_i(\vec{x}))$	$\leftrightarrow$	$P_i(\vec{x})$

As in definition 22, we can assign an $\mathcal{L}_\mathsf{T}$ code $[A]$ to each $\mathcal{L}_\mathsf{T}$ formula A in a natural way. The following lemma is proved by a trivial induction on the complexity of formulas.

Lemma 90 (Tarski biconditionals) *Let A be a $\mathcal{L}_\mathsf{T}$ formula. Then we have*

$$\mathcal{U}_\mathsf{T}(\mathsf{FEA}) \vdash A \leftrightarrow \mathsf{T}([A])$$

This lemma shows that in our weak setting, full Tarski biconditionals can be achieved without having to type the truth predicate. Of course, this is due to the fact that negation is only present at the level of sequents.

We close this section by noting that the generalized substitution rule $(\mathsf{S3})$ can be stated in a somewhat more general form for $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$. Recall that in $\mathcal{U}(\mathsf{FEA})$, the rule in the premise of $(\mathsf{S3})$ is required to be in $\mathcal{L}_1$. Due to the fact that each $\mathcal{L}_\mathsf{T}$ formula can be represented by a term, we can allow rules in $\mathcal{L}_\mathsf{T}$ in the premise of the generalized substitution rule, as long as we substitute formulas and associated terms for the predicates P_i and constants p_i simultaneously [12]. In the following we denote by $\Sigma[\vec{B}/\vec{P}; \vec{t}/\vec{\mathsf{p}}]$

[12]The soundness of this rule crucially depends on the fact that the only axioms available for p_i are its truth-biconditionals, and general combinatorial axioms.

the simultaneous substitution of the predicates $\vec{P}$ by the formulas $\vec{B}$ and of the constants $\vec{\mathsf{p}}$ by the $\mathcal{L}_\mathsf{T}$ terms $\vec{t}$. The generalized substitution rule for $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ can now be stated as follows. Assume that the rule $\Sigma_1, \dots, \Sigma_n \Rightarrow \Sigma$ is derivable with the axioms and rules at hand. Assume further that the terms $\vec{t_B}$ correspond to the $\mathcal{L}_\mathsf{T}$ formulas $\vec{B}$ according to the lemma above. Then we can adjoin the rule

$$\Sigma_1[\vec{B}/\vec{P}; \vec{t_B}/\vec{\mathsf{p}}], \dots, \Sigma_n[\vec{B}/\vec{P}; \vec{t_B}/\vec{\mathsf{p}}] \Rightarrow \Sigma[\vec{B}/\vec{P}; \vec{t_B}/\vec{\mathsf{p}}]$$

as a new rule of inference to our unfolding system $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$. This concludes the description of $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$.

It is easy to see that the full predicate unfolding $\mathcal{U}(\mathsf{FEA})$ is contained in the truth unfolding $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$. The argument proceeds along the same line as the embeddings presented in chapter 4, and is also very similar to the embedding of $\mathcal{U}(\mathsf{FEA})$ into T_PT which will be described in some detail in the next section.

As in chapter 4, the reverse embedding does not seem to be possible without additional principles, and even then, only a levelled version of $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ could be embedded.

5.6 Proof-theoretical analysis

In this section we will find a suitable upper bound for $\mathcal{U}(\mathsf{FEA})$ and $\mathcal{U}_\mathsf{T}(\mathsf{FEA})$ thus showing that their provably total functions are indeed computable in polynomial time. We will obtain the upper bound via an embedding of a total version of $\mathcal{U}(\mathsf{FEA})$ into T_PT; to be precise, we consider a slight (conservative) extension T^*_PT of T_PT which facilitates the treatment of the generalized substitution rule.

The language of T^*_PT extends the one of T_PT by the predicates $P_0, P_1, \dots$ and the constants $\mathsf{p}_0, \mathsf{p}_1, \dots$. It contains the additional axiom $\mathsf{T}(\mathsf{p}_i\vec{x}) \leftrightarrow P_i(\vec{x})$ for every $i \in \mathbb{N}$. Since no other axioms for the P predicates and the p constants are present, T^*_PT is clearly a conservative extension of T_PT.

Next we describe a direct embedding of $\mathcal{U}(\mathsf{FEA})$ into T^*_PT which resembles the one described in chapter 4. Nevertheless, we have to consider some peculiarities of our system: we take special care of the FEA function constants which are not present in the language of $\mathsf{T}^*_{\mathsf{PT}}$ and map the two kinds of variables to disjoint sets of $\mathsf{T}^*_{\mathsf{PT}}$ variables.

Definition 91 (Translation * of $\mathcal{L}_2$ terms) *The translation of $\mathcal{L}_2$ terms is given inductively on their complexity.*

- *Let c be an applicative constant. Then $c^* \equiv c$.*
- *Let α_i be an $\mathcal{L}_0$ variable. Then $\alpha_i^* \equiv x_{2i}$.*
- *Let x_i be a variable of $\mathcal{L}_1 \setminus \mathcal{L}_0$. Then $x_i^* \equiv x_{2i+1}$.*
- $\mathsf{leq}^* \equiv \lambda a.\lambda z.z \doteq \langle \mathsf{p}_0 z, \mathsf{p}_1 z\rangle \mathbin{\dot\wedge} \dot\exists \lambda y.\dot{\mathsf{W}}(\mathsf{p}_1 z)y \mathbin{\dot\wedge} a\langle \mathsf{p}_0 z, y\rangle$
- $\mathsf{id}^* \equiv \lambda z.z = \langle \mathsf{p}_0 z, \mathsf{p}_1 z\rangle \mathbin{\dot\wedge} \mathsf{p}_0 z \doteq \mathsf{p}_1 z$
- $\mathsf{int}^* \equiv \lambda a.\lambda b.\lambda z.\ az \mathbin{\dot\wedge} bz$
- $\mathsf{un}^* \equiv \lambda a.\lambda b.\lambda z.\ az \mathbin{\dot\vee} bz$
- $\mathsf{inv}^* \equiv \lambda f.\lambda a.\lambda z.\ a(fz)$
- $\mathsf{j}^* \equiv \lambda f.\lambda a.\lambda z.z = \langle \mathsf{p}_0 z, \mathsf{p}_1 z\rangle \mathbin{\dot\wedge} \dot{\mathsf{W}}a(\mathsf{p}_0 z) \mathbin{\dot\wedge} f(\mathsf{p}_0 z)(\mathsf{p}_1 z)$
- $\mathsf{p}_i^* \equiv \mathsf{p}_i$
- *Let t be $s_0 s_1$. Then $t^* \equiv s_0^* s_1^*$.*
- *Let G be an n-ary $\mathcal{L}_0$ function symbol, g_{App} its applicative analogue, and $\vec{t}$ a sequence of terms of suitable arity. Then $G(\vec{t})^* \equiv g_{App}\vec{t^*}$.*

For the translation of $\mathcal{L}_2$ formulas, we interpret elementship using the truth predicate as usual and trivialize the relation $\Re$.

Definition 92 (Translation * of $\mathcal{L}_2$ formulas) *The translation of $\mathcal{L}_2$ formulas is given inductively on their complexity.*

- $\Re(s)^* \equiv 0 = 0$
- $(s = t)^* \equiv s^* = t^*$

- $(s \in t)^* \equiv \mathsf{T}(t^* s^*)$
- $(\exists \alpha \leq \tau) A[\alpha]^* \equiv (\exists \alpha^* \leq_{\mathsf{W}} \tau^*) A^*[\alpha^*]$
- *The translation commutes with the connectives $\wedge$ and $\vee$.*

The translation * is extended in the obvious way to sequences and sequents of $\mathcal{L}_2$ formulas. Further, for the statement of the embedding theorem below, the following notation is handy.

Definition 93 *Let $\diamond$ be an $\mathcal{L}_2$ term, formula or sequence of formulas. Then $\vec{y}(\diamond) \in \mathsf{W}$ denotes the sequence $x_{n_0} \in \mathsf{W}, \ldots, x_{n_m} \in \mathsf{W}$ where the x_{n_i} enumerate the variables with even subscripts occurring freely in $\diamond^*$.*

The next two lemmas will be used in the proof of the embedding theorem below. Lemma 94 is proved by a trivial induction on the complexity of the FEA term. Lemma 95 is proved by induction on the complexity of A. For the case where A is of the form $(\exists \alpha \leq \tau) B[\alpha]$, we use lemma 94.

Lemma 94 *Let τ be an $\mathcal{L}_0$ term. Then we have*

$$\mathsf{T}^*_{\mathsf{PT}} \vdash \vec{y}(\tau) \in \mathsf{W} \rightarrow \tau^* \in \mathsf{W}.$$

Lemma 95 *Let A be an $\mathcal{L}_2$ formula. Then we have*

$$\mathsf{T}^*_{\mathsf{PT}} \vdash \vec{y}(A) \in \mathsf{W} \rightarrow \mathsf{T}(\ulcorner A^* \urcorner) \leftrightarrow A^*.$$

We are now ready to state the main embedding lemma of $\mathcal{U}(\mathsf{FEA})$ into $\mathsf{T}^*_{\mathsf{PT}}$ and sketch its proof.

Lemma 96 (Embedding lemma) *Assume $\mathcal{U}(\mathsf{FEA}) \vdash \Gamma \rightarrow A$. Then we have*

$$\mathsf{T}^*_{\mathsf{PT}} \vdash \vec{y}(\Gamma, A) \in \mathsf{W}, \Gamma^* \rightarrow A^*.$$

Proof.(Sketch) In order to prove the lemma, one shows a stronger assertion, namely that the * translation of each derivable rule of $\mathcal{U}(\mathsf{FEA})$ is also derivable in $\mathsf{T}^*_{\mathsf{PT}}$. Let us exemplary discuss some crucial examples. First, let us look at * translations of axioms of $\mathcal{U}(\mathsf{FEA})$ and distinguish the following cases:

(i) The translations of the axioms about the (word) function symbols of $\mathcal{L}_0$ hold, because the $\mathcal{L}_0$ variables are assumed to range over words;

(ii) The translations of the axioms about the applicative combinators clearly hold;

(iii) The translations of the axioms about the correspondence between the function symbols and the applicative constants follow directly from the definition of the translation;

(iv) The translations of the axioms about the predicate constructors hold because of their translation by suitable elementhood conditions and because of the trivial interpretation of the relation $\Re$;

(v) The translations of the axioms $P_i(\vec{x}) \leftrightarrow \vec{x} \in \mathsf{p}_i$ clearly hold.

Towards the treatment of the generalized substitution rule, assume that the rule with premises $\Gamma_i[\vec{P}] \rightarrow A_i[\vec{P}]$ for $1 \leq i \leq m$ and conclusion $\Gamma[\vec{P}] \rightarrow A[\vec{P}]$ is derivable in $\mathcal{U}(\mathsf{FEA})$. Let us look at the * translation of the proof for derivability which is a proof of derivability in $\mathsf{T}^*_{\mathsf{PT}}$ by induction hypothesis. It can be easily seen that for each sequence of formulas $\vec{B} \in \mathcal{L}_2$ we still have a proof if we replace each occurrence of P_i by B^*_i and each occurrence of p_i by $[B^*_i]$ and add $\vec{y}(\vec{B}) \in \mathsf{W}$ to each antecedent. Here, we use lemma 95 to justify induction and the substituted P biconditionals. Thus the * translation of the rule with conclusion $\Gamma[\vec{B}] \rightarrow A[\vec{B}]$ and premises $\Gamma_i[\vec{B}] \rightarrow A_i[\vec{B}]$ for $1 \leq i \leq m$ is derivable in $\mathsf{T}^*_{\mathsf{PT}}$ as desired. This ends the treatment of the generalized substitution rule and hence the proof sketch of the embedding lemma. $\square$

The embedding lemma immediately implies that each function which is provably total in $\mathcal{U}(\mathsf{FEA})$ is also provably total in $\mathsf{T}^*_{\mathsf{PT}}$ in the usual sense. Since $\mathsf{T}^*_{\mathsf{PT}}$ is conservative over T_{PT} this delivers the desired upper bound for the unfoldings $\mathcal{U}_0(\mathsf{FEA})$ and $\mathcal{U}(\mathsf{FEA})$. Together with Lemma 89, we obtain sharp proof theoretic bounds.

Theorem 97 *The provably total functions of* $\mathcal{U}_0(\mathsf{FEA})$ *and* $\mathcal{U}(\mathsf{FEA})$ *are exactly the polynomial time computable functions.*

An embedding of $\mathcal{U}_{\mathsf{T}}(\mathsf{FEA})$ into $\mathsf{T}^*_{\mathsf{PT}}$ can be found in a very similar way as for $\mathcal{U}(\mathsf{FEA})$. Just interpret the constants $\dot{=}$, $\dot{\wedge}$ and $\dot{\vee}$ as themselves and $\dot{\exists}$ as $\lambda y.\lambda z.\dot{\exists}\lambda x.\dot{\mathsf{W}}yx \mathbin{\dot{\wedge}} zx$. Thus we obtain the following theorem.

Theorem 98 *The provably total functions of $\mathcal{U}_{\mathsf{T}}(\mathsf{FEA})$ are exactly the polynomial time computable functions.*

This concludes the computation of the upper bounds and hence the proof-theoretic analysis of our various unfolding systems.

The enbedding given above relies on the standard technique of embedding systems of explicit mathematics into theories of truth. Let us now introduce an alternative, and simpler embedding of $\mathcal{U}(\mathsf{FEA})$ into $\mathsf{T}^*_{\mathsf{PT}}$ which relies on the fact that $\mathsf{T}^*_{\mathsf{PT}}$ proves the Tarski biconditional for each of its formulas. The idea is to produce for each formula $\mathcal{U}(\mathsf{FEA})$ a corresponding code by a translation ^{c}.

Definition 99 (Translation ^{c} of $\mathcal{L}_2$ formulas) *The translation of $\mathcal{L}_2$ formulas is given inductively on their complexity. Let the translation ^{c} for terms be defined as the translation * in definition 91.*

- $\Re(s)^c \equiv 0 \mathbin{\dot{=}} 0$
- $(s = t)^c \equiv s^c \mathbin{\dot{=}} t^c$
- $(s \in t)^c \equiv t^c s^c$
- $P_i(t)^c \equiv \mathsf{p}_i t^c$
- $(\exists \alpha \leq t)A[\alpha]^c \equiv \dot{\exists}\lambda\alpha^c.\dot{\mathsf{W}}\tau^c\alpha^c \mathbin{\dot{\wedge}} A^c[\alpha^c]$
- $(A \wedge B)^c \equiv A^c \mathbin{\dot{\wedge}} B^c$
- $(A \vee B)^c \equiv A^c \mathbin{\dot{\vee}} B^c$

Then, the following embedding claim is proved as the previous one.

Lemma 100 (Embedding lemma) *Assume $\mathcal{U}(\mathsf{FEA}) \vdash A_1, \cdots, A_n \to D$. Then we have*

$$\mathsf{T}^*_{\mathsf{PT}} \vdash \vec{y}(A_1, \cdots, A_n, D) \in \mathsf{W}, \mathsf{T}(A_1^c), \cdots, \mathsf{T}(A_n^c) \to \mathsf{T}(D^c)$$

Chapter 6

Applicative theories for logarithmic complexity classes

The contents of this chapter have been submitted for publication as [28].

6.1 Introduction

In this chapter, we present applicative theories of words corresponding to weak, and especially logarithmic, complexity classes. The theories for the logarithmic hierarchy and alternating logarithmic time formalise function algebras with concatenation recursion as main principle. We present two theories for logarithmic space where the first formalises a new two-sorted algebra which is very similar to Cook and Bellantoni's famous two-sorted algebra B for polynomial time [7]. The second theory describes logarithmic space by justifying concatenation - and sharply bounded recursion. All theories contain the predicates W representing words, and V representing temporary inaccessible words. They are inspired by Cantini's theories [17] formalising B.

There are many examples of logical theories corresponding to complexity classes given by function algebras. Research on this subject was started by Buss in [11] where theories of bounded arithmetic are introduced for the subclasses of the polynomial hierarchy. Further theories of bounded arithmetic have been introduced by Clote and Takeuti in [22] amongst others for the logarithmic hierarchy, alternating logarithmic time, logarithmic space, and various circuit complexity classes. Weak arithmetic *second-order* theories corresponding to various complexity classes, analysed by various researchers, are collected in Cook and Nguyen's [24]. *Applicative* theories corresponding to complexity classes have been introduced e.g. by Strahm [78, 79] for linear and polynomial time - and space classes, by Cantini for polynomial time [17], or by Kahle and Oitavem [62] for the polynomial hierarchy.

In contrast to corresponding theories of bounded arithmetic, applicative theories allow to prove the totality of the functions of their complexity class without coding. This makes the lower bound proofs typically easier and more transparent. For an overview of weak applicative theories, we recommend Strahm's [81].

We present applicative theories for various logarithmic complexity classes. The theories are formulated over a base theory of words and contain induction principles to justify the suitable forms of recursion. As Cantini's mentioned system they contain two predicates W and V. In contrast to Cantini's theory, we have to be more restrictive about the permitted operations on elements of V to achieve logarithmic strength: We cannot allow case distinction for elements of V of the following form.

$$\mathsf{case}(;a,b,c) := \begin{cases} b, & \text{if } mod_2(a) = 0 \\ c, & \text{else} \end{cases}$$

Accordingly, the intended meaning of being an element of V differs from Cantini's theory, where elements of V just have a different role in the induction scheme. For the weaker theories, $t \in \mathsf{V}$ informally means that t is a temporary inaccessible word, e.g. it can only be the input of functions not requiring to read off any of its bits. This property is fulfilled e.g. for the successor - and predecessor functions, and is designed to describe the role intermediate values play during concatenation recursion. For the stronger theory of logspace strength, we allow at least to determine, whether a $t \in \mathsf{V}$ equals the empty word ϵ by the following case distinction given for safe inputs.

$$\mathsf{case}(;a,b,c) := \begin{cases} b, & \text{if } a = \epsilon \\ c, & \text{else} \end{cases}$$

If we compare our two-sorted theories with Cantini's, the main difference is that we do not only forbid elements of V to control recursion, but also during recursion restrict the way they can be used heavily.

Let us summarize the content of this chapter. In section two, we develop word-versions for number function algebras developed by Clote in [20] for the complexity classes LH, $\mathsf{ALOGTIME}$ and P [1]. It is not enough to just reformulate Clote's algebras because the successor function on numbers, in contrast to the one on words, does not always increase its argument; we have $\mathsf{S}_0(0) = 0$. For this reason concatenation recursion on notation is a stronger principle on numbers than on words. To achieve the equivalence of the number - and word function algebras, we strengthen the word algebras by an additional initial function which erases leading zeros.

In the third section, we design applicative theories justifying concatenation recursion. We present three applicative theories of words in detail that correspond to implicit characterisations of complexity classes whose main principle is concatenation recursion or an extension thereof. Their provably total functions are the elements of the logarithmic hierarchy for LogT, alternating logarithmic time for AlogT, and polynomial time for PT.

The theories LogT, AlogT, and PT are introduced simultaneously, since they only differ very slightly in their induction schemes. Their induction formulas have two free variables, as in Kahle and Oitavem's [61], which allows to express that $F(\mathsf{s}_i w, \vec{z})$ is a successor of $F(w, \vec{z})$ for a function F defined by concatenation recursion.

The lower bound of the theories LogT, AlogT, and PT is established by proving totality for all elements of the word function algebras developed in section 6.2. Because of the computational completeness of the underlying combinatory algebra, we can directly produce terms representing these functions without any coding. Then, we prove by an easy induction that these terms represent total functions. In section 6.3.5, we prove their upper bound using

[1] LH denotes the set of functions computable on an alternating Turing machine with random access in logarithmic time with a bounded number of alternations. $\mathsf{ALOGTIME}$ extends LH by allowing an arbitrary number of alternations. Finally, P denotes the polytime functions.

a modification of Strahm's realisation approach [79]. This delivers an exact characterisation of LogT, AlogT and PT in terms of provably total functions.

In section four, we present a new two-sorted algebra $\mathfrak{LS}$ for logarithmic space, which is just Cook and Bellantoni's B with a weakened case distinction and an additional initial function yielding the length of its input. In contrast to Bellantoni's description of logarithmic space as safe *unary* algebra, $\mathfrak{LS}$ contains also the fast growing members of logspace. The prize one has to pay is the addition of the initial function abs.

In section 5, we formalise $\mathfrak{LS}$ and Clote's algebra for logspace containing sharply bounded recursion [20], and obtain new two-sorted applicative theories of the same strength. These theories contain ordinary one-variable induction schemes.

