Universiteit Utrecht

Master thesis

Arithmetical conservativity results, a theory of operations and Goodman's theorem

Author:
Lotte van Slooten

Supervisor:
Benno van den Berg
UU supervisor:
Jaap van Oosten
UU second reader:
Albert Visser

May 28, 2014

Abstract

We present a new theory of operations $\mathbf{HAP}_\epsilon$ and show that it is a conservative extension of Heyting Arithmetic. An important property of $\mathbf{HAP}_\epsilon$ is that in this system all arithmetical formulas are self-realising. This will allow us to give a new proof of Goodman's theorem. Our proof of Goodman's theorem uses the proof interpretations realizability and forcing and is inspired by the work of Michael Beeson [**?**] and Gerard Renardel de Lavalette [**?**]. In contrast to their proofs, we broke up the proof of Goodman's theorem into four steps, making sure we only use one proof interpretation at the time. This makes each step easier to understand.

Contents

Introduction

This thesis is about conservativity results for some constructive systems. We will explain these notions below.

Conservativity results and Proof interpretations

In logic if one asks the question: Is ϕ valid?, a natural response would be: In which theory or system do you want to know this? Suppose one comes up with the answer ϕ is valid in a theory T. A next question would be: But is it also valid in another theory T_1? This way a study arises of comparing logical systems. If a system T proves the same formulas as system T_1 we say that T is conservative over T_1. A formal definition is the following.

Definition 1. *Let T_1 be a theory in the language $\mathcal{L}_1$, T_2 a theory in the language $\mathcal{L}_2$ and $\mathcal{L} \subseteq \mathcal{L}_1 \cap \mathcal{L}_2$. Then T_1 is $\mathcal{L}$-conservative over T_2 if for all formulas ϕ in the language $\mathcal{L}$ the following holds: $T_1 \vdash \phi \Rightarrow T_2 \vdash \phi$.*

If it is proven that $T_1 \supset T_2$ is $\mathcal{L}$-conservative over T_2, we get extra axioms from T_1 to come up with interesting results about $\mathcal{L}$-formulas in T_2. So we would have extra tools to prove the same things; the lengths of proofs might become shorter and we might come up with results we did not come up with when we had just our system T_2.

A neat way to prove these conservativity results is with a proof interpretation

Definition 2 (Proof interpretation)**.** *A proof interpretation consists of a transformation $(\ )^*$ with the two properties listed below. The transformation sends an $\mathcal{L}_1$-formula A to an $\mathcal{L}_2$-formula A^* and the two properties are the following:*

1. $T_1 \vdash \phi \Rightarrow T_2 \vdash \phi^*$ *for all $\mathcal{L}_1$-formulas ϕ*

2. $T_2 \vdash \phi \leftrightarrow \phi^*$ *for all $\mathcal{L}$-formulas ϕ*

In this thesis these steps will be proven several times.

Constructive systems and Goodman's theorem

In this thesis we will be talking about constructive systems. In order to get a feeling for constructivism, we will first explain what the ideas behind constructivism are.

In constructivism there are many different 'schools'. A basic idea they all agree on is that a statement is true if we have a proof for it and false if we can show that the assumption that there is a proof for the statement leads to a contradiction. Thus for an arbitrary statement we can not say if it is either true or false. In order to get a better feeling for this concept we will get into constructivsm a bit more.

For a constructivist the meaning a formula has is different than for most people, especially when we are talking about $\exists x.A(X)$ or $A \vee B$. Constructively these statements should be read as 'we can construct an x such that $A(x)$' and 'we can decide between A and B'. Constructivists are not satisfied knowing that somewhere there is an x, but they really want to know which one.

In general we can ask which objects exist as contructions. Natural numbers are usually viewed as unproblematic and are used by constructivists. Classically one can define a natural number like this

$$n = \begin{cases} 1 & \text{if } A \text{ holds} \\ 2 & \text{otherwise} \end{cases}$$

In this definition, A might be a statement which has neither been proved nor refuted. Constructively this is unacceptable as the description of a natural number, since we can not identify n until the truth of A has been decided.

One school in constructive mathematics is intuitionism, which is the approach in the spirit of Brouwer and Heyting. Formally, intuitionistic logic is a restriction of classical logic in which the law of excluded middle ($A \vee \neg A$) and double negation eliminination ($\neg\neg A \rightarrow A$) are not admitted as axioms. Especially for the law of excluded middle, it makes sence that it is not an axiom in intuitionism, since in general we can not always decide whether A is true or refutable.

A basic example of a formal system based on intuitionistic logic is Heyting Arithmetic:

The system Heyting Arithmetic (HA). Heyting Arithmetic is an axiomatization of arithmetic in accordance with the philosophy of intuitionism. Our definition of **HA** comes from [**?**]. The language $\mathcal{L}(\mathbf{HA})$ contains the constant 0, a unary function symbol S, function symbols for all primitive recursive functions and the relation symbol $'='$. The logical basis will be intuitionistic predicate logic with equality.

Terms:
i) The constants and variables.
ii) If t, t' are terms then so is St, $t + t'$ and $t \cdot t'$.

Formulas:
i) The atomic formulas are expressions of the form $t_1 = t_2$, where t_1, t_2 are terms.
ii) Formulas are built from other formulas with the logical operators $\rightarrow, \wedge, \vee, \forall x, \exists x$.

Axioms:
The axioms of the logical basis, intuitionistic predicate logic are

$(A \wedge B) \to A,\ (A \wedge B) \to B,$
$A \to (B \to (A \wedge B)),$
$A \to (A \vee B),\ B \to (A \vee B),$
$(A \to C) \to ((B \to C) \to ((A \vee B) \to C)),$
$A \to (B \to A),$
$(A \to (B \to C)) \to ((A \to B) \to (A \to C)),$
$\forall x(B \to A) \to (B \to \forall x.A) \quad (x \notin FV(B)),$
$\forall x.A \to A(t/x),$
$\forall x(A \to B) \to (\exists x.A \to B) \quad (x \notin FV(B)),$
$A(t/x) \to \exists x.A,$
$\bot \to A.$

And we have the modus ponens rule.

As equality axioms we get

$$x = x, \quad x = y \to y = x, \quad x = y \wedge y = z \to x = z.$$

The defining equations for all primitive recursive functions, such as

$$\begin{aligned} &x + 0 = x, \\ &x + Sy = S(x + y), \\ &x \cdot 0 = 0, \\ &x \cdot Sy = x \cdot y + x. \end{aligned}$$

Finally we have axioms defining the successor and induction

$$\begin{aligned} &(Sx = Sy) \to (x = y), \\ &\neg 0 = Sx, \\ &(A(0) \wedge \forall x\,(A(x) \to A(Sx))) \to \forall y\,A(y). \end{aligned}$$

Here $\bot := (0 = S0)$ and $\neg A := A \to \bot$.

Definition 3. *A is a arithmetical sentence if $A \in \mathcal{L}(\boldsymbol{HA})$*

In **HA** we speak of primitive recursive functions, here we give the precise definition of these functions.

Definition 4. *The class of primitive recursive functions is generated by the following clauses*

1. *0 is a 0-ary primitive recursive function*

2. *$Z = \lambda x.0$ is a primitive recursive function*

3. $S = \lambda x.x + 1$ *is a primitive recursive function*

4. $\Pi k_i = \lambda x_1, \ldots, x_k.x_i$ *(for* $1 \leq i \leq k$*) are primitive recursive*

5. *the primitive recursive functions are closed under composition and definition by primitive recursion*

These primitive recursive functions are computable functions. Later on we will need an application in **HA**. To be able to work with an application we will define the relation T by

$T(e, \vec{x}, y)$ holds if and only if e is the code of a program and y is the code for a terminating computation with P and input $\vec{x}$.

There is a primitive recursive function U which extracts the result from the code for a terminating computation ($U(y)$). As an application we will use the notation of the kleene-brackets $\{e\}(x)$, this is the output of the program with code e and input x, if defined.

An extension of **HA** we will use in this thesis is the typed system $\mathbf{HA}^\omega$:

The system $\mathbf{HA}^\omega$. The language $\mathcal{L}(\mathbf{HA}^\omega)$ contains an unary function symbol S, the relation symbols $=_\sigma$ for all types σ, an application operator $Ap^{\sigma,\tau}$ and the constants 0, $\mathbf{p}^{\sigma,\tau}$, $\mathbf{p}_0^{\sigma,\tau}$, $\mathbf{p}_1^{\sigma,\tau}$, $\mathbf{k}^{\sigma,\tau}$, $\mathbf{s}^{\rho,\sigma,\tau}$, $\mathbf{r}^\sigma$. $Ap^{\sigma,\tau}(t, t')$ will just be written as tt'.