6.2 Function algebras

For the logarithmic hierarchy, alternating logarithmic time, and polynomial time there exist corresponding function algebras $\mathfrak{A}_1$, $\mathfrak{A}_2$ and $\mathfrak{A}_3$ on numbers which will be defined below. $\mathfrak{A}_1$ and $\mathfrak{A}_2$ were developed by Clote in [20]. $\mathfrak{A}_3$ was developed by Ishihara in [51].

Definition 101 *$\mathfrak{A}_1$, $\mathfrak{A}_2$ and $\mathfrak{A}_3$ are function algebras on natural numbers. They contain the following initial functions.*

- *the constant zero function.*
- *the binary successor functions S_0 and S_1 with $\mathsf{S}_0(x) = 2x$ and $\mathsf{S}_1(x) = 2x + 1$.*
- *projection functions of arbitrary arity.*
- *the function BIT such that $\mathsf{BIT}(i, x) = modulo2(\lfloor \frac{x}{2^i} \rfloor)$. So, $\mathsf{BIT}(i, x)$ yields the coefficient of 2^i in the binary representation of x.*
- *the function ABS such that $\mathsf{ABS}(x) = \lceil log(x + 1) \rceil$. So, $\mathsf{ABS}(x)$ yields the length of the binary representation of x.*

- *the function $\#$ such that $\#(x, y) = 2^{\mathsf{ABS}(x)\cdot\mathsf{ABS}(y)}$.*

The algebras are closed under various operations.

- *The function algebras $\mathfrak{A}_1$, $\mathfrak{A}_2$ and $\mathfrak{A}_3$ are closed under composition.*
- *The function algebras $\mathfrak{A}_1$ and $\mathfrak{A}_2$ are closed under the following scheme of concatenation recursion on notation CRN.*

$$\begin{aligned} f(0, \vec{y}) &= g(\vec{y}) \\ f(\mathsf{S}_0(x), \vec{y}) &= \mathsf{S}_{\mathsf{BIT}(0, h_0(x,\vec{y}))}(f(x, \vec{y})) \\ f(\mathsf{S}_1(x), \vec{y}) &= \mathsf{S}_{\mathsf{BIT}(0, h_1(x,\vec{y}))}(f(x, \vec{y})) \end{aligned}$$

- *The function algebra $\mathfrak{A}_2$ is closed under the following scheme of k-bounded recursion $k - BRN$ for each $k \in \mathbb{N}$.*

$$\begin{aligned} f(0, \vec{y}) &= g(\vec{y}) \mid k \\ f(\mathsf{S}_0(x), \vec{y}) &= h_0(x, \vec{y}, f(x, \vec{y}))) \mid k \ , \\ f(\mathsf{S}_1(x), \vec{y}) &= h_1(x, \vec{y}, f(x, \vec{y}))) \mid k \end{aligned}$$

where

$$x|y = \begin{cases} x, & \textit{if } x \leq y \\ y, & \textit{else} \end{cases}$$

- *The function algebra $\mathfrak{A}_3$ is closed under the following scheme of extended concatenation recursion on notation CRN^+.*

$$\begin{aligned} f(0, \vec{y}) &= g(\vec{y}) \\ f(\mathsf{S}_0(x), \vec{y}) &= \mathsf{S}_{\mathsf{BIT}(0, h_0(x, f(x,\vec{y}),\vec{y}))}(f(x, \vec{y})) \\ f(\mathsf{S}_1(x), \vec{y}) &= \mathsf{S}_{\mathsf{BIT}(0, h_1(x, f(x,\vec{y}),\vec{y}))}(f(x, \vec{y})) \end{aligned}$$

For concatenation recursion on notation, if h_0 and h_1 give only 0 or 1 as output, we drop BIT*.*

However, we formulate our theories for words, and therefore it will be practical to work with function algebras *on words* of corresponding strengths.

Remember that the relation $\prec$ orders the words first by length, and if they have the same length lexicographically, see page 9 for its definition. The existence of an order isomorphism I from $(\mathbb{W}, \preceq)$ to $(\mathbb{N}, \leq)$ allows us to give a bit function with two *words* as input.

Definition 102 $\mathsf{bit} : \mathbb{W}^2 \to \mathbb{W}$ *is given as the function fulfilling the following specifications.*

- *If $I(w)$ is smaller than the length of v, $\mathsf{bit}(w, v)$ equals the $I(w)$-th bit of v in the sense of the* BIT *function.*
- *In all other cases, $\mathsf{bit}(w, v)$ is* 0.

Next, we define a length function on words giving a word as output, relying on I.

Definition 103 $\mathsf{abs} : \mathbb{W} \to \mathbb{W}$ *is defined such that $\mathsf{abs}(w) = v$ exactly if the length of w is $n \in \mathbb{N}$ and $I^{-1}(n) = v$.*

Definition 104 *The function algebras $\mathfrak{W}_1$, $\mathfrak{W}_2$ and $\mathfrak{W}_3$ on words (corresponding to $\mathfrak{A}_1$, $\mathfrak{A}_2$ and $\mathfrak{A}_3$) contain the following initial functions.*

- *the constant empty word function.*
- *the word successor functions s_0, s_1, concatenating 0 or 1, respectively, at the right side of their input.*
- *projection functions of arbitrary arity.*
- *the function* bit.
- *the function* abs.
- *the function $\times$ where $w \times v$ is the length of the v fold concatenation of w with itself.*
- *the function e where $\mathsf{e}(w)$ is the word w without its leading zeros, e.g. zeros bits at the left of any one bit.*

The algebras are closed under various operations.

- *The function algebras $\mathfrak{W}_1$, $\mathfrak{W}_2$ and $\mathfrak{W}_3$ are closed under composition.*
- *The function algebras $\mathfrak{W}_1$ and $\mathfrak{W}_2$ are closed under the following scheme of concatenation recursion on notation CRN. The notation $\mathsf{s}_{\mathsf{bit}(\cdots)}$ abbreviates in the following always a case distinction on the value of* $\mathsf{bit}(\cdots)$.

$$\begin{aligned} f(\epsilon, \vec{y}) &= g(\vec{y}) \\ f(\mathsf{s}_0(x), \vec{y}) &= \mathsf{s}_{\mathsf{bit}(\epsilon, h_0(x,\vec{y}))}(f(x, \vec{y})) \\ f(\mathsf{s}_1(x), \vec{y}) &= \mathsf{s}_{\mathsf{bit}(\epsilon, h_1(x,\vec{y}))}(f(x, \vec{y})) \end{aligned}$$

- *The function algebra $\mathfrak{W}_2$ is closed under the following scheme of k-bounded recursion $k - BRN$ for each $k \in \mathbb{W}$. We write $w \leq v$ for $w, v \in \mathbb{W}$ exactly if v is at least as long as w.*

$$\begin{aligned} f(\epsilon, \vec{y}) &= g(\vec{y}) \mid k \\ f(\mathsf{s}_0(x), \vec{y}) &= h_0(x, \vec{y}, f(x, \vec{y}))) \mid k \ , \\ f(\mathsf{s}_1(x), \vec{y}) &= h_1(x, \vec{y}, f(x, \vec{y}))) \mid k \end{aligned}$$

where

$$x|y = \begin{cases} x, & \text{if } x \leq y \\ y, & \text{else} \end{cases}$$

- *The function algebra $\mathfrak{W}_3$ is closed under the following scheme of extended concatenation recursion on notation CRN^+.*

$$\begin{aligned} f(\epsilon, \vec{y}) &= g(\vec{y}) \\ f(\mathsf{s}_0(x), \vec{y}) &= \mathsf{s}_{\mathsf{bit}(\epsilon, h_0(x,\vec{y},f(x,\vec{y})))}(f(x, \vec{y})) \\ f(\mathsf{s}_1(x), \vec{y}) &= \mathsf{s}_{\mathsf{bit}(\epsilon, h_1(x,\vec{y},f(x,\vec{y})))}(f(x, \vec{y})) \end{aligned}$$

For concatenation recursion on notation, if h_0 and h_1 give only 0 or 1 as output, we drop bit.

We have to include the eraser function because the scheme of concatenation recursion is weaker in word algebras than in number algebras. As we have mentioned above, this is because of the different properties of the successor functions. In the following, we prove that the addition of the eraser is necessary. For $i = 1, 2, 3$ let $\mathfrak{W}_i^-$ be the function algebra $\mathfrak{W}_i$ without e. For $i = 1, 3$ $\mathfrak{W}_i^-$ is strictly weaker than $\mathfrak{W}_i$ because for each function $f \in \mathfrak{W}_i^-$, we have that $|f(\vec{w})|$ is function of $|\vec{w}|$. Finally, for $f \in \mathfrak{W}_2^-$ with n arguments, we can prove that there is a bound k such that for any $\vec{w} \in \mathbb{W}$ the set

$$\{|f(\vec{w'})| : \vec{w'} \in \mathbb{W} \wedge |w'_1| = |w_1| \wedge \cdots \wedge |w'_n| = |w_n|\}$$

has at most k elements. $|f(\vec{w})|$ has at most $c_f \in \mathbb{N}$ different values for fixed lengths of all components of its input. Therefore the eraser e cannot be part of any of the $\mathfrak{W}_i^-$ which implies that they do not contain the logarithmic hierarchy. Also the extension of $\mathfrak{W}_i^-$ by e.g. the predecessor function or definition by cases is weaker than $\mathfrak{W}_i$ for the same reasons.

So, let us look again at the algebras $\mathfrak{W}_i$ including the eraser. In the following, we prove that they contain some important functions. The following lemma is proved as in Clote [20].

Lemma 105 *The following functions are contained in $\mathfrak{W}_1$.*

- *The signum functions sg and $\overline{\mathsf{sg}}$ which have outputs 0 or 1 depending on whether the input equals ϵ.*
- *The function rev that reverses the order of the bits of its input.*
- *The function inv that replaces zero bits by one bits and one bits by zero bits.*
- *The concatenation function $*$.*

Lemma 106 *$\mathfrak{W}_1$ is closed under sharply bounded quantification in the sense of $\preccurlyeq$.*

Proof. We can use the word version of the function in Clote's proof for $\mathfrak{A}_1$ in [20]. Then, we erase leading zeros using the eraser. $\square$

Lemma 107 *The predecessor function is contained in $\mathfrak{W}_1$.*

Proof. Our first goal is to define a function h which maps a word w to a word whose length is $|\mathsf{p_W}w|$. The function $w \mapsto \mathsf{e}\big[h^*(w)\big]$ delivers this with h^* defined as follows.

$$\begin{array}{lcl} h^*(\epsilon) & = & \epsilon \\ h^*(s_i w) & = & \mathsf{s}_{\mathsf{sg}(w)} h^*(w) \end{array}$$

We read $|\mathsf{e}\big[h^*(w)\big]|$ bits from input w to produce the predecessor. We define the auxiliary function $\mathsf{p_W}'$ as follows.

$$\begin{array}{lcl} \mathsf{p_W}'(\epsilon, v) & = & \epsilon \\ \mathsf{p_W}'(\mathsf{s}_i w, v) & = & \mathsf{s}_{\mathsf{bit}(|\mathsf{s}_i w|, v)} \mathsf{p_W}'(w, v) \end{array}$$

The predecessor $\mathsf{p_W}(w)$ is defined as $rev\Big[\mathsf{p_W}'\Big(\mathsf{e}\big[h^*(w)\big], w\Big)\Big]$. □

The most significant part function msp is given in Clote's [20] on page 20 for number inputs. $\mathsf{msp}(n, m)$ outputs the leftmost n bits of m. Using the isomorphism between $(\mathbb{W}, \preceq)$ and $(\mathbb{N}, \leq)$ this function is transfered to word inputs, analogously as bit and abs on page 145.

Lemma 108 *The following functions belong to $\mathfrak{W}_1$.*

- *Definition by cases.*
- *The most significant part function* msp.
- *Lexicographic addition $+_\mathsf{W}$ and subtraction $-_\mathsf{W}$, where $w +_\mathsf{W} v := I^{-1}(I(w) + I(v))$. Subtraction is defined analogously.*

Proof. As in Clote [20] but using the eraser, we can define a function $cond^*(w, v, u)$ with

$$cond^*(w, v, u) = \begin{cases} \mathsf{e}v, & \text{if } w = \epsilon \\ \mathsf{e}u, & else \end{cases}$$

To define definition by cases $\mathsf{d_W}$, we expand v and u by a leading 1, and use $cond^*$.

$$\mathsf{d_W}(w, v, u) := rev\Big(\mathsf{p_W}\Big(rev\Big(cond^*\Big(w, rev(\mathsf{s_1}(rev(v))), rev(\mathsf{s_1}(rev(u)))\Big)\Big)\Big)\Big)$$

Let us define the most significant part function. The auxiliary function msp^* is defined as follows.

$$\begin{array}{lcl}\mathsf{msp}^*(v,\epsilon) & = & \epsilon\\ \mathsf{msp}^*(v,\mathsf{s}_i w) & = & \mathsf{s}_{\mathsf{bit}(v,s_i w)}(\mathsf{msp}(v,w))\end{array}$$

We define $\mathsf{msp}(v,w)$ by a case distinction as follows: Check whether $\mathsf{bit}(|\mathsf{p}_{\mathsf{W}} w|, w) = 0$. If this is not the case we define $\mathsf{msp}(v,w) := \mathsf{e}(\mathsf{msp}^*(v,w))$. Else, we define $\mathsf{msp}(v,w)$ as

$$inv\Big(\mathsf{e}\big[\mathsf{msp}^*(v, inv(w))\big]\Big).$$

For lexicographic addition and subtraction it makes a difference whether one refers to the lexicographic order of words or the lexicographic order of binary representations of numbers. As a function of this, 1+1 equals 10 or 01, respectively. With the help of the most significant part function, one can define part-of quantifiers as in Clote [20]. This allows to define functions $+_n, -_n$ that work on binary representations of numbers as number addition and subtraction. To define addition and subtraction relative to the lexicographic order of *words*, we use that the number with binary representation $(1*w) -_n 1$ and the word w are identified by the isomorphism between the lexicographic order of numbers and words. This allows to define word addition and subtraction via the functions $+_n, -_n$. $\square$

In the following, we establish the equivalence of the number - and the word function algebras: We produce for any function $f \in \mathfrak{A}_i$ a corresponding function f^* in $\mathfrak{W}_i$ [2], and for any function f in $\mathfrak{W}_i$ a corresponding function f° in $\mathfrak{A}_i$. The correspondence works in such a way that for any Turing machine whose behaviour is described by $f \in \mathfrak{A}_i$ its behaviour can also be described by $f^* \in \mathfrak{W}_i$ and analogously for $f \in \mathfrak{W}_i$ and $f^\circ \in \mathfrak{A}_i$. This immediately implies the equivalence of the two function algebras with respect to computational power.

Definition 109 *Let f be a function from (tuples of) numbers to numbers. A function f^* from (tuples of) words to words is an extension of f iff we have*

[2]We will even produce a set of such functions to be precise.

for all $\vec{x} \in \mathbb{N}^n$ with binary representation $\vec{w} \in \mathbb{W}$ that $f^*(\vec{w})$ is the binary representation of $f(\vec{x})$.

Lemma 110 *Let f be in $\mathfrak{A}_i$. Then there is an extension of f in $\mathfrak{W}_i$.*

Proof. Extensions of the constant zero function, and the projection functions are trivially in $\mathfrak{W}_1$. For S_0 and S_1, we use a case distinction whether the input is zero. To simulate $\mathsf{BIT}(x, y)$ we have to calculate the x-th word from the binary representation of the number x. We use that the number with binary representation $(1 * w) -_n 1$ and the word w are identified by the isomorphism between the lexicographic order of numbers and words. Therefore, $\mathsf{BIT}(x, y)$ can be extended by manipulating its first input as hinted above and then using bit. We extend ABS in a similar way. The smash function is treated by word multiplication. For a function defined by concatenation recursion on notation, we can use the same definition scheme for its extension except that we replace the base case $F(0, \vec{z})$ by $F(\epsilon, \vec{z})$. At the end, we have to delete superfluous zeros using the eraser. For k-bounded recursion, just use the same principle in $\mathfrak{W}_2$ and change the base case as above. $\Box$

Now, we want to translate functions on words into functions on numbers. Each word can be mapped to a binary representation of a number by concatenating 1 from the left. Each number except of 0 is the image of exactly one word relative to the map described above. This motivates the following definition.

Definition 111 *Let w be a word. Then w° denotes the number with binary representation $1w$. For a function $f : \mathbb{W}^n \to \mathbb{W}$ we define $f^\circ : \mathbb{N}^n \to \mathbb{N}$ as*

$$f^\circ(\vec{n}) := \begin{cases} 0, & \textit{if one of the components of } \vec{n} \textit{ equals } 0 \\ f(\vec{w})^\circ, & \textit{if } \vec{n} = \vec{w}^\circ \end{cases}$$

(The first case takes care of the possibility that the vector of number inputs does not correspond to a vector of word inputs.)

Lemma 112 *Let f be in $\mathfrak{W}_i$. Then f° is in $\mathfrak{A}_i$.*

Proof. For the constant empty word function, the successor functions, and the projection functions we can take similar functions in $\mathfrak{A}_i$. For bit and abs, we use again that the number with binary representation $(1 * w) -_n 1$ and the word w are identified by the isomorphism between the lexicographic order of numbers and words. To calculate $(w \times v)^\circ$ from inputs $x = w^\circ$ and $y = v^\circ$, we first calculate the i-th bit of this expression, which is given as

$$(\mu j \leq 1)(\exists z \leq |y|{-}2) z{\cdot}(|x|{-}1) \leq i < (z{+}1){\cdot}(|x|{-}1) \land j = \mathsf{BIT}(i{-}z{\cdot}(|x|{-}1), x)$$

Using Clote's results in [20], especially that sharply bounded multiplication is in $\mathfrak{A}_1$, we derive that this function is in $\mathfrak{A}_i$. A concatenation recursion on notation with recursion step functions h_0, h_1 both defined as the bit function above displayed yields the searched function, using a recursion variable produced by the smash function.

To calculate $(\mathsf{e}w)^\circ$ from input $x = w^\circ$ we have to find the position of a one that has only zeros to its left in w. This position j is given as

$$(\mu j < |x|)(\forall z < |x|)\big[\mathsf{BIT}(z, x) = 1 \rightarrow z \leq j \lor z = |x| - 1\big],$$

where μ is the usual maximal witness operator. Now we check whether $j = |x| - 1$. If so, we give output 1 since the eraser completely destroys w. Else, $(\mathsf{e}w)^\circ$ equals $1 * \mathsf{lsp}(j, x)$, where $\mathsf{lsp}(y, z)$ gives the y least significant bits of z starting at the j-th bit.

For functions defined by composition the claim follows easily from the induction hypothesis. If for concatenation recursion on notation and w-bounded recursion, we use the same scheme with obvious modifications of the base case, the recursion step functions are applied once more than needed. For f defined by w-bounded recursion with recursion step functions h_0, h_1, we deal with this problem by modifying h_1° such that $h_1^\circ(0, n, \vec{m}) := n$. For concatenation recursion on notation, we just delete the superfluous bit. □

6.3 The systems LogT, AlogT and PT

In this section, we will formalise the word function algebras, we have defined in the last section. Our theories differ only minimally, so we develop them simultaneously. We make use of a safe predicate V to express that for F defined by concatenation recursion, the recursion step function must not depend on the intermediate values of F. The systems are based on an applicative base theory including the axioms for a combinatory algebra and basic types W, V which are interpreted as the set $\mathbb{W} = \{0,1\}^*$ of binary words in the standard interpretation.

6.3.1 The applicative language L

Our basic language L is a first order language for the logic of partial terms which includes:

- variables $a, b, c, x, y, z, u, v, f, g, h, \ldots$
- the applicative constants k, s, p, $\mathsf{p_0}$, $\mathsf{p_1}$, $\mathsf{d_W}$, ϵ, $\mathsf{s_0}$, $\mathsf{s_1}$, $\mathsf{p_W}$, $\mathsf{s_\ell}$, $\mathsf{p_\ell}$, $\mathsf{c_\subseteq}$, $*$, $\times$, abs, bit, e.
- relation symbols $=$ (equality), $\downarrow$ (definedness), W (binary words)
- arbitrary term application $\circ$

The terms $(r, s, t, \ldots)$ and formulas $(A, B, C, \ldots)$ of L are defined in the expected manner.

6.3.2 Rules and axioms of LogT, AlogT and PT

We use a base theory B' that is very similar to Strahm's theory B introduced in [78, 79].

The logic of B' is the classical logic of partial terms due to Beeson [4, 5]. The non-logical axioms of B' include axioms of a partial combinatory algebra:

$$\mathsf{k}xy = x, \qquad \mathsf{s}xy\downarrow \wedge\, \mathsf{s}xyz \simeq xz(yz)$$

We also have the usual axioms for pairing p with projections $\mathsf{p_0}$ and $\mathsf{p_1}$. Then, we add axioms stating that the further applicative constants, representing simple functions on words in the standard model, fulfil the expected recursion equations on W. These axioms do not contain the predicate V and are given as follows.

- defining axioms for the binary words W with ϵ, the successors s_0, s_1 which concatenate 0, 1, respectively, at the right side of a word, and the predecessor p_W which deletes the least significant bit.
- defining axioms for $\mathsf{c}_\subseteq$ which represents the initial subword relation.
- defining axioms for s_ℓ, p_ℓ which yield the lexicographic successor or -predecessor, respectively.
- definition by cases d_W on W
- word concatenation $*$, word multiplication $\times$

These axioms are fully spelled out in chapter 1.

We have the following axioms for the new constants.

(abs.1) $\mathsf{abs} : \mathsf{W} \to \mathsf{W}$

(abs.2) $\mathsf{abs}\epsilon = \epsilon$

(abs.2) $x \in \mathsf{W} \to \mathsf{abs}(\mathsf{s}_i x) = \mathsf{s}_\ell(\mathsf{abs} x)$

(bit.1) $\mathsf{bit} : \mathsf{W}^2 \to \mathsf{W}$

(bit.2) $x \in \mathsf{W} \to \mathsf{bit}(\epsilon, x) = \mathsf{d}_\mathsf{W}(1, 0, \mathsf{s}_1(\mathsf{p}_\mathsf{W} x), x)$

(bit.3) $x \in \mathsf{W} \to \mathsf{bit}(x, \epsilon) = 0$

(bit.4) $x, y \in \mathsf{W} \to \mathsf{bit}(x, y) = \mathsf{bit}(\mathsf{s}_\ell x, \mathsf{s}_i y)$

(e.1) $\mathsf{e} : \mathsf{W} \to \mathsf{W}$

(e.2) $\mathsf{e}\epsilon = \epsilon$

(e.3) $x \in \mathsf{W} \to \mathsf{e}(\mathsf{s}_0 x) = \mathsf{d}_\mathsf{W}(\epsilon, \mathsf{s}_0(\mathsf{e}x), \mathsf{e}x, \epsilon)$

(e.4) $x \in \mathsf{W} \to \mathsf{e}(\mathsf{s}_1 x) = \mathsf{s}_1(\mathsf{e}x)$

Since abs yields the length of words, we often write $|t|$ instead of $\mathsf{abs} \circ t$.

To motivate the axioms and rules for V, we have to give the informal meaning of the predicates W and V. As we mentioned already, $v \in \mathsf{V}$ is intended to mean that v is a stored, temporary inaccessible word while W contains fully accessible words. Let us explain these ideas in more detail. We can sensitively apply any of our initial functions to $w \in \mathsf{W}$, especially, we can calculate its bits, i.e. we can fully access w. So, $w \in \mathsf{W}$ is given to us similarly as content stored on a usual read-write tape of a Turing machine. On the other hand, to $v \in \mathsf{V}$ we allow only the application of the successor and predecessor functions. The motivation behind this is that given a word $v \in \mathbb{W}$ its successors and predecessor can be produced without knowing any of its bits. The knowledge where the word ends is already sufficient. Content in V is given to us similarly as content stored on a write only tape to a Turing machine having the write head always on the rightmost bit of the word it contains. Content in V can be bitwise extended or deleted but not accessed.

Sequents of the form $s \in \mathsf{V} \to t \in \mathsf{V}$ are interpreted as claiming that a transformation of content s into content t is possible where both are temporary inaccessible. Content in V is not inaccessible forever: if it is possible to produce temporary inaccessibly stored content without assuming anything about other temporary inaccessible content, we can transfer it into stable fully accessible content. This corresponds to the idea that at the end of a computation process, we can read off the result, even if during the computation there is no time or space to do so.

To formalise concatenation recursion, we let the intermediate values $f(x, \vec{y})$ be elements of V. In this way, we express that the induction step functions do not depend on them. Only at the end of computation, we dispose of $f(x, \vec{y})$. According to our motivation, we give the following axioms.

(V-intro)	$x \in \mathsf{W} \to x \in \mathsf{V}$
(V-ext)	$x \in \mathsf{V} \to \mathsf{s}_i x \in \mathsf{V}$
(V-del)	$x \in \mathsf{V} \to \mathsf{p_W} x \in \mathsf{V}$

To define the rule which allows to replace V- by W-occurrences, we need the concept of a positive L formula.

Definition 113 *For s, t being* L *terms, an* L *formula A is positive if A is build from formulas of the form $s = t$, $s \simeq t$, $s\downarrow$, $s \in \mathsf{W}$, $s \in \mathsf{V}$, using the connectives $\wedge, \vee$ and the quantifiers $\forall, \exists$.*

Now, let A be an arbitrary formula not containing V, and B_W being the positive formula B with all occurrences of V replaced by W.

(V-elim)
$$\frac{A \to B}{A \to B_{\mathsf{W}}}$$

This concludes the description of the axioms available for V. Note that we cannot even check for elements in V whether they equal ϵ. We will allow this later to construct systems of logspace strength.

The induction scheme formalising concatenation recursion has to be formulated such that for an induction formula $A[x,y] \equiv fx \simeq y$ and for a function f defined by concatenation recursion, y is stored but temporary inaccessible. In contrast, the extended concatenation recursion is formalised by a scheme that allows full access to such a y. This results in the following induction schemes for PT and AlogT, respectively, where $A[x,y]$ is a positive formula without any occurrences of W or V.

(PT-Ind)
$$\begin{aligned}&(\exists y \in \mathsf{W})A[\epsilon, y] \wedge \\ &(\forall x, y \in \mathsf{W})\big(A[x,y] \to A[\mathsf{s}_i x, \mathsf{s}_0 y] \vee A[\mathsf{s}_i x, \mathsf{s}_1 y]\big) \to \\ &(\forall x \in \mathsf{W})(\exists y \in \mathsf{W})A[x,y]\end{aligned}$$

(AlogT-Ind)
$$\begin{aligned}&(\exists y \in \mathsf{V})A[\epsilon, y]) \wedge \\ &(\forall x \in \mathsf{W})(\forall y \in \mathsf{V})\big(A[x,y] \to A[\mathsf{s}_i x, \mathsf{s}_0 y] \vee A[\mathsf{s}_i x, \mathsf{s}_1 y]\big) \to \\ &(\forall x \in \mathsf{W})(\exists y \in \mathsf{V})A[x,y]\end{aligned}$$

The scheme for AlogT is weaker than that of PT because we have to prove the induction step for inaccessible words y. Finally, the induction scheme for LogT restricts the scheme of AlogT by allowing only positive induction formulas A which are W, V and *disjunction* free, except of disjunctions occurring within formulas of the form $s \simeq t$. In the following, we drop the V-axioms from PT since they are unnecessary. This concludes the description of the theories LogT, AlogT, and PT.

The standard open term model $\mathcal{M}(\lambda\eta)$ for B can be easily generalised to model the introduced theories: Take the universe of open λ terms and consider the usual reduction of the extensional untyped lambda calculus $\lambda\eta$, augmented by suitable reduction rules for the constants other than k and s.

Interpret application as juxtaposition. Two terms are equal if they have a common reduct, W and V denote the set of terms which reduce to a "standard" word $\overline{w}$.

6.3.3 Versions of LogT and AlogT without V

It is possible to define applicative theories LogT^*, AlogT^* without V that prove totality for the same class of functions as LogT, AlogT, respectively. LogT^*, AlogT^* are simpler but more obscure than their analogons. They are produced from LogT, AlogT by removing the axioms and the rule for V and replacing induction by LogT^*-Ind, or AlogT^*-Ind, respectively. AlogT^*-Ind is defined as follows, for $A[x,y]$ being a positive formula without any occurrences of W or V.

$$(\mathsf{AlogT}^*\text{-Ind}) \qquad \begin{aligned}&(\exists y \in \mathsf{W})A[\epsilon, y] \wedge \\ &(\forall x \in \mathsf{W})(\forall y)\big(A[x,y] \rightarrow A[\mathsf{s}_i x, \mathsf{s}_0 y] \vee A[\mathsf{s}_i x, \mathsf{s}_1 y]\big) \rightarrow \\ &(\forall x \in \mathsf{W})(\exists y \in \mathsf{W})A[x,y]\end{aligned}$$

The scheme for LogT^*-Ind restricts the scheme of AlogT^* by allowing only formulas not containing disjunctions, except of the ones occurring within formulas of the form $s \simeq t$.

Lemma 114 *Let T be the theory LogT or AlogT, and T^* the theory LogT^* or AlogT^*, respectively. For all formulas A of* L *we have*

$$\mathsf{T}^* \vdash A \Rightarrow \mathsf{T} \vdash A.$$

Proof. We only have to prove the induction axiom of T^*. We argue informally in T and assume

$$(\mathsf{T}^*\text{-Ind}) \qquad \begin{aligned}&(\exists y \in \mathsf{W})A[\epsilon, y] \wedge \\ &(\forall x \in \mathsf{W})(\forall y)\big(A[x,y] \rightarrow A[\mathsf{s}_i x, \mathsf{s}_0 y] \vee A[\mathsf{s}_i x, \mathsf{s}_1 y]\big)\end{aligned}$$

Since W is a subset of V, and by restricting the universal quantifier, we obtain the premisses of T-Ind and deduce $(\forall x \in \mathsf{W})(\exists y \in \mathsf{V})A[x,y]$. An application of the V elimination rule delivers the desired result. $\square$

For the lower bound proof of the systems LogT^*, AlogT^*, we refer to section 6.3.4.

6.3.4 Lower bound

We find a lower bound for the theories LogT, AlogT and PT in the sense of provably total functions. We use the following standard definition.

Definition 115 *A function $F : \mathbb{W}^n \to \mathbb{W}$ is called provably total in an $\mathcal{L}$ theory Th, if there exists a closed L term t_F such that*

(i) $\mathsf{Th} \vdash t_F : \mathsf{W}^n \to \mathsf{W}$ *and, in addition,*

(ii) $\mathcal{M}(\lambda\eta) \vDash t_F\overline{w}_1 \cdots \overline{w}_n = \overline{F(w_1, \dots, w_n)}$ *for all* $w_1, \dots, w_n$ *in* $\mathbb{W}$.

Lemma 116

- *All functions of $\mathfrak{W}_1$ are provably total in* LogT.
- *All functions of $\mathfrak{W}_2$ are provably total in* AlogT.
- *All functions of $\mathfrak{W}_3$ are provably total in* PT.

Proof.

The initial functions are clearly provably total and the provably total functions are closed under composition for all introduced theories. Now, let us deal with the case where the function $F(x, \vec{z})$ is defined by concatenation recursion applied to $G(\vec{z})$ and $H_i(x, \vec{z})$ being both in $\mathfrak{W}_1$ or $\mathfrak{W}_2$, respectively. The induction hypothesis delivers terms t_G and t_{H_i} representing G and H_i. We find a closed term t_F such that

$$t_F\epsilon\vec{z} \simeq t_G\vec{z}$$
$$t_F(\mathsf{s}_i w)\vec{z} \simeq \mathsf{d_W}(\mathsf{s}_0(t_F w\vec{z}), \mathsf{s}_1(t_F w\vec{z}), \mathsf{bit}(\epsilon, t_{H_i} w\vec{z}), 0)$$

We take $t_F x\vec{z} = y$ as induction formula $A[x, y]$ and assume that $\vec{z} \in \mathsf{W}$. Let us show that the premisses of (LogT-Ind) or (AlogT-Ind), respectively, hold. The first conjunct holds because of the induction hypothesis for G and because

$\mathsf{W} \subseteq \mathsf{V}$. To prove the second conjunct, we can assume $t_F x\vec{z} = y \in \mathsf{V}$ for $x \in \mathsf{W}$. Because of the properties of the H_i, we have that $\mathsf{bit}(\epsilon, t_{H_i} x\vec{z})$ equals 0 or 1. $y \in \mathsf{V}$ implies $\mathsf{s}_i y \in \mathsf{V}$, which means that all components of the definition by cases that is partially equal to $t_F(\mathsf{s}_i x)\vec{z}$ are defined. If we now assume that $\mathsf{bit}(\epsilon, t_{H_i} w\vec{z})$ equals 0, we derive $t_F \mathsf{s}_i x\vec{z} = \mathsf{s}_0 y$. The other case works analogously. The application of induction delivers $(\forall x \in \mathsf{W})(\exists y \in \mathsf{V}) t_F x = y$, which yields the totality of t_F using (V-elim).

For a function F defined by extended concatenation recursion, we define the term t_F similarly as before. The crucial difference is that the step functions H_i now depend on the intermediate values of F represented by y. To justify the application of t_{H_i} to y we have to impose $y \in \mathsf{W}$.

Now, let us deal with the case where the function $F(x, \vec{y})$ is defined by w-bounded recursion applied to $G(\vec{y})$ and $H_i(x, \vec{y})$ both being in $\mathfrak{W}_2$. The induction hypothesis delivers terms t_G and t_{H_i} representing G, H_i. We find a closed term t_F such that

$$t_F \epsilon \vec{z} \simeq t_G \vec{z} \mid \overline{w}$$
$$t_F(\mathsf{s}_i w)\vec{z} \simeq t_{H_i} w (t_F w \vec{z})\vec{z} \mid \overline{w}$$

We define the formula $s \preccurlyeq \overline{w}$ for each $w \in \mathbb{W}$ to be a disjunction composed of all disjuncts of the form $s = \overline{v}$ where $v \leq w$ (assume an arbitrary but fixed bracketing). In the base theory B (see page 152) we can prove the following by external induction for all $w \in \mathbb{W}$.

$$\mathsf{B} \vdash s \preccurlyeq \overline{w} \leftrightarrow s \leq_{\mathsf{W}} \overline{w}$$

To prove the totality of t_F, we take $t_F x\vec{z} \preccurlyeq \overline{w}$ as induction formula and assume $\vec{z} \in \mathsf{W}$. Then, the first conjunct of the antecedent holds because of the induction hypothesis for G. To prove the second conjunct, we can assume $t_F x\vec{z} \preccurlyeq \overline{w}$ for $x \in \mathsf{W}$. This implies that $t_F x\vec{z}$ is a word, which yields together with the induction hypothesis for the H_i the desired result. Note, that the axiom $x \in \mathsf{V} \rightarrow \mathsf{p_W} x \in \mathsf{V}$ was not needed to prove the lower bound. □

For the systems LogT^* and AlogT^* we can prove the lower bound in a very similar way. To justify concatenation recursion, we use induction for the formula $t_F x\vec{z} \simeq y$ where t_F is defined as before except that it uses the version of case distinction whose definedness does not depend on the definedness of the not chosen argument [3]. These modifications of the induction formula and the term t_F are necessary because we cannot guarantee the definedness of $\mathsf{s}_0 y$ or $\mathsf{s}_1 y$ for an arbitrary y [4].

6.3.5 Upper bound

For the upper bound proof, we work with total versions $\mathsf{LogT}{\downarrow}$, $\mathsf{AlogT}{\downarrow}$ and $\mathsf{PT}{\downarrow}$ of the introduced theories. They are obtained from the corresponding partial theories by the following modifications:

- We drop $\downarrow$, and replace all formulas of the form $s \simeq t$ by $s = t$ in the axioms and rules.
- We work with the usual classical logic.
- We redefine the notion of positive formulas as follows.

Definition 117 *For s, t being L terms, an L formula A is positive if A is build from formulas of the form $s = t$, $s \in \mathsf{W}$, $s \in \mathsf{V}$, using the connectives $\wedge, \vee$ and the quantifiers $\forall, \exists$.*

It is easy to see that $\mathsf{LogT}{\downarrow}$, $\mathsf{AlogT}{\downarrow}$ and $\mathsf{PT}{\downarrow}$ are equivalent to the extensions of LogT, AlogT and PT by the axiom $(\forall x, y)xy{\downarrow}$.

We formulate $\mathsf{LogT}{\downarrow}$, $\mathsf{AlogT}{\downarrow}$ and $\mathsf{PT}{\downarrow}$, which we just call LogT, AlogT and PT in the following, in Gentzen's classical sequent calculus LK. We assume familiarity with LK as it is presented, for example, in Girard's [47]. By formulating induction as a rule, we obtain systems with only positive main

[3]Such a case distinction d'_{W} is defined as follows.

$$\mathsf{d}'_{\mathsf{W}}(x, y, v, w) := \mathsf{d}_{\mathsf{W}}(\lambda z.x, \lambda z.y, v, w)0$$

[4]Nevertheless, if we add the axiom $(\forall x)\mathsf{s}_i x{\downarrow}$ we are able to prove the antecedent of induction for the induction formula $t_f x\vec{z} = y$ similarly as above displayed for LogT and AlogT. The axiom $(\forall x)\mathsf{s}_i x{\downarrow}$ is usually not present in axiomatisations of combinatory algebras, but its addition does not increase the proof theoretic strength of any of our systems and is true in both, recursion theoretic - and term models.

formulas for all theories. For the sequent style systems, we give the V-elimination rule as follows for V-free Γ, Δ, a positive B, and B_{W} being B with all occurrences of V replaced by W.

$$\frac{\Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow B_{\mathsf{W}}, \Delta}$$

Note that compared to Cantini's [17], we allow more general side formulas, and B can be a positive formula of arbitrary complexity. The reason why this is possible is that for our realisation approach an elimination of non-positive cuts is sufficient.

By standard techniques, see e.g. Girard's [47], we eliminate all cuts with non-positive cut-formula. This implies that sequents containing only positive formulas have proofs containing only positive formulas. The above sketched transformation of the theories into sequent style, and the subsequent partial cut-elimination are described in detail in Strahm's [79] on pages 24-26 for similar theories.

Due to the cut-elimination result, we can restrict the realisation to positive formulas. We use two realisation relations $\mathfrak{r}$ and $\mathfrak{R}$. The first trivializes the V-predicate, the second treats V exactly like W. Correspondingly, we deliver for provable sequents two realisation functions: F operating on $\mathfrak{r}$ realisers, and $F^{\Box}$ on $\mathfrak{R}$ realisers. The interplay of F and $F^{\Box}$ allows to catch both aspects of V: On the one hand the possibilities to derive something from $t \in \mathsf{V}$ are very restricted, so we are not allowed to use its realiser as freely as a realiser of $t \in \mathsf{W}$; on the other hand because of (V-elim) under some conditions we have to produce a full W-atom (e.g. a formula of the form $t \in \mathsf{W}$) realiser from a V-atom realiser.

Let us define both realisation relations. Remember that $\langle \cdots \rangle$ denotes Clote's pairing function [20] within the logarithmic hierarchy.

Definition 118 (Realisation relation $\mathfrak{r}$)

$$
\begin{array}{lll}
\rho \ \mathfrak{r} \ \mathsf{W}(t) & \textit{iff} & \mathcal{M}(\lambda\eta) \vDash t = \overline{\rho}, \\
\rho \ \mathfrak{r} \ \mathsf{V}(t) & \textit{iff} & \rho = \epsilon, \\
\rho \ \mathfrak{r} \ (t_1 = t_2) & \textit{iff} & \rho = \epsilon \textit{ and } \mathcal{M}(\lambda\eta) \vDash t_1 = t_2, \\
\rho \ \mathfrak{r} \ (A \wedge B) & \textit{iff} & \rho = \langle \rho_1, \rho_2 \rangle \textit{ and } \rho_1 \ \mathfrak{r} \ A \textit{ and } \rho_2 \ \mathfrak{r} \ B \\
\rho \ \mathfrak{r} \ (A \vee B) & \textit{iff} & \rho = \langle i, \rho_2 \rangle \textit{ and either } i = 0 \textit{ and } \rho_2 \ \mathfrak{r} \ A \textit{ or} \\
 & & \qquad\qquad\qquad i = 1 \textit{ and } \rho_2 \ \mathfrak{r} \ B, \\
\rho \ \mathfrak{r} \ (\forall x)A(x) & \textit{iff} & \rho \ \mathfrak{r} \ A(u) \textit{ for a fresh variable } u, \\
\rho \ \mathfrak{r} \ (\exists x)A(x) & \textit{iff} & \rho \ \mathfrak{r} \ A(t) \textit{ for some term } t.
\end{array}
$$

Definition 119 (Realisation relation $\mathfrak{R}$) *The realisation relation $\mathfrak{R}$ is defined as $\mathfrak{r}$ except that the clause*

$$\rho \ \mathfrak{r} \ \mathsf{V}(t) \qquad \textit{iff} \qquad \rho = \epsilon$$

is replaced by

$$\rho \ \mathfrak{R} \ \mathsf{V}(t) \qquad \textit{iff} \qquad \mathcal{M}(\lambda\eta) \vDash t = \overline{\rho}$$

Sequences of formulas are realised as usual giving tuples of realisers.