Terms:
i) The constants and variables of type σ are terms of type σ.
ii) If t is a term of type 0, then so is St.
iii) If t is a term of type $\sigma \to \tau$ and t' a term of type σ then $Ap^{\sigma\to\tau,\sigma}(t, t')$ is a term of type τ.

Formulas:
i) The atomic formulas are expressions of the form $t_1 =_\sigma t_2$, where t_1, t_2 are terms of type σ.
ii) Formulas are built from other formulas with the logical operators $\to, \wedge, \vee, \forall x^\sigma, \exists x^\sigma$.

We have the same axioms for the logical basis and equality as in **HA**, but also some additional axioms for equality

$y = z \to xy = xz$,
$x = y \to xz = yz$.

The axioms defining the constants are

$\mathbf{p}_0(\mathbf{p}xy) = x,$
$\mathbf{p}_1(\mathbf{p}xy) = y,$
$\mathbf{p}(\mathbf{p}_0 z)(\mathbf{p}_1 z) = z,$
$\mathbf{k}xy = x,$
$\mathbf{s}xyz = xz(yz),$
$\mathbf{r}xy0 = y,$
$\mathbf{r}xy(Sz) = x@z@(\mathbf{r}xyz).$

Finally we have the arithmetical axioms

$(Sx = Sy) \to (x = y),$
$\neg 0 = Sx,$
$(A(0) \wedge \forall x\,(A(x) \to A(Sx))) \to \forall y\, A(y).$

There are two variants of $\mathbf{HA}^\omega$, the extension with the intensionality axiom **I** or with the extensionality axiom **E**.

$\mathbf{I}$: $\mathbf{e}^\sigma xy = 0 \vee \mathbf{e}^\sigma xy = 1,$ here $\mathbf{e}$ is an equality functional
$\mathbf{e}^\sigma xy = 0 \leftrightarrow x =_\sigma y$
$\mathbf{E}$: $\forall y^{\sigma\to\tau}, z^{\sigma\to\tau}(\forall x^\sigma(yx =_\tau zx) \to y =_{\sigma\to\tau} z)$

We write $\mathbf{I}$-$\mathbf{HA}^\omega$ for $\mathbf{HA}^\omega$ together with the axiom **I** and $\mathbf{E}$-$\mathbf{HA}^\omega$ for $\mathbf{HA}^\omega$ together with the axiom **E**. The last common axiom we will use in this thesis is the axiom of choice (**AC**).

$$\mathbf{AC} : \forall x^\sigma \exists y^\tau A(x,y) \to \exists f^{\sigma\to\tau} \forall x^\sigma A(x, f(x))$$

Now we have the needed information to get into Goodman's theorem. In 1976 Nicholas Goodman published a paper [**?**] in which he proved a theorem which would later be known as Goodman's theorem.

Theorem 1 (Goodman's theorem)**.** ***HA***$^\omega + AC$ *is* ***HA****-conservative over* ***HA****.*

The theorem itself sparked interest with the people who work with constructive mathematics and some tried to make better or clearer ways to prove this theorem. The proof interpretations forcing and realizability became the key-methods to prove Goodman's theorem. Studying these proofs and proof interpretations we introduced our own system $\mathbf{HAP}_\epsilon$ which is conservative over **HA** and is self-realising for all arithmetical formulas. In this thesis we will write "conservative" instead of "**HA**-conservative" for convenience. When we look at the definition of a proof interpretation we see that in this thesis $\mathcal{L}$ will be $\mathcal{L}(\mathbf{HA})$, since we will prove the conservativity results for all arithmetical formulas.

Literature

Over the years Nicholas Goodman and several other people worked on Goodman's theorem. Here will be given a brief overview of there work, for a more detailed discussion and comparison to our work see chapter **??**.

It all started of course with Nicholas Goodman himself, he introduced his theorem and wrote two papers [**?**, **?**] in which he proved his theorem in two different ways. His first proof [**?**] was based on the interpretation of $\mathbf{HA}^\omega$ in his arithmetical theory of constructions. In his second paper [**?**] he used the more modern techniques realizability and forcing, which were used by other people as well to prove his theorem, we will mention their papers below. In Goodman's paper, forcing and realizability were both part of one proof interpretation, he called it relativised realizability.