Note that for a formula A realised by ρ relative to $\mathfrak{r}$ or $\mathfrak{R}$, we can talk about the atoms of A that are realised by ρ in a natural way. This is so because ρ just contains individual realisers of possibly substituted atoms of A (with multiplicity), within a structure of pairs allowing to find for each individual realiser the corresponding atom. Let A be e.g. the formula

$$(\exists x)(x \in \mathsf{W} \wedge (x = 0 \vee x = 0)).$$

Its $\mathfrak{r}$ -realiser $\langle 0, \langle 0, \epsilon \rangle \rangle$ realises the first and the second atom from the left, but does not realise the third one.

For any positive formula A, there is a function $\cdot^A$ (projection function) within the logarithmic hierarchy that transfers a given $\mathfrak{R}$ realisers of A into a $\mathfrak{r}$ realiser of A, just by inserting ϵ at the suitable positions.

Lemma 120 *For any formula A, there exists a projection $\cdot^A$ within the logarithmic hierarchy such that for any word ρ and any substitutions $[\vec{s}]$, $[\vec{t}]$*

- $\rho \ \mathfrak{R} \ A \Rightarrow \rho^A \ \mathfrak{r} \ A$
- $\cdot^{A[\vec{s}]}$ *and* $\cdot^{A[\vec{t}]}$ *denote the same function.*
- ρ *and* ρ^A *realise the same atoms in the sense described above.*
- $\cdot^{QxA[x]}$ *is given as* $\cdot^{A[u]}$ *for* $Q = \exists, \forall$ *and a fresh variable* u.

Proof. We prove the claim by induction on the complexity of A. The fact that the given function lies within the logarithmic hierarchy is always obvious. If A is an atom without occurrence of V the relations $\mathfrak{r}$ and $\mathfrak{R}$ coincide, so we can choose $\cdot^A$ as identity. For atoms of the form $t \in \mathsf{V}$, we take $\cdot^A$ as the constant ϵ function. For $A \equiv A_0 \wedge A_1$, we define ρ^A as $\langle \rho^{A_0}, \rho^{A_1} \rangle$. For $A \equiv A_0 \vee A_1$, we define

$$\rho^A := \begin{cases} \langle 0, \rho_1^{A_0} \rangle, & \text{if } \rho_0 = 0 \\ \langle 1, \rho_1^{A_1} \rangle, & \text{else} \end{cases}$$

In all cases analysed above, the given function obviously fulfils the claimed properties. Let $A \equiv (\exists x)B[x/u]$. We show that $\cdot^A$, defined as required, has the correct properties. Assume $\rho \ \mathfrak{R} \ A$. This implies $\rho \ \mathfrak{R} \ B[t]$ for a certain term t. Since the functions $\cdot^{B[u]}$ and $\cdot^{B[t]}$ are identical, the induction hypothesis delivers $\rho^A \ \mathfrak{r} \ B[t]$ and therefore $\rho^A \ \mathfrak{r} \ A$ as required. For $\cdot^{\forall x B[x]}$, we use the same argument. $\square$

The projection function can easily be generalised to tuples of realisers and sequences of formulas, and is written as $\cdot^\Gamma$ in such cases. We write just $\cdot^*$ for projection functions, if they are clear from the context.

Theorem 121 *Let* T_0, T_1, T_2 *denote* LogT, AlogT, PT, *respectively. Let* $\Gamma \rightarrow \Delta$ *be a sequent of positive formulas with* $\Gamma \equiv A_1, \ldots, A_n$ *and* $\Delta \equiv D_1, \ldots, D_m$ *and assume* $\mathsf{T}_i \vdash^\star \Gamma[\vec{u}] \rightarrow \Delta[\vec{u}]$. *Then there exists functions* $F, F^\Box : \mathbb{W}^n \rightarrow \mathbb{W}$ *in* $\mathfrak{W}_i$ *such that for each substitution* $[\vec{s}]$ *and each* $\vec{\rho} \ \mathfrak{R} \ \Gamma[\vec{s}]$ *the following conditions hold.*

- $F^\Box(\vec{\rho}) \ \mathfrak{R} \ \Delta[\vec{s}]$

- $F^{\Box}(\vec{\rho})_1 = F(\vec{\rho}^{\,\Gamma})_1 = k$
- $F^{\Box}(\vec{\rho})_2^{D_k} = F(\vec{\rho}^{\,\Gamma})_2$

Let us explain the three conditions. The first conditions claims that a function $F^{\Box}$ delivers an $\mathfrak{R}$ realiser of $\Delta[\vec{s}]$, which means a realiser reflecting the V-predicate, on input $\vec{\rho}$. The second condition claims that the realisation functions F and $F^{\Box}$ pick the same formula $D_k[\vec{s}]$ of $\Delta[\vec{s}]$ to realise. Note, that the input of the realisation function F is a projection of the input of $F^{\Box}$. Finally, the third condition connects the realisers delivered by F and $F^{\Box}$, stating that the first is a projection of the second.

Proof. We proof the theorem by induction on the depth of the positive proof. Note that for PT, we only have to give a realisation function F. All logical and equation axioms can be realised by suitable projections. The applicative axioms are realised using suitable initial functions of $\mathfrak{W}_i$. The axioms

$$t \in \mathsf{W} \Rightarrow t \in \mathsf{V} \;/\; t \in \mathsf{V} \Rightarrow \mathsf{s}_i t \in \mathsf{V}$$

are realised by F the constant $\langle 1, \epsilon\rangle$ function and $F^{\Box}$ defined as $\lambda x.\langle 1, x\rangle$, $\lambda x.\langle 1, \mathsf{s}_i x\rangle$, respectively. The totality axiom for the predecessor is realised similarly.

Let us switch to the realisation of the rules. The logical rules do not pose special difficulties. Let us realise the V-elimination rule given as follows where V does not occur in Γ, Δ is positive, and B_{W} is B with all V replaced by W.

$$\frac{\Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow B_{\mathsf{W}}, \Delta}$$

For the premise, we have realisation functions $P, P^{\Box}$. We construct the realisation function F. Let $\vec{\rho}$ be the given realisers for Γ relative to realisation relation $\mathfrak{r}$. Since Γ does not contain V they are realisers of Γ relative to $\mathfrak{R}$. An application of $P^{\Box}$ delivers an $\mathfrak{R}$ realiser of B, Δ. Because $\mathfrak{R}$ realisers of $t \in \mathsf{V}$ are equal to $\mathfrak{r}$ realisers of $t \in \mathsf{W}$, $P^{\Box}(\vec{\rho})$ simultaneously yields an $\mathfrak{r}$ and $\mathfrak{R}$ realiser of B_{W}, Δ. Therefore, we define F and $F^{\Box}$ as $P^{\Box}$.

Let us now realise the induction scheme of AlogT with premisses

- $\Gamma \Rightarrow (\exists y \in \mathsf{V})A[\epsilon, y], \Delta$

- $\Gamma, x \in \mathsf{W}, y \in \mathsf{V}, A[x, y] \Rightarrow A[\mathsf{s}_0 x, \mathsf{s}_0 y] \lor A[\mathsf{s}_0 x, \mathsf{s}_1 y], \Delta$
- $\Gamma, x \in \mathsf{W}, y \in \mathsf{V}, A[x, y] \Rightarrow A[\mathsf{s}_1 x, \mathsf{s}_0 y] \lor A[\mathsf{s}_1 x, \mathsf{s}_1 y], \Delta$

and the conclusion

- $\Gamma, t \in \mathsf{W} \Rightarrow (\exists y \in \mathsf{V})A[t, y], \Delta,$

where the usual variable condition applies. We assume realisation functions $G, G^{\Box}, H_0, H_0^{\Box}, H_1, H_1^{\Box}$ for the premisses and that Γ contains n formulas. First, we define a function $I : \mathbb{W}^{n+1} \to \mathbb{W}$ which produces $\mathfrak{r}$ -realisers of the induction formula A relative to specific substitutions. This can be done by the following c_A-bounded recursion for a $c_A \in \mathbb{W}$ such that for all $\rho \in \mathbb{W}$ and all terms s, t

$$\rho \ \mathfrak{r} \ A[s, t] \Rightarrow \rho \leq c_A.$$

The existence of c_A easily follows from the following lemma which can be proved by induction on the complexity of the formula A.

Lemma 122 *Let A be a positive, W-free formula. Then there exists a word c_A such that for any substitution $[\vec{s}]$ and any word ρ*

$$\rho \ \mathfrak{r} \ A[\vec{s}] \Rightarrow \rho \leq c_A.$$

We abbreviate multiple projections of a word w as $w_{n_0,\cdots,n_m}$ and define I as follows.

$$\begin{aligned} I(\vec{z}, \epsilon) \quad &= \quad G(\vec{z})_{2,2} \mid c_A \\ I(\vec{z}, \mathsf{s}_i w) \quad &= \quad H_i(\vec{z}, w, \epsilon, I(\vec{z}, w))_{2,2} \mid c_A \end{aligned}$$

Under the assumption $\vec{z} \ \mathfrak{r} \ \Gamma$, for any $w \in \mathbb{W}$, $I(\vec{z}, w)$ delivers a realiser of $A[\overline{w}, \overline{v}]$ for some $v \in \mathbb{W}$, as long as no side formula is realised.

Let us define a second auxiliary function $Q : \mathbb{W}^{n+1} \to \mathbb{W}$ producing an $\mathfrak{R}$ realisers for the first inner conjunct of $(\exists y \in \mathsf{V})A[t, y]$ given an $\mathfrak{R}$ -realiser $\vec{z}$ of Γ, presupposed that no side formula is realised.

$$\begin{aligned} Q(\vec{z}, \epsilon) \quad &= \quad G^{\Box}(\vec{z})_{2,1} \\ Q(\vec{z}, \mathsf{s}_i w) \quad &= \quad \mathsf{s}_{\mathsf{bit}(\epsilon, H_i[(\vec{z})^*, w, \epsilon, I((\vec{z})^*, w)]_{2,1})} Q(\vec{z}, w) \end{aligned}$$

Note, that we have to use H_i instead of $H_i^{\Box}$ to find the suitable successor because we are not allowed to replace ϵ by a term containing intermediate values of Q. Because of the induction hypothesis for the premise realisation functions, both auxiliary functions are in $\mathfrak{W}_2$.

To define the realisation function F, we have to decide whether a side formula is realised. This case distinction cannot be integrated into the recursive definition of the realisation function as usual, because of the weakness of CRN. Therefore, we have to distinguish cases independently of earlier values of F. Let us define the realisation function F as follows:

Case 1 $G(\vec{z})_1 \neq 1$. We define $F(\vec{z}, w)$ as $G(\vec{z})$.

Case 2 $G(\vec{z})_1 = 1 \land (\exists l \preccurlyeq |w|)P(l, w, \vec{z})$ where

$$\begin{aligned} P(l, w, \vec{z}) :\Leftrightarrow & (\exists i)(\mathsf{s}_i(\mathsf{msp}(l, w)) = \mathsf{msp}(l, w) \land \\ & H_i(\vec{z}, \mathsf{msp}(l, w), \epsilon, I(\vec{z}, \mathsf{msp}(l, w)))_1 \neq 1) \end{aligned}$$

This case applies if in the course of recursion, we hit a side formula. We define $F(\vec{z}, w)$ as

$$H_i(\vec{z}, \mathsf{msp}(j, w), \epsilon, I(\vec{z}, \mathsf{msp}(j, w))),$$

where $j = (\nu y \preceq |w|)P(y, w, \vec{z})$, with ν being the usual maximal witness operator.

Case 3 Case 1 and 2 are not satisfied. We define $F(\vec{z}, w)$ as $\langle 1, \langle \epsilon, I(\vec{z}, w)\rangle\rangle$.

We can define the realisation function $F^{\Box}$ very similarly as F, using the same case distinction. The main difference is that in the third case we have to produce a non trivial realiser of the V-occurrence of the induction formula. We use in the following the same abbreviations as above.

Case 1 $G((\vec{z})^*)_1 \neq 1$. We define $F^{\Box}(\vec{z}, w)$ as $G^{\Box}(\vec{z})$.

Case 2 $G((\vec{z})^*)_1 = 1 \land (\exists l \preccurlyeq |w|)P(l, w, \vec{z})$. We define $F^{\Box}(\vec{z}, w)$ as

$$H_i^{\Box}(\vec{z}, \mathsf{msp}(j, w), Q(\vec{z}, \mathsf{msp}(j, w)), I((\vec{z})^*, \mathsf{msp}(j, w))),$$

where $j = (\nu y \preceq |w|)P(y, w, \vec{z})$.

Case 3 Case 1 and 2 are not satisfied. We define $F^{\Box}(\vec{z}, w)$ as

$$\langle 1, \langle Q(\vec{z}, w), I((\vec{z})^*, w)\rangle\rangle.$$

$\mathfrak{W}_1$ is closed under sharply bounded quantification and the sharply bounded minimal witness - and maximal witness operator (see Clote's [20]), which implies that the functions F, $F^{\Box}$ are in $\mathfrak{W}_2$. Let us prove their correctness. We assume $\vec{z}\ \mathfrak{R}\ \Gamma[\vec{s}]$ and let w be an arbitrary word. We prove the required properties by external induction on w. We use that for the arguments $\vec{z}, w$ of F and the arguments $\vec{z}^*, w$ of $F^{\Box}$ always the same case is fulfilled. Let w be ϵ. Then the assertion follows immediately from the induction hypothesis about G, $G^{\Box}$ and the fact that case 2 cannot be fulfilled. Let us prove the assertion for $\mathsf{s}_i w$. We abbreviate $F^{\Box}(\vec{z}, w)$ as a. First, we assume that case 3 is fulfilled for $\vec{z}, w$, so a_2 does not realise a side formula, and the induction hypothesis delivers

$$a_2\ \mathfrak{R}\ (\exists y \in \mathsf{V})A[\overline{w}, y][\vec{s}]. \tag{6.1}$$

This implies

$$I((\vec{z})^*, w)\ \mathfrak{r}\ A\big(\overline{w}, \overline{a_{2,1}}\big). \tag{6.2}$$

The induction hypothesis for the induction step premisses delivers

$$H_i((\vec{z})^*, w, \epsilon, I((\vec{z})^*, w))\ \mathfrak{r}\ A\big(\overline{\mathsf{s}_i w}, \mathsf{s}_0\overline{a_{2,1}}\big) \vee A\big(\overline{\mathsf{s}_i w}, \mathsf{s}_1\overline{a_{2,1}}\big), \Delta[\vec{s}]. \tag{6.3}$$

First, assume that a formula in $\Delta[\vec{s}]$ is realised. Then for $\vec{z}, \mathsf{s}_i w$ the second case holds with the only witness 0. This implies that $F^{\Box}(\vec{z}, \mathsf{s}_i w)$ is defined as $H_i^{\Box}(\vec{z}, w, Q(\vec{z}, w), I((\vec{z})^*, w))$. F is defined analogously and the assertion holds because of the induction hypothesis for $H_i, H_i^{\Box}$. Assume now that the main formula is realised. Then, because case 3 holds for $(\vec{z}, w)$, case 3 also holds for $(\vec{z}, \mathsf{s}_i w)$. $\mathsf{s}_0 a_{2,1}$ or $\mathsf{s}_1 a_{2,1}$ is the witness of the restricted existential quantifier. The value of $\mathsf{bit}(\epsilon, H_i[(\vec{z})^*, w, \epsilon, I((\vec{z})^*, w)]_{2,1})$ determines correctly which one we have to choose. This implies that Q yields the correct witness. The correctness of the component produced by I follows easily from (3). Therefore, the definition of $F^{\Box}$ implies

$$F^{\Box}(\vec{z}, \mathsf{s}_i w)\ \mathfrak{R}\ (\exists y \in \mathsf{V})A[\overline{\mathsf{s}_i w}, y][\vec{s}]. \tag{6.4}$$

$F(\vec{z}, \mathsf{s}_i w)$ is defined as projection of $F^{\square}(\vec{z}, \mathsf{s}_i w)$, so the other required properties of realisation functions immediately follow. Finally, if for $\vec{z}, w$ the first or the second case is fulfilled, then $F^{\square}(\vec{z}, w)_2$ and $F(\vec{z}, w)_2$ realise side formulas, and the claim follows from the monotonicity of the cases 1 and 2.

To deal with the weaker induction scheme of LogT, we argue similarly. Since the induction formula does not contain disjunctions this time, we can assume that it is always realised by the same word c_A. Therefore, we get correct realisation functions from the functions F, $F^{\square}$ above by replacing all terms of the form $I(a, b)$ by c_A. Since I is not needed, the modified realisation functions are in $\mathfrak{W}_1$.

To deal with the induction scheme of PT, we define the realisation function F by bounded recursion. We abbreviate $H_i(\vec{z}, w, F(\vec{z}, w)_{2,1}, F(\vec{z}, w)_{2,2})$ as $\tilde{H}_i(\vec{z}, w)$ and suppress a suitable polynomial bound which can be found easily.

- $F(\vec{z}, \epsilon) = G(\vec{z})$
- $$F(\vec{z}, \mathsf{s}_i w) = \begin{cases} \tilde{H}_i(\vec{z}, w), & \text{if } F(\vec{z}, w)_1 = 1 \text{ and} \\ & \tilde{H}_i(\vec{z}, w)_1 \neq 1 \\ \langle 1, \langle \mathsf{s}_{\mathsf{bit}(\epsilon, \tilde{H}_i(\vec{z}, w)_{2,1})} F(\vec{z}, w)_{2,1}, & \\ \tilde{H}_i(\vec{z}, w)_{2,2} \rangle \rangle, & \text{if } F(\vec{z}, w)_1 = 1 \text{ and} \\ & \tilde{H}_i(\vec{z}, w)_1 = 1 \\ F(\vec{z}, w), & \text{else} \end{cases}$$

The function F is in $\mathfrak{W}_3$ because of lemmas 110, 112 and Ishihara's result delivering the equivalence of $\mathfrak{A}_3$ and $[0, I, \mathsf{S}_0, \mathsf{S}_1, \#, COMP, BRN]$, where BRN denotes bounded recursion on notation. Again, an external induction on the value of w yields the correctness of the realisation function. As mentioned earlier, no function $F^{\square}$ is needed. $\square$

The previous lemma implies together with the lower bound lemma 116 the proof theoretic characterisation of the theories:

Theorem 123

- *The provably total functions of* LogT *are exactly the functions in the logarithmic hierarchy.*

- *The provably total functions of* AlogT *are exactly the functions computable in alternating logarithmic time.*
- *The provably total functions of* PT *are exactly the functions computable in polynomial time.*

6.3.6 Extending the theories by V induction

The theories introduced before all contain two variable induction schemes to express the dependence between $F(w, \vec{z})$ and $F(\mathsf{s}_i w, \vec{z})$ for F defined by concatenation recursion. There does not seem to be a way to justify precisely this recursion with an ordinary *one* variable induction scheme. We now address the question whether ordinary one variable induction schemes can be added sensibly to the introduced base theories. Let us first consider the addition of V induction to the theory LogT. Let $*$ be a closed term satisfying the recursion equations for concatenation. Using V induction, we easily prove

$$x \in \mathsf{W}, y \in \mathsf{V} \Rightarrow y * x \in \mathsf{V}.$$

Using this fact and again V induction we can justify the following recursion scheme, treated by Lind in [65].

$$\begin{aligned} F(\vec{z}, \epsilon) &:= G(\vec{z}) \\ F(\vec{z}, \mathsf{s}_i w) &:= F(\vec{z}, w) * H_i(\vec{z}, w) \end{aligned}$$

It is unknown, whether the algebras $\mathfrak{W}_1$ or $\mathfrak{W}_2$ extended by Lind's concatenation recursion correspond to natural complexity classes. Still, it can be shown that logspace is closed under the new recursion principle by keeping track of the length of F by sharply bounded recursion. Therefore, for our theories with one-variable induction schemes, we will aim at strength logspace.

6.4 A new safe function algebra for logspace

We define a two-sorted algebra $\mathfrak{LS}$ of logspace strength. $\mathfrak{LS}$ merits attention because it allows to describe logspace from natural initial functions with only one recursion scheme that does not contain explicit bounds. It differs from the famous Cook-Bellantoni safe algebra for polynomial time only by restricting case distinction, and by allowing an additional initial function abs yielding the length of its input.

Definition 124 *The algebra $\mathfrak{LS}$ is the smallest function algebra (on words) which contains the following initial functions and is closed under the following operations:*

Initial functions

- ϵ, s_0, s_1, p_W *with safe input, and* abs *with normal input.*
- *Case distinction for safe arguments.*

$$\mathsf{case}(;y_1,y_2,y_3) := \begin{cases} y_2, & \textit{if } y_1 = \epsilon \\ y_3, & \textit{else} \end{cases}$$

- *Projections $\pi_i^{n,m}$ with both normal and safe inputs.*

Operations

- *Safe composition.*

$$f(\vec{x};\vec{y}) = h(\vec{g}(\vec{x};);\vec{j}(\vec{x};\vec{y}))$$

- *Safe recursion on notation.*

$$\begin{aligned} f(\epsilon,\vec{x};\vec{y}) &:= g(\vec{x};\vec{y}) \\ f(\mathsf{s}_i(w),\vec{x};\vec{y}) &:= h_i(w,\vec{x};f(w,\vec{x};\vec{y}),\vec{y}), \end{aligned}$$

Note that Cook and Bellantoni's safe algebra B allows the following stronger case distinction.

$$\mathsf{case}(;y_1,y_2,y_3) := \begin{cases} y_2, & \text{if } mod_2(y_1) = 0 \\ y_3, & \text{else} \end{cases}$$

In addition, one does not need to include abs as an initial function of B since it is definable. Apart from these differences, $\mathfrak{LS}$ and B are given in exactly the same way.

Let us mention authors that have constructed other safe descriptions of logspace. Bellantoni [6] described logspace as B without s_0. His algebra only delivers the sharply bounded functions computable in logspace.

Another two-sorted algebra was introduced in Oitavem's [68]. Let us compare $\mathfrak{LS}$ to Oitavem's system $Logspace_{CT}$. The main difference is that in $\mathfrak{LS}$, safe recursion is stronger since successor - and predecessor functions can be applied to safe inputs. Therefore, our algebra dispenses with further recursion schemes as Oitavem's log-transition recursion, and safe concatenation recursion on notation. Also some initial functions as multiplication, and iterated predecessor can be dropped.

Møller Neergaard introduced a two-sorted characterisation almost exactly corresponding to B using a composition - and a recursion scheme that allows the use of safe variables only once.

Let us prove now the lower bound for $\mathfrak{LS}$.

Lemma 125 *For each logspace function F there exists an $f \in \mathfrak{LS}$ such that $F(\vec{x}) = f(\vec{x};)$ for all $\vec{x} \in \mathbb{W}$.*

Proof.

Let us introduce the word algebra

$$[\epsilon, I, \mathsf{s}_0, \mathsf{s}_1, \mathsf{abs}, \mathsf{bit}, \times, \mathsf{e}, COMP, CRN, SBRN],$$

equivalent to Clote's algebra for logspace [20], where $SBRN$ denotes sharply bounded recursion, given as follows.

$$\begin{array}{rcl} F(\epsilon, \vec{y}) & = & G(\vec{y}) \mid |M(\epsilon, \vec{y})| \\ F(\mathsf{s}_0(x), \vec{y}) & = & H_0(x, \vec{y}, F(x, \vec{y}))) \mid |M(\mathsf{s}_0(x), \vec{y})| \\ F(\mathsf{s}_1(x), \vec{y}) & = & H_1(x, \vec{y}, F(x, \vec{y}))) \mid |M(\mathsf{s}_1(x), \vec{y})|, \end{array}$$

where

$$x|y = \begin{cases} x, \text{ if } x \preceq y \\ y, \text{ else} \end{cases}$$

Our algebra clearly contains the functions $\epsilon, I, \mathsf{s}_0, \mathsf{s}_1, \mathsf{abs}$. Word multiplication is defined via successor, and concatenation as usual for safe function algebras. The eraser is defined as follows using safe case distinction.

$$\begin{aligned} \mathsf{e}(\epsilon;) &:= \epsilon \\ \mathsf{e}(\mathsf{s}_0 w;) &:= \begin{cases} \epsilon, & \text{if } \mathsf{e}(w;) = \epsilon \\ \mathsf{s}_0(;\mathsf{e}(w;)), & \text{else} \end{cases} \\ \mathsf{e}(\mathsf{s}_1 w;) &:= \mathsf{s}_1(;\mathsf{e}(w;)) \end{aligned}$$

$\mathfrak{LS}$ is clearly closed under composition. Next, we prove that the algebra is closed under CRN. We use the following auxiliary function b.

$$\begin{aligned} b(\epsilon; y) &:= \mathsf{s}_0 y \\ b(\mathsf{s}_0 w; y) &:= \mathsf{s}_0 y \\ b(\mathsf{s}_1 w; y) &:= \mathsf{s}_1 y \end{aligned}$$

Let F be defined by concatenation recursion from G, H_0, H_1. Define f as

$$\begin{aligned} f(\epsilon, \vec{x};) &:= g(\vec{x};) \\ f(\mathsf{s}_i w, \vec{x};) &:= b(h_i(w, \vec{x};); f(w, \vec{x};)) \end{aligned}$$

Note that the definitions of b and f are given without using safe case distinction. For bit and $SBRN$, we have to do bootstrapping. The following function $\dot{-}(w; y)$, written as $y \dot{-} w$, is contained in $\mathfrak{LS}$.

$$\begin{aligned} y \dot{-} \epsilon &:= y \\ y \dot{-} \mathsf{s}_i w &:= \mathsf{p_W}(; y \dot{-} w) \end{aligned}$$

Now, we can define the characteristic function of the lexicographic ordering $\preceq$ for two normal arguments w, v. If one of the arguments is larger than the other, we give the suitable output. If both inputs are of equal length, we output $h(w, w, v;)$ where h is defined as follows:

$$\begin{aligned} h(\epsilon, w, v;) &:= 1 \\ h(\mathsf{s}_i x, w, v;) &:= \begin{cases} 0, \text{ if } b(w \dot{-} x; \epsilon) = 1 \wedge b(v \dot{-} x; \epsilon) = 0 \\ 1, \text{ if } b(w \dot{-} x; \epsilon) = 0 \wedge b(v \dot{-} x; \epsilon) = 1 \\ h(x, w, v;), \text{ else} \end{cases} \end{aligned}$$

The necessary case distinctions are justified using the case distinction implicit in the recursion schema. We define *exp* as follows.

$$\begin{aligned} exp(\epsilon, x;) &:= \epsilon \\ exp(\mathsf{s}_i w, x;) &:= \begin{cases} \mathsf{s}_i w, & \text{if } |\mathsf{s}_i w| \preceq x \\ exp(w, x;), & \text{else} \end{cases} \end{aligned}$$

Again, the necessary case distinctions are justified using the case distinction implicit in the recursion schema. The following modified bit function is member of $\mathfrak{LS}$.

$$\mathsf{bit}^*(x, y;) := b(y \dot{-} x; \epsilon)$$

This allows to define the usual bit function.

$$\mathsf{bit}(x, y;) := \mathsf{bit}^*(exp(y, x;), y;)$$

Now, we show that $\mathfrak{LS}$ is closed under sharply bounded recursion. Let F be defined by sharply bounded recursion from G, H_0, H_1 with sharp bound M, where corresponding functions g, h_0, h_1, m are given by induction hypothesis. We show $f \in \mathfrak{LS}$ for a function f such that $F(w, \vec{x}) = |f(w, \vec{x};)|$ which immediately yields the claim. The idea is to determine in each recursion step the length of the recursion argument, which can be assumed to be normal in a certain sense, to apply h_i, and to expand the output using *exp*.

The function $step_i(w, v, \vec{x}; y)$ we will define below allows to transfer the recursion step of F to f. Its intended first input w is a term bounding the recursion. Its second input corresponds to the recursion argument of f. Its safe argument y is intended to equal $f(v, \vec{x};)$, the other arguments represent side arguments of the recursion. $step_i$ counts down its recursion argument w until we arrive at a w' with $|w'| = |y|$. At this point, the step function h_i representing H_i will be applied with last argument $|w'|$. Finally, $f(\mathsf{s}_i v, \vec{x};)$ is constructed using the function *exp* with bound $m(\mathsf{s}_i v, \vec{x};)$ as first argument.

Accordingly, $step_i$ is given as follows for $i = 0, 1$.

$$step_i(\epsilon, v, \vec{x}; y) := exp(m(\mathsf{s}_i v, \vec{x};\,), h_i(v, \vec{x}, \epsilon;\,);\,)$$

$$step_i(\mathsf{s}_j w, v, \vec{x}; y) := \begin{cases} step_i(w, v, \vec{x}; y), \text{ if } y \dot{-} w = \epsilon \\ exp(m(\mathsf{s}_i v, \vec{x};\,), h_i(v, \vec{x}, |\mathsf{s}_j w|;\,);\,), \text{ else} \end{cases}$$

In the definition of $step_i$, we essentially use safe case distinction. The requested function f is defined as follows.

$$f(\epsilon, \vec{x};\,) := exp(m(\epsilon, \vec{x};\,), g(\vec{x};\,);\,)$$
$$f(\mathsf{s}_i v, \vec{x};\,) := step_i(m(v, \vec{x};\,), v, \vec{x}; f(v, \vec{x};\,))$$

This implies the lower bound for $\mathfrak{LS}$. $\square$

Let us switch to the upper bound proof which we prove for an extension $\mathfrak{LS}'$ of $\mathfrak{LS}$ formulated in analogy to Bellantoni's *BC* [6]. As *BC*, also $\mathfrak{LS}'$ separates not only its input but also its output into safe and normal. $\mathfrak{LS}'$ will be useful for the realisation of theories of logspace strength formulated in the next section. In addition, it can be seen easily that Oitavem's algebra $Logspace_{CT}$ can be embedded into $\mathfrak{LS}'$.

Definition 126 *The algebra $\mathfrak{LS}'$ is the smallest function algebra (on words) which contains the following initial functions and is closed under the following operations:*

Initial functions

- ϵ, s_0, s_1, p_W *with safe input and safe output.*
- abs *with normal input and normal output.*
- $\pi_i^{n,m}$ *(projections) with safe output and both normal and safe inputs.*
- $\mathsf{init}(x; y)$ *with normal output which returns the* $\mathsf{abs}(\mathsf{abs}(x))$ *most significant bits of* y*. Usually, we write* y/x *for* $\mathsf{init}(x; y)$.

Operations

- *Composition with*

$$f(\vec{x};\vec{y}) = h(\vec{g}(\vec{x};\vec{y});\vec{j}(\vec{x};\vec{y})),$$

  where the g_i have normal - and the j_i safe output. f has the same sort of outputs as h.

- *Simultaneous safe recursion on notation defined as follows for $1 \leq j \leq m$, $i = 0, 1$*

$$\begin{aligned} f_j(\epsilon,\vec{x};\vec{y}) &:= g_j(\vec{x};\vec{y}) \\ f_j(\mathsf{s}_i w,\vec{x};\vec{y}) &:= h_{j,i}(w,\vec{x}; f_1(w,\vec{x};\vec{y}),\cdots, f_m(w,\vec{x};\vec{y}),\vec{y}), \end{aligned}$$

  where $g_1,\cdots,g_m$, $h_{1,0},\cdots,h_{m,0},h_{1,1},\cdots,h_{m,1}$ have safe output. The f_j have safe output.

- *Raising: from $f(\vec{x};)$ with safe output obtain $f^\nu(\vec{x};)$ with normal output.*

The function algebra $\mathfrak{LS}'$ contains $\mathfrak{LS}$ since it can define the function case using the function init, and a case distinction $d(x;y_1,y_2)$ with

$$d(x;y_1,y_2) := \begin{cases} y_1, & \text{if } x = \epsilon \\ y_2, & \text{else }, \end{cases}$$

which can be defined easily using safe recursion on notation.

Note, that because of projections, functions with normal instead of safe output are admissible in all schemes. $\mathfrak{LS}'$ and BC are formulated very similarly, the most important difference is that in $\mathfrak{LS}'$ only a *sharply* bounded segment of a safe input can by shifted to the normal side, whereas the function *mod* in BC allows to shift a bounded segment. Note that it is important that $\mathfrak{LS}'$ shifts *initial* segments since otherwise an embedding of Cook and Bellantoni's B is clearly possible. The main theorem, stated below, immediately implies that $\mathfrak{LS}'$ is contained in logspace.

Theorem 127 *Let $f(\vec{x};\vec{y})$ be an element of $\mathfrak{LS}'$ where $\vec{y} := y_1,\cdots,y_n$. Then there exist logspace computable functions $ch, del, const$ on words, and unary* *(monotone) polynomials Q, M on words such that for all $\vec{x}, \vec{y} \in \mathbb{W}$ the following properties hold. We write $M(\vec{x})$ for $M(max(\vec{x}))$, and $\vec{y}/M(\vec{x})$ for $y_1/M(\vec{x}), \cdots, y_n/M(\vec{x})$. The output of ch is displayed as a number.*

- $0 \leq ch(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|) \leq n.$
- *If f has normal output, $ch(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|) = 0$.*
- *If $ch(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|) = 0$, we have*
$$f(\vec{x}; \vec{y}) = const(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|).$$
- *If $0 < ch(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|) = k \leq n$, we have*
$$f(\vec{x}; \vec{y}) = \big(y_k \dot{-} del(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|)\big) * const(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|).$$
- *$del(\vec{x}, \cdots), const(\vec{x}, \cdots) \leq Q(\vec{x})$, where $\cdots$ stands for an arbitrary input of fitting size.*

Let us explain now the functions $ch, del, const$ mentioned in the theorem. First, consider the case where $1 \leq ch(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|) = i \leq n$. Then, $f(\vec{x}; \vec{y})$ is calculated by first deleting $|del(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|)|$ bits from y_i. (In the whole proof only the lengths of the *del*-outputs matter.) Then, we concatenate $const(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|)$ (*const* stands for construct). If the value of the choice function is zero, $f(\vec{x}; \vec{y})$ is given as $const(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|)$.

Note that in both cases, $f(\vec{x}; \vec{y})$ fully depends only on a single safe input. For the other safe inputs, only their length, and sharply bounded initial segments matter. Also the chosen safe input can only be manipulated in a very restricted way. This will allow us to simulate safe recursion essentially by sharply bounded recursion since only a very small amount of information about the intermediate values has to be stored.

In the theorem, we additionally claim

$$del(\vec{x}, \cdots), const(\vec{x}, \cdots) \leq Q(\vec{x}).$$

This property is important to prove that the functions in $\mathfrak{LS}'$ have at most polynomial growth. To treat recursion, it is crucial that the polynomials M

and Q do not depend on the safe arguments $\vec{y}$. Let us start with the proof of the theorem.

Proof. Let us first introduce some notations used in this section. We work with the analogues on words of the arithmetical plus and minus operations, given as $+, -$ using again the isomorphism between $(\mathbb{W}, \preceq)$ and $(\mathbb{N}, \leq)$. As for numbers, these operations can be extended within the logarithmic hierarchy to negative words, displayed as $-w$, using a natural coding. Also the ordering $\preceq$ is extended as expected to negative words. We work for technical reasons with a bit function on words, which enumerates the bits in the opposite way as before, which we call bit as well. The most significant bit of w is given as $\mathsf{bit}(0, w)$ and the least significant as $\mathsf{bit}(|w|, w)$.

We prove the claim by induction on the complexity of f, and detail the most interesting steps.

For the initial functions the claim clearly holds. We treat the successor s_i as an example. We define $ch_{\mathsf{s}_i} = 1$, $del_{\mathsf{s}_i} = \epsilon$, $const_{\mathsf{s}_i} = i$. The bounds M and Q are chosen as 1. The predecessor is treated similarly. E.g. for $\mathsf{init}(x; y)$, which has normal output, $const$ is given as y/x.

Composition

Assume that f is defined by composition as follows for $\vec{j}$ containing m components.

$$f(\vec{x}; \vec{y}) = g(\vec{h}(\vec{x}; \vec{y}); \vec{j}(\vec{x}; \vec{y}))$$

Let us first reflect the induction hypothesis for the components of $\vec{j}$. $j_i(\vec{x}, \vec{y})$ for $1 \leq i \leq m$ is given as a sum of at most three summands (always relative to $\dot{-}$ and $*$). If $0 < ch(\vec{x}, \vec{y}/M(\vec{x}), |\vec{y}|) = k \leq n$, this sum is composed of y_k, del_{j_i}, and $const_{j_i}$ (with suppressed inputs). The same argument for the function g implies that $f(\vec{x}; \vec{y})$ is given as a sum of at most 5 summands containing at most two del - and two $const$ summands. The strategy is first to produce these summands as logspace functions of our initial inputs. Then, we combine them to get del_f and $const_f$.

In the following, we define auxiliary functions for arbitrary inputs $\vec{b}, \vec{c}, \vec{d}$. We motivate them for the case that the inputs $\vec{b}, \vec{c}, \vec{d}$ are given as intended, i.e. as $\vec{x}, \vec{y}/M_f(\vec{x}), |\vec{y}|$, where $M_f(x)$ is given as $M_g(Q_h(x))$, where Q_h denotes a bounding polynomial for the bounds Q_{h_i}. The following term equals $h_i(\vec{x};\vec{y})$ for intended inputs according to the induction hypothesis.

$$const_{h_i}(\vec{b},\vec{c}/M_{h_i}(\vec{b}),\vec{d}) := q_i$$

Next, we have to collect information about the $j_i(\vec{x};\vec{y})$. We abbreviate

$$ch_{j_i}(\vec{b},\vec{c}/M_{j_i}(\vec{b}),\vec{d})$$

as k_i. We consider the case $1 \leq k_i \leq n$, and define ℓ_i as follows.

$$\ell_i := max(\epsilon, d_{k_i} - |del_{j_i}(\vec{b},\vec{c}/M_{j_i}(\vec{b}),\vec{d})|),$$

where max denotes a maximum function relying on the lexicographic order extended to negative words, as explained at the beginning of the proof. If $1 \leq k_i \leq n$ does not hold, we define ℓ_i as ϵ. The length of $j_i(\vec{x};\vec{y})$ is given as

$$\tilde{d}_i := \ell_i + |const_{j_i}(\vec{b},\vec{c}/M_{j_i}(\vec{b}),\vec{d})|.$$

We can now define a function F_i which constructs initial segments of $j_i(\vec{x};\vec{y})$ as follows.

$$F_i(\epsilon,\vec{b},\vec{c},\vec{d}) := \epsilon$$

$$F_i(\mathsf{s}_i z,\vec{b},\vec{c},\vec{d}) := \begin{cases} \mathsf{s}_{\mathsf{bit}(|\mathsf{s}_i z|, c_{k_i})} F_i(z,\vec{b},\vec{c},\vec{d}), & \text{if } |\mathsf{s}_i z| \preceq \ell_i \\ \mathsf{s}_{\mathsf{bit}(|\mathsf{s}_i z| - \ell_i, const_{j_i}(\vec{b},\vec{c}/M_{j_i}(\vec{b}),\vec{d}))} F_i(z,\vec{b},\vec{c},\vec{d}), & \text{if } \ell_i \prec |\mathsf{s}_i z| \preceq \tilde{d}_i \\ F_i(z,\vec{b},\vec{c},\vec{d}), & \text{else} \end{cases}$$

Note that an initial segment of $j_i(\vec{x};\vec{y})$ is constructed by first reading bits of the chosen safe input, and then reading the word provided by $const_{j_i}$. For $1 \leq i \leq m$, $F_i(|M_g(\vec{q})|,\vec{b},\vec{c},\vec{d})$ produces initial segments of sufficient length for the later application of g (the q_i are defined above). Now, we can calculate the safe input the function g chooses as follows.

$$ch_g(\vec{q},\vec{F}(|M_g(\vec{q})|,\vec{b},\vec{c},\vec{d}),\vec{\tilde{d}}) := ch_g$$

We are ready to define the searched functions ch_f, del_f, $const_f$. This will be done using a case distinction on the value of ch_g.