After that, Micheal Beeson came with an article [**?**] in 1979 on the subject and a book [**?**] in 1985 in which again he proved Goodman's theorem. From Beeson's work we got most inspiration for our own work in this thesis. Beeson gave mostly an outline on how he wanted to prove this theorem and let the details up to the reader, focussing only on one or two parts. He did use realizability and forcing as two techniques instead of one, but he still used forcing to prove a result about realizers. Our goal in this thesis will be to really separate them.

In 1990 Gerard Renardel de Lavalette published a paper [**?**] on the subject. The first part of his proof has the same ideas as Beeson's proof. In the second part he moved away from the direct approach which was used so far, but used an interpretation in a modal theory with modal logic to finish his proof instead.

In 2012 Thierry Coquand published a paper [**?**] with as a goal to clarify the proofs which were presented so far. He did this by proving the theorem for a specific arithmetical formula. He made the proof almost as an algorithm, which made some of the steps more clear.

The last paper we want to mention is a paper of Ulrich Kohlenbach [**?**], in which he did not prove a conservativity result. On the contrary, he proved that $(\mathbf{E}\text{-})\mathbf{HA}^\omega_- + \mathbf{AC}_{\mathrm{ar}}$ is not conservative over $\mathbf{HA}_-$, where $(\mathbf{E}\text{-})\mathbf{HA}^\omega_-$ is $(\mathbf{E}\text{-})\mathbf{HA}^\omega$ with induction restricted to quantifier-free induction and $\mathbf{HA}_-$ is **HA** with induction restricted to quantifier-free induction. Quantifier-free induction is induction over a quantifier free formula A. $\mathbf{AC}_{ar}$ is the axiom of choice with all the quantifiers restricted to type 0. He proved this by building a specific arithmetical formula A such that $(\mathbf{E}\text{-})\mathbf{HA}^\omega_- + \mathbf{AC}_{\mathrm{ar}} \vdash A$ and $\mathbf{HA}_- \nvdash A$.

This thesis

In this thesis we will prove that $(\mathbf{E}\text{-})\mathbf{HA}^\omega + \mathbf{AC}$ and $(\mathbf{I}\text{-})\mathbf{HA}^\omega + \mathbf{AC}$ are conservative over **HA**. We will prove this by introducing the new systems **HAP** and $\mathbf{HAP}_\epsilon$. $\mathbf{HAP}_\epsilon$ has the interesting properties that every arithmetical formula is self-realising and that $\mathbf{HAP}_\epsilon$ is conservative over **HA**. We will prove these properties in chapter 4 and 5.

With these extra systems we are able to break the proof of Goodman's theorem up in four parts. In every step we use a different proof interpretation. We will use the Kleene brackets, realizability, forcing, and we will change the application. For example, with forcing we will construct approximations to an oracle function which can answer our constructive questions. Since every step is a separate proof we will never use two proof interpretations at the same time. This makes our proof of Goodman's theorem a lot easier to understand.

Chapter 1

Formal systems

In this section we will introduce two formal logical systems. A logical system starts with a language $\mathcal{L}$ consisting of constants, function symbols, and relation symbols. From that language we can make terms, and formulas. These are the things we will work with. Then in the end we get a number of axioms from which we can determine whether a formula is valid in this system or not.

1.1 The system HAP

We will start with a system **HAP**, which is quite similar to **APP**.

APP is a type-free system with a partial application and a predicate N. A partial application is a partial binary function, which we call an application. This predicate N says whether x is a natural number or not. **APP** has as constants zero, the successor, the predecessor, pairing and unpairing operators, constants for the combinators and the numerical definition by cases. The system is based on logic with partial terms with equality. (See [**?**] for a better description.)

We can see that **APP** has a lot of similarities with $\mathbf{HA}^\omega$. The main difference is that $\mathbf{HA}^\omega$ is typed and that the operation of **APP** is partial, where in $\mathbf{HA}^\omega$ $Ap^{\sigma,\tau}(t,t')$ is always defined if t is a term of type $\sigma \to \tau$ and t' a term of type σ. The smaller differences are that $\mathbf{HA}^\omega$ does not have a predecessor or a predicate N, has a recursor instead of a constant for the numerical definition by cases and is based on intuitionistic predicate logic with equality.