First, we assume $1 \leq ch_g \leq m$. Then, $ch_f(\vec{b}, \vec{c}, \vec{d})$ is given as

$$ch_{(j_{ch_g})}(\vec{b}, \vec{c}/M_{j_{ch_g}}(\vec{b}), \vec{d}) := k.$$

This means that the safe input we use as first summand when writing $f(\vec{x}; \vec{y})$ as sum of five summands is the safe input the chosen function j_{ch_g} choses for $\vec{b}, \vec{c}, \vec{d}$ given as intended. $del_f(\vec{b}, \vec{c}, \vec{d})$ is defined as

$$\begin{aligned} & del_{j_k}(\vec{b}, \vec{c}/M_{j_k}(\vec{b}), \vec{d}) * \\ & (del_g(\vec{q}, \vec{F}(|M_g(\vec{q})|, \vec{b}, \vec{c}, \vec{d}), \vec{\tilde{d}}) \dot{-} \\ & const_{j_k}(\vec{b}, \vec{c}/M_{j_k}(\vec{b}), \vec{d})), \end{aligned}$$

$const_f(\vec{b}, \vec{c}, \vec{d})$ as

$$\begin{aligned} & \big(const_{j_k}(\vec{b}, \vec{c}/M_{j_k}(\vec{b}), \vec{d}) \dot{-} \\ & del_g(\vec{q}, \vec{F}(|M_g(\vec{q})|, \vec{b}, \vec{c}, \vec{d}), \vec{\tilde{d}})\big) * \\ & const_g(\vec{q}, \vec{F}(|M_g(\vec{q})|, \vec{b}, \vec{c}, \vec{d}), \vec{\tilde{d}}) \end{aligned}$$

Assume now that $1 \leq ch_g \leq m$ does not hold. Then, $ch_f(\vec{b}, \vec{c}, \vec{d})$ is given as 0. $del_f(\vec{b}, \vec{c}, \vec{d})$ is given as ϵ, and $const_f(\vec{b}, \vec{c}, \vec{d})$ as

$$const_g(\vec{q}, \vec{F}(|M_g(\vec{q})|, \vec{b}, \vec{c}, \vec{d}), \vec{\tilde{d}})$$

Finally, we define the function Q_f as $Q_j(\vec{b}) * Q_g(Q_h(\vec{b}))$ (independent of ch_g), where Q_h, Q_j denote bounding polynomials for the Q_{h_i}, Q_{j_i}, respectively.

Recursion

First, we analyse the case of ordinary safe recursion. Then, it is easy to generalise the argument to simultaneous safe recursion. Assume that f is defined as follows.

$$\begin{aligned} f(\epsilon, \vec{x}; \vec{y}) &:= g(\vec{x}; \vec{y}) \\ f(\mathsf{s}_i z, \vec{x}; \vec{y}) &:= h_i(z, \vec{x}; f(z, \vec{x}; \vec{y}), \vec{y}) \end{aligned}$$

Let us give the intuitive reason why this recursion goes through in logspace. Because our induction hypothesis about g, h_0, h_1, we know that for any $z \in \mathbb{W}$

we can write $f(z, \vec{x}; \vec{y})$ ultimately as sum of at most one safe input and several *del* and *const* terms, depending on the order of applications of h_0, h_1 necessary to calculate $f(z, \vec{x}; \vec{y})$. These terms only depend on the length, and a in z and $\vec{x}$ sharply bounded initial segment of the intermediate values $f(z', \vec{x}; \vec{y})$ with $z' \subseteq z$, which allows to simulate safe recursion using sharply bounded recursion.

Let us give the calculations in more detail. First, we have to make sure until which bit we have to know the safe inputs $\vec{y}$ to compute all necessary $ch, del, const$ terms during the recursion. It is easy to see that a bounding polynomial M_f of M_g, M_{h_0}, and M_{h_1} is sufficient.

We define an auxiliary function $H(w, a, \vec{b}, \vec{c}, \vec{d})$ such that

$$H(w, z, \vec{x}, \vec{y}/M_f(w, \vec{x}), |\vec{y}|) := H$$

contains information about $f(z, \vec{x}; \vec{y})$ for $z \subseteq w$. We sketch the definition of H, and explain the meaning of its output for the intended input, then we give a precise definition of H. The output of H will always be a tuple of five words. The first component is displayed as a number, and tells us for $z = \mathsf{s}_i z' \subseteq w$ which safe argument of $h_i(z', \vec{x}; f(z', \vec{x}; \vec{y}), \vec{y})$ is used when writing $f(z, \vec{x}; \vec{y})$ as sum of three summands using the *del-const* unfolding. For $z = \epsilon$, we use g instead of h_i. If there is no such safe input, we stipulate $H_0 = 0$.

H_1 is displayed as number as well, and tells us which safe input y_ℓ of $\vec{y}$ is needed when we write $f(z, \vec{x}; \vec{y})$ as a totally unfolded *del-const* sum. This means, for $z = \mathsf{s}_i z' \subseteq w$ we write $f(z, \vec{x}; \vec{y})$ first (if possible) as $f(z', \vec{x}; \vec{y})$ minus a *del*, plus an *const* summand, then unfold $f(z', \vec{x}; \vec{y})$ analogously, and so on. If there is no such safe input, we stipulate $H_1 = 0$. For $z = \epsilon$, $H_1 = H_0$.

We assume $z = \mathsf{s}_i z'$ and abbreviate

$$|del_{h_i}(z', \vec{x}, f(z', \vec{x}; \vec{y})/M_{h_i}(z', \vec{x}), \vec{y}/M_{h_i}(z', \vec{x}), |f(z', \vec{x}; \vec{y})|, |\vec{y}|)|$$

as *del*. H_2 is the length of $(f(z', \vec{x}; \vec{y}) \dot{-} del)$ minus the length of y_ℓ, defined above. Note that H_2 is possibly a negative word. If no y_ℓ exists H_2 is ϵ. For $z = \epsilon$ H_2 is given analogously but using del_g, and y_ℓ instead of $f(z', \vec{x}; \vec{y})$.

The fourth component H_3 gives the length of $f(z, \vec{x}; \vec{y})$ minus the length of y_ℓ, or simply the length of $f(z, \vec{x}; \vec{y})$ if y_ℓ does not exist. The reason for giving H_2 and H_3 as lengths relative to the length of y_ℓ is to insure that they can be sharply bounded by a term only depending on the normal arguments of f.

H_4 contains the $||M_f(w, \vec{x})||$ most significant bits of $f(z, \vec{x}; \vec{y})$.

In the following, we give a precise definition of H and argue that it is logspace computable. We define $H(w, a, \vec{b}, \vec{c}, \vec{d})$ by sharply bounded induction on a. We suppress a bound in the following and argue in the end that it can be found easily. We let n denote the number of components of $\vec{c}$ and of $\vec{d}$.

The first and second component of $H(w, \epsilon, \vec{b}, \vec{c}, \vec{d})$ are defined as

$$ch_g(\vec{b}, \vec{c}/M_g(\vec{b}), \vec{d}) := k|n$$

for $|$ being the cut function defined on page 170. We suppress such bounds for the first and second component in the following. For $1 \leq k \leq n$ the third component is given by

$$max(-d_k, -|del_g(\vec{b}, \vec{c}/M_g(\vec{b}), \vec{d})|) := q_2.$$

We abbreviate $const_g(\vec{b}, \vec{c}/M_g(\vec{b}), \vec{d})$ as $const$. The fourth component is given as $q_2 + |const|$.

The fifth component is constructed by glueing c_k and $const$ together, very similarly as on page 177. If $1 \leq k \leq n$ does not hold, we output ϵ, $|const|$, $const/M_f(w, \vec{b})$ as third until fifth component.

Now, we show how to calculate $H(w, \mathsf{s}_i a, \vec{b}, \vec{c}, \vec{d}) := r$ from

$$w, a, \vec{b}, \vec{c}, \vec{d}, H(w, a, \vec{b}, \vec{c}, \vec{d}) := q$$

in logspace. For convenience, in the following for all words i if $1 \leq i \leq n$ does not hold, we let d_i denote ϵ. r_0 is given as

$$ch_{h_i}(a, \vec{b}, q_4/M_{h_i}(a, \vec{b}), \vec{c}/M_{h_i}(a, \vec{b}), d_{q_1} + q_3, \vec{d}) := k.$$

Note that we have inserted the values $q_4/M_{h_i}(a, \vec{b})$ and $d_{q_1} + q_3$ corresponding to a sharply bounded initial segment, and the length of an intermediate value of the recursion for intended inputs. r_1 is given as q_1 if r_0 equals 1, as $r_0 - 1$ if $r_0 > 1$, and as 0 else. For the other components, we use the following case distinction: We first assume that k equals 1. Then, r_2 is given as

$$max(-d_{q_1}, q_3 - |del_{h_i}(a, \vec{b}, q_4/M_{h_i}(a, \vec{b}), \vec{c}/M_{h_i}(a, \vec{b}), d_{q_1} + q_3, \vec{d})|),$$

r_3 is given as

$$r_2 + |const_{h_i}(a, \vec{b}, q_4/M_{h_i}(a, \vec{b}), \vec{c}/M_{h_i}(a, \vec{b}), d_{q_1} + q_3, \vec{d})|.$$

We abbreviate the *const*-term above as *const*. r_4 is defined by glueing together q_4 and *const*, similarly as on page 177.

If we have $2 \leq k \leq n + 1$, the components r_2, r_3, r_4 are given in a very similar way as for $H(w, \epsilon, \vec{b}, \vec{c}, \vec{d})$ since in this case we build a new *del-const* sum from a safe input. *In all other cases*, we output ϵ, $|const|$, $const/M_f(w, \vec{b})$ as r_2, r_3, r_4.

Let us finally say a word about the sharp bound we have to deliver for this recursion. It suffices to show that all components of $H(w, a, \vec{b}, \vec{c}, \vec{d})$ are sharply bounded by polynomials of $w, a, \vec{b}$ since we are using a linear pairing function. For the first, second, and fifth component this is clear. For the third and fourth component, we use a bounding polynomial

$$\left[Q_g(\vec{b}) * (Q_h(a, \vec{b}) \times a)\right] \times c,$$

where the constant c depends on the exact definition of the coding for negative words. This concludes the proof that the auxiliary function H is logspace computable.

It is easy to define ch_f, del_f, $const_f$ from H: $ch_f(a, \vec{b}, \vec{c}, \vec{d})$ is simply given as $H(a, a, \vec{b}, \vec{c}, \vec{d})_1 := k$. del_f and $const_f$ are constructed using a case distinction on k. First, we assume $1 \leq k \leq n$. Let us find the smallest $z \subseteq a$ such that $z \subset z' \subseteq a$ implies $H(a, z', \vec{b}, \vec{c}, \vec{d})_0 = 1$ [5]. For $z \subseteq z' \subseteq a$, $f(z', \vec{x}; \vec{y})$ is given as a *del-const* sum starting at y_k. Find the minimal

$$H(a, z', \vec{b}, \vec{c}, \vec{d})_2$$

[5]Sharply bounded quantification which immediately yields subword quantification is admissible within the logarithmic hierarchy, see Clote's [20].

for $z \subseteq z' \subseteq w$ which we abbreviate as q. $del_f(a, \vec{b}, \vec{c}, \vec{d})$ is given as

$$exp(Q_f(w, \vec{x}), max(\epsilon, -q)),$$

where exp is defined on page 172.

Next, we show how $const_f(a, \vec{b}, \vec{c}, \vec{d})$ can be calculated in logspace, by calculating its i-th bit in logspace for an arbitrary i. We abbreviate $|y_k \dot{-} del_f(a, \vec{b}, \vec{c}, \vec{d})|$ as ℓ. We search the largest $z \subseteq z' \subseteq w$ such that

$$d_k + H(a, z', \vec{b}, \vec{c}, \vec{d})_2 \prec \ell + i \preceq d_k + H(a, z', \vec{b}, \vec{c}, \vec{d})_3.$$

If no such z' exists, i exceeds the length of $const_f(a, \vec{b}, \vec{c}, \vec{d})$. Assume $z' = \mathsf{s}_i v$. Then, we easily find the value of the searched i-th bit by calculating

$$const_{h_i}(v, \vec{b}, r_4/M_{h_i}(v, \vec{x}), \vec{y}/M_{h_i}(v, \vec{x}), d_k + r_3, \vec{d}),$$

where $r := H(a, v, \vec{b}, \vec{c}, \vec{d})$. If $z' = \epsilon$, we use $const_g$ instead.

If $1 \leq k \leq n$ does not hold, $const_f$ is defined similarly, and del_f as ϵ. This finishes our argument for the ordinary safe recursion.

Simultaneous safe recursion

We sketch how to produce the functions ch_{f_ℓ}, del_{f_ℓ}, $const_{f_\ell}, M_{f_\ell}, Q_{f_\ell}$ for $1 \leq \ell \leq m$ if f_ℓ is defined by simultaneous safe recursion.

- For all $1 \leq \ell \leq m$, we give the same M_{f_ℓ}, Q_{f_ℓ} which we call M_f, Q_f. We define them as for ordinary safe recursion using bounds that work for all base - and recursion step functions.

- We use again an auxiliary function H'. It contains the same information as H but simultaneously for all f_ℓ with $1 \leq \ell \leq m$. So, e.g., we collect the first $||M_f(w, \vec{x})||$ bits for all f_ℓ. The components of H' can be calculated very similarly as the ones of H.

- ch_{f_ℓ} is given analogously as before. For del_{f_ℓ}, and $const_{f_\ell}$ the complication is that it is not sufficient to know which y_k occurs as first summand in the total unfolding of $f_\ell(z, \vec{x}; \vec{y})$ as sum of del and $const$ terms. This is so because it does not give us the information in which order which base - and recursion step functions were applied. The problem is solved by tracing back which safe input is needed to write $f_\ell(z, \vec{x}; \vec{y})$ as a *del-const* sum using $H(z, a', \vec{b}, \vec{c}, \vec{d})_0$ for $a' \subseteq z$. Then, following this trace, del_{f_ℓ} and $const_{f_\ell}$ are defined very similarly as before. The back-tracing is possible using constantly bounded recursion.

This concludes the proof of the theorem. $\Box$

The lower bound result implies together with the previous theorem the following corollary.

Corollary 128 *The normal segments of the algebras $\mathfrak{LS}$ and $\mathfrak{LS}'$ describe exactly the logspace computable functions.*

The technique used to show that $\mathfrak{LS}'$ has strength logspace can also be directly applied to $\mathfrak{LS}$. The dependence of *ch*, *del*, and *const* on sharply bounded initial segments of safe inputs can then be dropped. This immediately implies that Oitavem's log-transition recursion cannot be defined in $\mathfrak{LS}$. Nevertheless, note that it can be defined easily in $\mathfrak{LS}'$. This immediately implies that Oitavem's algebra $Logspace_{CT}$ is contained in $\mathfrak{LS}'$.

6.5 Two systems of strength logspace

We formalise the algebra $\mathfrak{LS}$ and Clote's function algebra for logspace [20] with concatenation and sharply bounded recursion within an applicative setting. The theories again contain a predicate for normal - and a predicate for safe words, interpreted similarly as before. In contrast to the theories presented earlier, for logspace strength we allow ordinary one-variable induction schemes.

6.5.1 Formalising $\mathfrak{LS}$

We introduce the theory LogST, formalising $\mathfrak{LS}$, and prove its lower bound in terms of provably total functions. We deliver the upper bound proof for a stronger system LogST' formalising $\mathfrak{LS}'$.

Definition 129 *The theory* LogST *is the theory* LogT *with the following modifications.*

- *The induction axiom is replaced by* V*-induction, defined as follows for* $x \notin FV(r)$*:*

$$r\epsilon \in \mathsf{V} \land (\forall x \in \mathsf{W})(rx \in \mathsf{V} \to r(\mathsf{s}_i x) \in \mathsf{V}) \to (\forall x \in \mathsf{W})(rx \in \mathsf{V})$$

- *We drop the axioms for bit, concatenation, multiplication, and eraser.*

- *Let* case *be a closed term corresponding to the function* $\mathsf{case} \in \mathfrak{LS}$ *for which the usual elementary properties are provable in* LogT*. Then we have as additional axioms the following.*

  - $x, y, z \in \mathsf{V} \to \mathsf{case}(x, y, z) \in \mathsf{V}$
  - $x, y \in \mathsf{V} \to \mathsf{case}(x, y, \epsilon) = x$
  - $x, y, z \in \mathsf{V} \to (\mathsf{case}(x, y, z) = y \lor z = \epsilon)$

The following lemma is proved by straightforward induction on the complexity of F.

Lemma 130 *For any* $F(\vec{x}; \vec{y}) \in \mathfrak{LS}$ *there is an* L *term* t_F *with*

- $\mathsf{LogST} \vdash \vec{x} \in \mathsf{W} \land \vec{y} \in \mathsf{V} \to t_F(\vec{x}, \vec{y}) \in \mathsf{V}$

- $\mathcal{M}(\lambda\eta) \models t_F(\vec{\overline{w}}, \vec{\overline{v}}) = \overline{F(\vec{w}; \vec{v})}$*, where* $\mathcal{M}(\lambda\eta)$ *denotes the standard open term model.*

Now, we switch to the theory LogST' formalising $\mathfrak{LS}'$ which is formulated with a more flexible induction scheme.

Definition 131 *The theory* LogST' *is the theory* LogT *with the following modifications.*

- *The induction axiom is replaced by positive* W*-free induction.*

- *We drop the axioms for bit, concatenation, multiplication, and eraser.*

- *Let* init *be a closed term corresponding to the function* $\mathsf{init} \in \mathfrak{LS}'$ *for which the usual elementary properties are provable in* LogT*. Then we have as additional axiom the following.*

$$x \in \mathsf{W} \wedge y \in \mathsf{V} \rightarrow \mathsf{init}(x, y) \in \mathsf{W}$$

A term case' corresponding to case in LogST can be defined in LogST' as follows.

$$\mathsf{case}' := \lambda x.\lambda y.\lambda z.\mathsf{d}_{\mathsf{W}}(x, y, \epsilon, \mathsf{init}(1, z))$$

The upper bound proof for LogST' is technically involved. The main problem is, that a pairing function p for safe inputs is not available. This is a consequence of theorem 127 as we will argue in the following. Assume that p is such a pairing function. Independently of whether we can write $\mathsf{p}(; y_1, y_2)$ according to the theorem as *const* term, or as sum of three summands, it can be bounded by one of its safe inputs concatenated with a fixed polynomial in the length of both safe inputs. This condition cannot be fulfilled for arbitrary y_1, y_2.

Let us now point out, why it is a problem not to have a pairing function for safe inputs: Complex formulas without occurrences of W have to be realised by a safe input of the realisation function to allow induction. Nevertheless, without pairing - and projection functions, we are unable to access the realisers of the components of the formula which makes the realisation approach impossible.

Cantini presented in [17] a realisation approach that also handles the problem of the missing pairing function for safe inputs. Nevertheless, we present an alternative realisation approach, which deals with all V-atoms separately, and uses a set of realisation functions such that each yields only the realiser of a single V-atom. In contrast to Cantini's approach, vector-valued functions are not necessary, and also proofs containing formulas with both, W and V-occurrences, can be treated. This allows the realisation of a more general V-elimination rule in comparison to Cantini's [17]. The presented approach also solves a technical problem Cantini's approach faces for the realisation of disjunctions whose realisers may have different types.

Let us give the details of the realisation approach. We number the V-atoms of a formula D from the left to the right and call the k-th such atom D_k. If $\rho \ \mathfrak{r} \ D$ or $\rho \ \mathfrak{R} \ D$ holds, for the realisation relations $\mathfrak{r}$ and $\mathfrak{R}$ defined on page 160, we can speak of the set of realised atoms as on page 161. We write $\rho \ \mathfrak{r}^\star \ D_k$ or $\rho \ \mathfrak{R}^\star \ D_k$, respectively, if D_k belongs to the set of realised atoms. If $\rho \ \mathfrak{R} \ D$ holds, we also use the expression "$\rho \ \mathfrak{R}^\star \ D_k$ by w" for a word w if w is the component of ρ realising the atom D_k.

We define a realisation relation $\mathcal{S}$ for sequents which realises the V-atoms separately. We let $< \cdot, \cdot >$ denote a standard set theoretic pairing function.

Definition 132 *Let Γ be given as $A_1, \cdots, A_n$ (with V-atoms given as $A_{i,1}, \cdots, A_{i,k_i}$). $\rho \ \mathcal{S} \ \Gamma$ holds exactly if*

- *ρ is of the form*

$$<< v_1, \cdots, v_n >, < w_{1,1}, \cdots, w_{1,k_1}, \cdots, w_{n,k_n} >>$$

  where k_i equals the number of V-atoms of A_i for each $1 \leq i \leq n$. (k_i might equal zero.)

- $< v_1, \cdots, v_n > \ \mathfrak{r} \ \Gamma$.

- *For any $1 \leq i \leq n$ there is a (unique) $\wp$ with $v_i = \wp^{A_i}$ and $\wp \ \mathfrak{R} \ A_i$ such that*

$$(v_i \ \mathfrak{r}^\star \ A_{i,j} \Leftrightarrow) \wp \ \mathfrak{R}^\star \ A_{i,j} \Leftrightarrow \wp \ \mathfrak{R}^\star \ A_{i,j} \text{ by } w_{i,j}.$$

Note that the use of two realisation relations $\mathfrak{r}$ and $\mathfrak{R}$ and their interplay using the projection function was already used in section 6.3 for the realisation of the weaker theories. Here, the new idea is the individual realisation of V-atoms.

Note that for formulas A_i without occurrence of V the third property is fulfilled for $\wp = v_i$ since the biconditionals hold trivially in this case as no $A_{i,j}$ exist. For the realisation relation $\mathcal{S}$ the usual properties with respect to quantification and equality of terms hold as a consequence of these properties for $\mathfrak{R}$, $\mathfrak{r}$.

Lemma 133 *For the realisation relation* $\mathcal{S}$ *the following properties hold. We use* $\vec{s} = \vec{t}$ *as an abbreviation of* $s_0 = t_0 \wedge \cdots \wedge s_n = t_n$.

- $\rho\ \mathcal{S}\ (\exists x)A[x] \Leftrightarrow \rho\ \mathcal{S}\ A[t]$ *for some term* t
- $\rho\ \mathcal{S}\ (\forall x)A[x] \Leftrightarrow \rho\ \mathcal{S}\ A[u]$
- $\rho\ \mathcal{S}\ A[\vec{s}] \Leftrightarrow \rho\ \mathcal{S}\ A[\vec{t}]$

Proof. Assume $\rho\,\mathcal{S}\,(\exists x)A[x]$. The realisation property 3 implies the existence of a $\wp$ such that $\wp\ \mathfrak{R}\ (\exists x)A[x]$ and $\wp^{(\exists x)A[x]} = \rho_{0,0}$. This implies $\wp\ \mathfrak{R}\ A[t]$ for some term t. The properties of the projection function imply $\wp^{A[t]} = \rho_{0,0}$ and $\rho_{0,0}\ \mathfrak{r}\ A[t]$. Therefore, realisation properties 2 and 3 hold relative to t. Also property 1 clearly holds. This implies the direction from left to right for the first claim. The other direction and the other claims clearly hold because of the analogous properties for $\mathfrak{R}$, $\mathfrak{r}$ and the properties of the projection function. $\Box$

Note that the lemma above also holds with side formulas present.

We are ready to state the main theorem for a total sequent-style formalisation of LogST, called LogST as well, which is constructed as for the theories presented in section 2. An $\mathcal{S}$ -realiser $\rho := <\rho_0, \rho_1>$ is inserted into realisation functions of $\mathfrak{L}\mathfrak{S}$ as

$$(\rho_{0,0}, \rho_{0,1}, \cdots ; \rho_{1,0}, \rho_{1,1}, \cdots)$$

which we abbreviate as $(\rho_0; \rho_1)$.

Theorem 134 *Assume that* Γ *and* $\Delta \equiv D_1, \cdots, D_m$ *(with* V*-atoms given as* $D_{i,1}, \cdots, D_{i,k_i}$*) are positive sequences of formulas. Assume* LogST $\vdash \Gamma \Rightarrow \Delta$ *with a proof containing only positive formulas. Then there are* $\mathfrak{L}\mathfrak{S}$*-functions* F *with normal output and* $f_{<1,1>}, \cdots, f_{<1,k_1>}, \cdots, f_{<m,k_m>}$ *with safe outputs such that for all substitutions* $[\vec{s}]$ *and for all* $\rho\ \mathcal{S}\ \Gamma[\vec{s}]$ *the following properties hold.*

- $1 \leq F(\rho_0;\rho_1)_0 := i \leq m$
- $<< F(\rho_0;\rho_1)_1 >, < f_{i,1}(\rho_0;\rho_1), \cdots, f_{i,k_i}(\rho_0;\rho_1) >>\ \ \mathcal{S}\ D_i$

Proof. We use an induction on the length of the positive proof.

Logical and applicative axioms

The logical axioms are realised easily, and the applicative axioms as usual since they do not contain V.

V axioms

- $x \in \mathsf{V} \Rightarrow \mathsf{s}_i x \in \mathsf{V}$
  We take F as the constant $\langle 1, \epsilon \rangle$ function. We define
  $$f_{1,1}(\rho_0; \rho_1) := \mathsf{s}_i \rho_1.$$
  The predecessor is realised similarly.

- $x \in \mathsf{W}, y \in \mathsf{V} \Rightarrow \mathsf{init}(x, y) \in \mathsf{W}$
  We define $F(\rho_{0,0}, \rho_{0,1}; \rho_{1,0})$ as $\langle 1, \mathsf{init}(\rho_{0,0}; \rho_{1,0}) \rangle$. There are no other realisation functions since the succedent does not contain V.

- $x \in \mathsf{W} \Rightarrow x \in \mathsf{V}$
  We take F as the constant $\langle 1, \epsilon \rangle$ function. We define $f_{1,1}$ as identity.

V-elimination

Let us realise V-elimination given as follows where Γ, Δ does not contain V, and B_{W} is B with all V replaced by W.

$$\frac{\Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow B_{\mathsf{W}}, \Delta}$$

We assume realisation functions $G, g_{1,1}, \cdots$ for the premise. Let us define F by the following algorithm for a given input $(\rho_0;)$. Let $G(\rho_0;)_0 = 1$. For each $1 \le i \le k_1$, replace the ϵ in $G(\rho_0;)_1$ responsible for the i-th V-atom by $g_{1,i}(\rho_0;)$ if there is such an ϵ. (The relevant ϵ's can be found by only considering the structure of $G(\rho_0;)_1$.) The resulting word α is clearly an $\mathfrak{r}$ realiser of B_{W}. Therefore, we define $F(\rho_0;)$ as $\langle 1, \alpha \rangle$ in this case. If $G(\rho_0;)_0 \neq 1$ we define $F(\rho_0;)$ just as $G(\rho_0;)$.

Let us argue that the function F is in $\mathfrak{LS}$ which follows from α being producible from ρ_0 within $\mathfrak{LS}$: $G(\rho_0;)_1$ delivers normal output, and the $g_{1,i}(\rho_0;)$ can be assumed to deliver normal output as well because of the raising rule.

Therefore, the lower bound lemma for $\mathfrak{LS}$ implies that their outputs can be freely used as input for logarithmic space functions. For each $1 \leq i \leq k_1$, we can find the ϵ responsible for the corresponding atom, if it exists, within the logarithmic hierarchy, since this only involves keeping track of the structure of $G(\rho_0;)_1$ relative to the pairing function. Once the positions of the ϵ's are found, the replacements can be clearly executed within the logarithmic hierarchy. Finally, the $f_{i,j}$ are found easily [6].

$\wedge$ right rule

Let the applied rule have the following form.

$$\frac{\Gamma \Rightarrow A, \Delta \quad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \wedge B, \Delta}$$

We assume that A has n and B m V-atoms. We assume realisation functions $G, g_{1,1}, \cdots, H, h_{1,1}, \cdots$ for the premisses, and $\rho\ \mathcal{S}\ \Gamma[\vec{s}]$. Assume first $G(\rho)_0 = 1$ and $H(\rho)_0 = 1$. Then $F(\rho)$ is given as $\langle 1, \langle G(\rho), H(\rho)\rangle\rangle$. The $f_{i,j}$ are given as follows.

- If $i = 1 \wedge j \leq n$ we have $f_{i,j} = g_{1,j}$.
- If $i = 1 \wedge n < j \leq n + m$ we have $f_{i,j} = h_{1,j-n}$.
- Else, we define $f_{i,j}$ arbitrary.

Assume $G(\rho)_0 = i \neq 1$. Then F is given as G, $f_{i,j}$ is given as $g_{i,j}$, and the other $f_{k,\ell}$ are given arbitrary. Analogously for the case $G(\rho)_0 = 1$ and $H(\rho)_0 \neq 1$.

$\vee$-left rule

Let the applied rule have the following form.

$$\frac{\Gamma, A \Rightarrow \Delta \quad \Gamma, B \Rightarrow \Delta}{\Gamma, A \vee B \Rightarrow, \Delta}$$

[6]Note that the realisation of V-elimination is not entirely trivial, because of the asymmetric treatment of the W and V-atoms in our approach. It would be possible to modify it such that single W-atoms are realised by functions with normal instead of safe outputs which would yield a trivial realisation of V-elimination.

We assume that Γ has $n-1$ formulas that A has m_0 and B m_1 V-atoms. We assume realisation functions $G, g_{1,1}, \cdots,$ $H, h_{1,1}, \cdots$ for the premisses, and $<< v_1, \cdots, v_n >; < w_{1,1}, \cdots, w_{n,1}, \cdots, w_{n,m_0+m_1} >>$ $\mathcal{S}$ $\Gamma, A \vee B[\vec{s}]$. Assume that v_n realises the first disjunct, the other case is treated analogously. In this case F is given as follows.

$$F(v_1, \cdots, v_n; w_{1,1}, \cdots, w_{n,1}, \cdots, w_{n,m_0+m_1}) := \\ G(v_1, \cdots, (v_n)_0; w_{1,1}, \cdots, w_{n,1}, \cdots, w_{n,m_0}).$$

Analogously for the $f_{i,j}$.

Cut

Assume that the cut rule is given as follows, where we have premise realisation functions $G, g_{1,1}, \cdots,$ $H, h_{1,1}, \cdots$.

$$\frac{\Gamma \Rightarrow A, \Delta \qquad \Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$$

Then the function F is defined as follows.

$$F(\vec{x}; \vec{y}) := \begin{cases} H(\vec{x}, G(\vec{x}; \vec{y})_1; \vec{y}, \vec{g_1}(\vec{x}; \vec{y})), & \text{if } G(\vec{x}; \vec{y})_0 = 1 \\ G(\vec{x}; \vec{y}), & \text{else} \end{cases}$$

The functions $f_{i,j}$ are defined as follows.

$$f_{i,j}(\vec{x}; \vec{y}) := \begin{cases} h_{i,j}(\vec{x}, G(\vec{x}; \vec{y})_1; \vec{y}, \vec{g_1}(\vec{x}; \vec{y})), & \text{if } G(\vec{x}; \vec{y})_0 = 1 \\ g_{i,j}(\vec{x}; \vec{y}), & \text{else} \end{cases}$$

The correctness of the realisation functions in the case where the cut formula is realised follows easily from

$$<< \vec{x}, G(\vec{x}; \vec{y})_1 >, < \vec{y}, \vec{g_1}(\vec{x}; \vec{y}) >> \ \mathcal{S} \ \Gamma, A[\vec{s}],$$

which is an immediate consequence of $<< \vec{x} >, < \vec{y} >>$ $\mathcal{S}$ $\Gamma[\vec{s}]$.

Induction

Induction is realised using the scheme of simultaneous recursion. Assume that the rule has the following form, where we have premise realisation functions $G, g_{1,1}, \cdots, H_i, h_{i,1,1}, \cdots$.

$$\frac{\Gamma \Rightarrow A[0], \Delta \qquad \Gamma, x \in \mathsf{W}, A[x] \Rightarrow A[\mathsf{s}_i x], \Delta}{\Gamma, t \in \mathsf{W} \Rightarrow A[t], \Delta}$$

Remember, that we use the standard linear pairing operation $\langle , \rangle$ which is in the logarithmic hierarchy. To determine, whether the first element of a certain pair is 1, we have to know only finitely many initial bits. Therefore, case distinctions with this property are permitted for safe inputs.

By lemma 122, we find a word c such that for all substitutions $[\vec{s}]$

$$\rho \ \mathfrak{r} \ A[\vec{s}] \Rightarrow \langle 1, \rho \rangle \leq |c|.$$

We define an auxiliary function F', motivated as follows. We keep its output small such that we can transform it into a normal output as demanded for F. This works without a problem as long as main formulas is realised. If in turn a side formula is realised, the produced realiser does not have to be sharply bounded. Therefore, in this case, F' only stores the information in which induction step for the first time a side formula is realised. Now, we define F', and the functions $f_{i,j}$ by simultaneous safe recursion. We abbreviate

$$H_i(\vec{x}, w, \mathsf{init}(c, F'(\vec{x}, w; \vec{y}))_1; \vec{y}, \vec{f_1}(\vec{x}, w; \vec{y}))$$

as $Q_i(\vec{x}, w; \vec{y})$.

$$F'(\vec{x}, \epsilon; \vec{y}) := \begin{cases} G(\vec{x}; \vec{y}), & \text{if } G(\vec{x}; \vec{y})_0 = 1 \\ \langle 2, \epsilon \rangle, & \text{else} \end{cases}$$

$$F'(\vec{x}, \mathsf{s}_i w; \vec{y}) := \begin{cases} Q_i(\vec{x}, w; \vec{y}), & \text{if } F'(\vec{x}, w; \vec{y})_0 = 1 \wedge Q_i(\vec{x}, w; \vec{y})_0 = 1 \\ \langle 3, |w| \rangle, & \text{if } F'(\vec{x}, w; \vec{y})_0 = 1 \wedge Q_i(\vec{x}, w; \vec{y})_0 \neq 1 \\ F'(\vec{x}, w; \vec{y}), & \text{else} \end{cases}$$

We abbreviate

$$h_{i,j,\ell}(\vec{x}, w, \mathsf{init}(c, F'(\vec{x}, w; \vec{y}))_1; \vec{y}, \vec{f_1}(\vec{x}, w; \vec{y}))$$

as $q_{i,j,\ell}(\vec{x}, w; \vec{y})$.

$$f_{j,\ell}(\vec{x}, \epsilon; \vec{y}) := g_{j,\ell}(\vec{x}, \epsilon; \vec{y})$$
$$f_{j,\ell}(\vec{x}, \mathsf{s}_i w; \vec{y}) := \begin{cases} q_{i,j,\ell}(\vec{x}, w; \vec{y}), & \text{if } F'(\vec{x}, w; \vec{y})_0 = 1 \\ f_{j,\ell}(\vec{x}, w; \vec{y}), & \text{else} \end{cases}$$

Finally, let us define the realisation function F with normal output. Let p be a polynomial such that $\langle 2, |w|\rangle \leq |p(w)|$. We define

$$r(\vec{x}, w; \vec{y}) := exp(w, \mathsf{init}(p(w); F'(\vec{x}, w; \vec{y}))_1;).$$

We let $F(\vec{x}, \mathsf{s}_i w; \vec{y})$ be given by the following case distinction.

$$\begin{cases} \mathsf{init}(c, F'(\vec{x}, \mathsf{s}_i w; \vec{y})), & \text{if } F'(\vec{x}, \mathsf{s}_i w; \vec{y})_0 = 1 \\ G(\vec{x}; \vec{y}), & \text{if } F'(\vec{x}, \mathsf{s}_i w; \vec{y})_0 = 2 \\ H_i(\vec{x}, r(\vec{x}, \mathsf{s}_i w; \vec{y}), \mathsf{init}(c, F'(\vec{x}, r(\vec{x}, \mathsf{s}_i w; \vec{y}); \vec{y})); \vec{y}), & \text{if } F'(\vec{x}, \mathsf{s}_i w; \vec{y})_0 = 3 \end{cases}$$

The correctness of the realisation functions is proved by an easy induction on the value of t in the standard model.

Other rules

The structural rules, the $\vee$-right -, and the $\wedge$-left rule are realised easily. For the quantifier rules, we use lemma 133. $\square$

From lemma 130 and the previous theorem we derive the following lemma.

Lemma 135 *The theories* LogST *and* LogST' *prove totality exactly for the logarithmic space computable functions.*

6.5.2 Formalising Clote's algebra for logspace

In the last section, we have defined a theory of strength logspace that formalises the algebra $\mathfrak{LS}$. Another possibility to produce a theory of this strength is to formalise the already mentioned algebra

$$[0, I, \mathsf{s}_0, \mathsf{s}_1, \mathsf{abs}, \mathsf{bit}, \mathsf{e}, \times, COMP, CRN, SBRN],$$

where $SBRN$ denotes sharply bounded recursion. We give an induction principle capturing both, CRN and $SBRN$.

Definition 136 *For any positive formula A, we denote the formula A with each subformula of the form $t \in \mathsf{W}$ replaced by $t \leq_{\mathsf{W}} u$ by A^u.*

Definition 137 *The theory* LogSB *is the theory* LogT *with the following modifications.*

- *The induction axiom is replaced by sharply bounded induction (SB-Ind) defined as follows, for A a positive formula and u a fresh variable.*

$$u \in \mathsf{W} \to \Big(A^{|u|}[\epsilon] \land (\forall x \in \mathsf{W})(A^{|u|}[x] \to A^{|u|}[\mathsf{s}_0 x] \land A^{|u|}[\mathsf{s}_1 x]) \to (\forall x \in \mathsf{W})(A^{|u|}[x])\Big)$$

- *We drop the axioms for bit, concatenation, multiplication, and eraser.*

Lemma 138 *The theory* LogSB *proves totality exactly for the logspace computable functions.*

Proof. The theory LogST′ can simulate induction over formulas of the form $y \leq_{\mathsf{W}} |x|$, since it proves

$$x \in \mathsf{W} \to (y \leq_{\mathsf{W}} |x| \leftrightarrow y \in \mathsf{V} \land \mathsf{init}(x, y) = y)$$

because of the elementary properties and the axiom for init. This immediately implies the upper bound using theorem 127.

For the lower bound, we show that the logspace functions are provably total by induction on their complexity using the earlier mentioned function algebra

$$[\epsilon, I, \mathsf{s}_0, \mathsf{s}_1, \mathsf{abs}, \mathsf{bit}, \times, \mathsf{e}, COMP, CRN, SBRN].$$

The totality of $\epsilon, I, \mathsf{s}_0, \mathsf{s}_1, \mathsf{abs}$ are clear. The totality of word multiplication follows as for LogST. For the definition of the eraser, we use the following auxiliary function h.

$$h(w) := \begin{cases} 1, & \text{if } w \text{ contains a } 1 \\ \epsilon, & \text{else} \end{cases}$$

The totality of h is proved by $(SB - Ind)$. Then the eraser function is given by a term e fulfilling the following recursion equations.

$$\begin{aligned}
\mathsf{e}(\epsilon;) &:= \epsilon \\
\mathsf{e}(\mathsf{s}_0 w;) &:= \begin{cases} \mathsf{s}_0\mathsf{e}(w), & \text{if } h(\mathsf{s}_0 w) = 1 \\ \mathsf{e}(w), & \text{else} \end{cases} \\
\mathsf{e}(\mathsf{s}_1 w;) &:= \mathsf{s}_1\mathsf{e}(w)
\end{aligned}$$

Its totality is proved by V-induction.

Next, we show how the totality of the bit function is proved. We prove consecutively the totality of the functions $\dot{-}$, h, exp, bit^* defined in the lower bound proof of $\mathfrak{LS}$ on page 170. The totality of all of these functions is proved by V-induction in LogSB because only case distinction over elements of W are necessary. Then, we define bit using bit^*, and exp.

We sketch in the following how to deal with the recursion schemes. Assume that t_F represents a function defined by concatenation recursion. The totality of t_F is proved using (SB-Ind) with induction variable x for the formula $t_F x\vec{z} \in \mathsf{V}$ and the V-elimination rule, where we assume $\vec{z} \in \mathsf{W}$. To prove totality of a function F represented by t_F defined by sharply bounded recursion with bound B represented by t_B, we use induction over initial segments $\{w \in \mathbb{W} | w \subseteq a\}$ of W with induction variable x for the formula $t_F x\vec{z} \leq_{\mathsf{W}} |t_B a\vec{z}|$ [7]. This concludes the proof of the lower bound. $\square$

6.6 Summary

We have shown that the introduction of two distinct word predicates W and V, first presented by Cantini [17], can be used nicely to reflect the different roles inputs play in weak recursion schemes. We propose a new reading of $t \in \mathsf{V}$ as "t is inaccessible" which is reflected by restricting the application of initial functions to elements of V.

We presented a restricted case distinction for safe inputs and have shown that it allows a simple two-sorted characterisation of logspace. Interestingly,

[7]We can assume that the bound B is monotone

the restriction of case distinction has a similar effect on the safe recursion scheme as affinity restrictions (see Neergaard's [66]).

Chapter 7

Concluding remarks and future research

In chapters 2,3,4, and 5, we demonstrated that the use of second-order notions like types and truth allows the definition of natural theories of high expressive power. For the second-order theories T_{PT}, $\mathcal{U}(\mathsf{FEA})$, and $\mathcal{U}_{\mathsf{T}}(\mathsf{FEA})$ introduced in our thesis on pages 21, 131, and 133, already the restriction to their first-order, purely combinatorial part reaches polynomial strength. In future research, it would be interesting to see, whether the expressive strength of flexible second-order induction schemes yields polynomial strength starting from weaker applicative first-order theories. We could e.g. drop polynomially growing initial functions. We would also like to analyse, which second order principles of these theories are needed exactly to reach polynomial strength.

We strongly assume that already a weakend version ${\mathsf{T}_{\mathsf{PT}}}^-$ of T_{PT} with the reflection principle for initial segments of the words is dropped, proves totality for all polynomial time computable relations. In ${\mathsf{T}_{\mathsf{PT}}}^-$ induction on the variable x over formulas of the form $x \in \mathsf{W} \wedge x \leq w$ with w being a word is not allowed in general. This would be in contrast to all other until now introduced applicative theories of this strength, given in [76, 79, 74], which more or less explicitly allow such inductions. The important difference between ${\mathsf{T}_{\mathsf{PT}}}^-$ and the systems given above is its more flexible induction scheme: the representation r of the induction predicate in $\mathsf{T}(rx)$ does not need to represent the same formula $A[x]$ for all inputs x, it can e.g. represent increasingly large conjunctions depending on x. Using this property it seems to be possible to simulate a Turing machine computing a polytime relation within the weakened theory.

We also imagine that these new ideas give rise to an alternative unfolding program for a base theory even more limited than the one introduced in chapter 5. The second unfolding would nevertheless reach polynomial strength.

The outlines given above might work for other complexity classes by varying the induction scheme. We conjecture a theory with the following scheme

$$\mathsf{T}(r\epsilon) \wedge (\forall x \in \mathsf{W})(\mathsf{T}(rx) \rightarrow \mathsf{T}(\mathsf{s}_\ell x)) \rightarrow (\forall x \in \mathsf{W})\mathsf{T}(x)$$

to prove totality for functions computable in exponential time, where s_ℓ denotes the lexicographic successor. Similarly as sketched above, one should be able to obtain an unfolding of exponential strength, using a base theory featuring lexicographic induction. The proof of these conjectures would show that second-order notions as types and truth in an applicative setting do not only increase the expressive power, but also add computational power to natural base theories.

In chapter 6, we highlighted the close connection between implicit characterisations of complexity classes by function algebras, and by applicative theories, respectively. Our characterisations of complexity classes by logical systems often presupposed existing function algebras. In contrast, the new function algebra characterisation of logspace given on page 168 is inspired by allowing less restrictive induction schemes than for the theories LogT and AlogT of logarithmic strength introduced earlier in chapter 6. It would be interesting to work out the connection between logical systems and function algebras in greater detail; logical systems could e.g. suggest new function algebras operating on alternative data types such as list and sets. Especially, the way in which the above mentioned weakened version of $\mathsf{T_{PT}}^-$ seems to achieve polynomial strength inspires the definition of new function algebraic characterisations of polynomial time, using words and nested lists as objects of computation.

Speaking very roughly, in our new algebra on words and lists, safe, respectively normal arguments behave like formulas of the form $\mathsf{T}(t)$, respectively

$t \in \mathsf{W}$ in $\mathsf{T_{PT}}^-$. Since formulas of the form $\mathsf{T}(t)$ are able to store nested conjunctions, they correspond to nested lists in our algebra. In greater detail, on words, simple constantly increasing functions are allowed. In addition, recursion on notation is allowed for words if it produces a list as output, just in analogy to truth induction in $\mathsf{T_{PT}}^-$. Finally, we include functions to copy, delete, insert or read an element of a nested list, where a word vector input describes the relevant position within the given input list. As the above sketched theory of truth, such a function algebra would achieve polynomial strength contrary to Cook and Bellantonis B not by executing a recursion with a sufficiently large recursion argument, but by using short recursions on increasingly complicated lists. An analogous new function algebra should be constructible also for exponential time.

Altogether, we hope that the thesis could convince the reader of the close and fruitful connection between complexity theory and logic. As sketched above, the second-order notions analysed in detail in our thesis could inspire the design of new function algebras on alternative data types such as lists and sets. The analysis of such function algebras is a very active branch of research (see e.g. [2, 3, 50]). The motivation of improving the understanding of computation on alternative data structures is twofold: First, computation models on such structures might have better chances of nicely implementing real-world algorithms, and secondly, the detour of using different data structures might also imply deep complexity-theoretic results in the standard setting of words.

Bibliography

[1] ACZEL, P. Frege structures and the notion of proposition, truth and set. In *The Kleene Symposium* (1980), J. Barwise, H. Keisler, and K. Kunen, Eds., North-Holland, pp. 31– 59.

[2] ARAI. Predicatively computable functions on sets. Research article, 2012. http://arxiv.org/abs/1204.5582.

[3] BECKMANN, A., BUSS, S. R., AND FRIEDMAN, S.-D. Safe recursive set-functions. Submitted for publication, 2012.

[4] BEESON, M. J. *Foundations of Constructive Mathematics: Metamathematical Studies.* Springer, Berlin, 1985.

[5] BEESON, M. J. Proving programs and programming proofs. In *Logic, Methodology and Philosophy of Science VII*, Barcan Marcus et. al., Ed. North Holland, Amsterdam, 1986, pp. 51–82.

[6] BELLANTONI, S. *Predicative Recursion and Computational Complexity.* PhD thesis, University of Toronto, 1992.

[7] BELLANTONI, S., AND COOK, S. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity 2* (1992), 97–110.

[8] BISHOP, E. *Foundations of Constructive Analysis.* McGraw-Hill, 1967.

[9] BISHOP, E., AND BRIDGES, D. *Constructive Analysis.* Springer-Verlag, 1985.

[10] BUSS, S., Ed. *Handbook of Proof Theory.* Elsevier, Amsterdam, 1998.

[11] Buss, S. R. *Bounded Arithmetic.* Bibliopolis, Napoli, 1986.

[12] Buss, S. R. The witness function method and fragments of Peano arithmetic. In *Proceedings of the Ninth International Congress on Logic, Methodology and Philosophy of Science, Uppsala, Sweden, August 7–14, 1991*, D. Prawitz, B. Skyrms, and D. Westerst r ahl, Eds. Elsevier, North Holland, Amsterdam, 1994, pp. 29–68.

[13] Cantini, A. *Logical Frameworks for Truth and Abstraction.* North-Holland, Amsterdam, 1996.

[14] Cantini, A. Proof-theoretic aspects of self-referential truth. In *Tenth International Congress of Logic, Methodology and Philosophy of Science, Florence, August 1995*, Maria Luisa Dalla Chiara et. al., Ed., vol. 1. Kluwer, September 1997, pp. 7–27.

[15] Cantini, A. Characterizing poly-time with an intuitionistic theory based on combinatory logic and safe induction. Preprint, Firenze, 1999. 14 pages.

[16] Cantini, A. Feasible operations and applicative theories based on $\lambda\eta$. *Mathematical Logic Quarterly 46*, 3 (2000), 291–312.

[17] Cantini, A. Polytime, combinatory logic and positive safe induction. *Archive for Mathematical Logic 41*, 2 (2002), 169–189.

[18] Cantini, A. Choice and uniformity in weak applicative theories. In *Logic Colloquium '01*, M. Baaz, S. Friedman, and J. Krajíček, Eds., vol. 20 of *Lecture Notes in Logic.* Association for Symbolic Logic, 2005, pp. 108–138.

[19] Cantini, A. Remarks on applicative theories. *Annals of Pure and Applied Logic 136* (2005), 91–115.

[20] Clote, P. Computation models and function algebras. In *Handbook of Computability Theory*, E. Griffor, Ed. Elsevier, 1999, pp. 589–681.

[21] Clote, P., and Krajíček, J., Eds. *Arithmetic, Proof Theory and Computational Complexity.* Claredon Press, Oxford, 1993.

[22] Clote, P., and Remmel, J., Eds. *Feasible Mathematics II*, vol. 13 of *Progress in Computer Science and Applied Logic*. Birkhäuser, Basel, 1995.

[23] Cobham, A. The intrinsic computational difficulty of functions. In *Logic, Methodology and Philosophy of Science II*. North Holland, Amsterdam, 1965, pp. 24–30.

[24] Cook, S. A., and Nguyen, P. *Logical Foundations of Proof Complexity*. ASL Prespectives in Logic. Cambridge University Press, 2010.

[25] Curry, H. Grundlagen der kombinatorischen Logik. *American journal of Mathematics 52* (1930), 509–536 and 789–834.

[26] Curry, H., and Feys, R. *Combinatory Logic, vol. 1*. North-Holland, 1958.

[27] Curry, H., Hindley, J. R., and Seldin, J. *Combinatory Logic, vol. 2*. North-Holland, 1972.

[28] Eberhard, S. Applicative theories for logarithmic complexity classes. Submitted, Oct. 2012.

[29] Eberhard, S. A feasible theory of truth over combinatory logic. Submitted, Oct. 2012.

[30] Eberhard, S., and Strahm, T. Towards the unfolding of feasible arithmetic (Abstract). *Bulletin of Symbolic Logic 18*, 3 (2012), 474–475.

[31] Eberhard, S., and Strahm, T. Unfolding feasible arithmetic and weak truth. In *Axiomatic Theories of Truth* (2012), T. Achourioti, H. Galinon, K. Fujimoto, and J. Martínez-Fernández, Eds., Logic, Epistemology and the Unity of Science, Springer. Being published.

[32] Eberhard, S., and Strahm, T. Weak theories of truth and explicit mathematics. In *Logic, Construction, Computation*, Ulrich Berger, Hannes Diener, and Peter Schuster, Eds. Ontos Verlag, 2012.

[33] Feferman, S. A language and axioms for explicit mathematics. In *Algebra and Logic*, J. Crossley, Ed., vol. 450 of *Lecture Notes in Mathematics.* Springer, Berlin, 1975, pp. 87–139.

[34] Feferman, S. Constructive theories of functions and classes. In *Logic Colloquium '78*, M. Boffa, D. van Dalen, and K. McAloon, Eds. North Holland, Amsterdam, 1979, pp. 159–224.

[35] Feferman, S. Iterated inductive fixed-point theories: application to Hancock's conjecture. In *The Patras Symposion*, G. Metakides, Ed. North Holland, Amsterdam, 1982, pp. 171–196.

[36] Feferman, S. Logics for termination and correctness of functional programs. In *Logic from Computer Science*, Y. N. Moschovakis, Ed., vol. 21 of *MSRI Publications.* Springer, Berlin, 1991, pp. 95–127.

[37] Feferman, S. Logics for termination and correctness of functional programs II: Logics of strength PRA. In *Proof Theory*, P. Aczel, H. Simmons, and S. S. Wainer, Eds. Cambridge University Press, Cambridge, 1992, pp. 195–225.

[38] Feferman, S. Gödel's program for new axioms: Why, where, how and what? In *Gödel '96*, P. Hájek, Ed., vol. 6 of *Lecture Notes in Logic.* Springer, Berlin, 1996, pp. 3–22.

[39] Feferman, S. Predicativity. In *The Oxford Handbook of the Philosophy of Mathematics and Logic*, S. Shapiro, Ed. Oxford University Press, 2005, pp. 590–624.

[40] Feferman, S. Axioms for the determinateness of truth. *Review of Symbolic Logic 1* (2008), 204–217.

[41] Feferman, S., and Jäger, G. Systems of explicit mathematics with non-constructive μ-operator. Part I. *Annals of Pure and Applied Logic 65*, 3 (1993), 243–263.

[42] Feferman, S., and Jäger, G. Systems of explicit mathematics with non-constructive μ-operator. Part II. *Annals of Pure and Applied Logic 79*, 1 (1996), 37–52.

[43] Feferman, S., and Strahm, T. The unfolding of non-finitist arithmetic. *Annals of Pure and Applied Logic 104*, 1–3 (2000), 75–96.

[44] Feferman, S., and Strahm, T. Unfolding finitist arithmetic. *Review of Symbolic Logic 3*, 4 (2010), 665–689.

[45] Ferreira, F. Polynomial time computable arithmetic. In *Logic and Computation, Proceedings of a Workshop held at Carnegie Mellon University, 1987*, W. Sieg, Ed., vol. 106 of *Contemporary Mathematics*. American Mathematical Society, Providence, Rhode Island, 1990, pp. 137–156.

[46] Friedman, H., and Sheard, M. An axiomatic approach to self-referential truth. *Annals of Pure and Applied Logic 33*, 1 (1987), 1–21.

[47] Girard, J.-Y. *Proof Theory and Logical Complexitiy.* Bibliopolis, Napoli, 1987.

[48] Gödel, K. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica 12*, 3-4 (1958), 280–287.

[49] Halbach, V. *Axiomatic Theories of Truth.* Cambridge University Press, 2011.

[50] Hofmann, M., and Schöpp, U. Pure pointer programs with iteration. *Lecture Notes in Computer Science 5213* (2008), 78–93.

[51] Ishihara, H. Function algebraic characterizations of the polytime functions. *Computational Complexity 8* (1999), 346–356.

[52] Jäger, G. Induction in the elementary theory of types and names. In *Computer Science Logic '87*, E. Börger, H. Kleine Büning, and M.M. Richter, Eds., vol. 329 of *Lecture Notes in Computer Science.* Springer, Berlin, 1988, pp. 118–128.

[53] Jäger, G. Fixed points in Peano arithmetic with ordinals. *Annals of Pure and Applied Logic 60*, 2 (1993), 119–132.

[54] Jäger, G., Kahle, R., and Studer, T. Universes in explicit mathematics. *Annals of Pure and Applied Logic 109*, 3 (2001), 141–162.

[55] Jäger, G., and Probst, D. The Suslin operator in applicative theories: its proof-theoretic analysis via ordinal theories. *Annals of Pure and Applied Logic* (2011), in press.

[56] Jäger, G., and Strahm, T. Totality in applicative theories. *Annals of Pure and Applied Logic 74*, 2 (1995), 105–120.

[57] Jäger, G., and Strahm, T. The proof-theoretic strength of the Suslin operator in applicative theories. In *Reflections on the Foundations of Mathematics: Essays in Honor of Solomon Feferman*, W. Sieg, R. Sommer, and C. Talcott, Eds., vol. 15 of *Lecture Notes in Logic*. Association for Symbolic Logic, 2002, pp. 270–292.

[58] Jäger, G., and Strahm, T. Reflections on reflections in explicit mathematics. *Annals of Pure and Applied Logic 136*, 1–2 (2005), 116–133.

[59] Kahle, R. *Applikative Theorien und Frege-Strukturen.* PhD thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 1997.

[60] Kahle, R. *The Applicative Realm.* Habilitation Thesis, Tübingen, 2007. Appeared in Textos de Mathemática 40, Departamento de Mathemática da Universidade de Coimbra, Portugal, 2007.

[61] Kahle, R., and Oitavem, I. An applicative theory for FPH. In *Proceedings Third International Workshop on Classical Logic and Computation CL&C* (2010), S. van Bakel, S. Berardi, and U. Berger, Eds., vol. 47 of *EPTCS*.

[62] Kahle, R., and Oitavem, I. Applicative theories for the polynomial hierarchy of time and its levels. Accepted for publication in Annals of Pure and Applied Logic, 2012.

[63] Krähenbühl, J. Explicit mathematics with positive existential comprehension and join. Master's thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 2006.

[64] Kripke, S. Outline of a Theory of Truth. *The journal of Philosophy* (1975), 690–716.

[65] Lind, J. Computing in logarithmic space. Tech. rep., Massachusetts Institute of Technology, 1974.

[66] Møller Neergaard, P. A functional language for logarithmic space. In *Asian Symposium on Programming Languages and Systems*. 2004, pp. 311–326.

[67] Moschovakis, Y. *Elementary induction on abstract structures. Studies in Logic and the Foundations of Mathematics*, vol. 77. North-Holland, 2008.

[68] Oitavem, I. Logspace without Bounds. In *Ways of proof theory*, R. Schindler, Ed. Ontos Verlag, 2010, pp. 349–356.

[69] Probst, D. The provably terminating operations of the subsystem PETJ of explicit mathematics. *Annals of Pure and Applied Logic 162*, 11 (2011), 934–947.

[70] Schönfinkel, M. Über die Bausteine der mathematischen Logik. *Mathematische Annalen 92* (1924), 305–316. Enter text here.

[71] Schönfinkel, M. On the building blocks of mathematical logic. In *From Frege to Gödel: A Source Book in Mathematical Logic, 1871-1931*, Jean van Heijenoort, Ed. Harvard University Press, 1967, pp. 355–366.

[72] Spescha, D. *Weak systems of explicit mathematics.* PhD thesis, Universität Bern, 2009.

[73] Spescha, D., and Strahm, T. Elementary explicit types and polynomial time operations. *Mathematical Logic Quarterly 55*, 3 (2009), 245–258.

[74] SPESCHA, D., AND STRAHM, T. Realizability in weak systems of explicit mathematics. *Mathematical Logic Quarterly 57*, 6 (2011), 551–565.

[75] STRAHM, T. Theories with self-application of strength PRA. Master's thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 1992.

[76] STRAHM, T. Polynomial time operations in explicit mathematics. *Journal of Symbolic Logic 62*, 2 (1997), 575–594.

[77] STRAHM, T. First steps into metapredicativity in explicit mathematics. In *Sets and Proofs*, S. B. Cooper and J. Truss, Eds. Cambridge University Press, 1999, pp. 383–402.

[78] STRAHM, T. *Proof-theoretic Contributions to Explicit Mathematics.* Habilitationsschrift, University of Bern, 2001.

[79] STRAHM, T. Theories with self-application and computational complexity. *Information and Computation 185* (2003), 263–297.

[80] STRAHM, T. A proof-theoretic characterization of the basic feasible functionals. *Theoretical Computer Science 329* (2004), 159–176.

[81] STRAHM, T. Weak theories of operations and types. In *Ways of Proof Theory*, R. Schindler, Ed. Ontos Verlag, 2010, pp. 441–468.

[82] TROELSTRA, A., AND VAN DALEN, D. *Constructivism in Mathematics*, vol. I. North-Holland, Amsterdam, 1988.

[83] TROELSTRA, A., AND VAN DALEN, D. *Constructivism in Mathematics*, vol. II. North Holland, Amsterdam, 1988.

Erklärung

gemäss Art. 28 Abs. 1 RSL 05

Name/Vorname:	Eberhard Sebastian
Matrikelnummer:	03-916-954
Studiengang:	Informatik, Dissertation
Titel der Arbeit:	Weak applicative theories, truth, and computational complexity
Leiter der Arbeit:	Prof. Dr. Thomas Strahm

Ich erkläre hiermit, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäss aus Quellen entnommen wurden, habe ich als solche gekennzeichnet. Mir ist bekannt, dass andernfalls der Senat gemäss Artikel 36 Absatz 1 Buchstabe o des Gesetzes vom 5. September 1996 über die Universität zum Entzug des aufgrund dieser Arbeit verliehenen Titels berechtigt ist.

Bern, den 14. Mai 2013	....
Ort/ Datum	Unterschrift

Lebenslauf

von Sebastian Eberhard

1984:	Geboren am 5. September in Flawil
1999-2003:	Kantonsschule am Burggraben, St. Gallen
2003-2004:	Erstes Vordiplom in Mathematik, ETH Zürich
2004-2005:	Zivildienst in Altersheimen in Wintherthur und St. Gallen
2005-2007:	Bachelor in Philosophie mit Nebenfach Mathematik an der Universität Bern
2007-2009:	Master in Mathematik an der Universität Bern
2009-2013:	PhD-Studium in Informatik bei Prof. Dr. Gerhard Jäger in der Logic and Theory Group (ehemals Forschungsgruppe für Theoretische Informatik und Logik) am Institut für Informatik und angewandte Mathematik der Universität Bern. Betreuer: Prof. Dr. Thomas Strahm