Just like **APP**, **HAP** is a type-free system with a partial application. The first difference is that in **HAP** we do not have this predicate N, but everything is a natural number. This is why we need a 'new' successor constant *Succ*. The second difference is that **HAP** is based on intuitionistic predicate logic with equality, just like **HA** and $\mathbf{HA}^\omega$. We have chosen for this logical basis for **HAP**, since the proofs in this thesis will work better with the axioms of **HAP** similar to the axioms of **HA**. The language $\mathcal{L}(\mathbf{HAP})$ contains a unary function symbol S (the successor), a ternary relation symbol *hap* and the binary relation symbol $' ='$. Furthermore it contains the following constants, 0, *Succ*, $\mathbf{p}$, $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{k}$, $\mathbf{s}$, $\mathbf{d}$.

In appendix **??** we explain how to use a binary partial function symbol @ instead of the ternary relation symbol *hap*. In order to do that we have to

figure out how we work with things like $t@t'$, which formally are not terms. We call these things we want to work with semiterms. In the appendix we give a formal definition for these semiterms and how we can work with them, but basically they are the terms of $\mathcal{L}(\mathbf{HAP}) \cup @$, and we work with @ as if it were a binary function symbol. So we have $x@y = z \to hap(x, y, z)$. Note that the operation @ does not need to be associative. To reduce the amount of brackets, we use the convention of association to the left: we write $a@b@c$, instead of $(a@b)@c$.

The symbol $\downarrow$ is used below. This symbol can be read as 'is defined'. The definition is $t@t'\downarrow := \exists z(t@t' = z)$, with t, t' terms.

Terms:
i) The constants and variables.
ii) If t is a term then so are St.

Formulas:
i) The atomic formulas are expressions of the form $t_1 = t_2$ or $hap(t_1, t_2, t_3)$, where t_1, t_2, t_3 are terms.
ii) Formulas are built from other formulas with the logical operators $\to, \wedge, \vee, \forall x, \exists x$.

Axioms:
The axioms of the logical basis are the same as for **HA**, but with some additional axioms for equality

$$y = z \to x@y = x@z,$$
$$x = y \to x@z = y@z.$$

The axioms defining the constants are

$Succ@x = Sx$

$\mathbf{p}_0@x\downarrow$, $\mathbf{p}_0@(\mathbf{p}@x@y) = x$,

$\mathbf{p}_1@x\downarrow$, $\mathbf{p}_1@(\mathbf{p}@x@y) = y$,

$\mathbf{p}@x@y\downarrow$, $\mathbf{p}@(\mathbf{p}_0@z)@(\mathbf{p}_1@z) = z$,

$\mathbf{k}@x\downarrow$, $\mathbf{k}@x@y = x$,

$\mathbf{s}@x@y\downarrow$, $\mathbf{s}@x@y@z = x@z@(y@z)$,

$t \neq t' \to \mathbf{d}@t_1@t_2@t@t' = t_1 \wedge \mathbf{d}@t_1@t_2@t@t = t_2$,

$hap(x, y, z) \wedge hap(x, y, z') \to z = z'$.

Finally we have the arithmetical axioms

$$(Sx = Sy) \to (x = y),$$
$$\text{IND: } A(0) \wedge \forall x\,(A(x) \to A(Sx) \to \forall y\, A(y).$$

The combinators $\mathbf{k}, \mathbf{s}$ permit us to have λ-abstraction defined by induction on the construction of semiterms. We do this similar to how it was done in [**?**] for $\mathbf{HA}^\omega$.

$$\begin{aligned}
\lambda x.t &:= \mathbf{k}@t \text{ if } x \notin FV(t),\\
\lambda x.x &:= \mathbf{s}@\mathbf{k}@\mathbf{k},\\
\lambda x.t@x &:= t \text{ if } x \notin FV(t),\\
\lambda x.t@t' &:= \mathbf{s}@(\lambda x.t)@(\lambda x.t') \text{ if } x \in FV(t) \text{ or } x \in FV(t') \text{ and } t' \neq x,\\
\lambda x.St &:= \lambda x.Succ@t.
\end{aligned}$$

For this definition, $t[x]$ is any term of **HAP** and x is free in t, then the following holds.

$$(\lambda x.t(x))(t') = t(t'), \qquad \lambda x.(t@x) = t \text{ if } x \notin FV(t),$$
$$x \notin FV(t') \cup FV(t'') \Rightarrow t' = t'' \rightarrow \lambda x.t(y/t') = \lambda x.t(y/t'').$$

1.1.1 $\mathbf{HAP}_\epsilon$

The system $\mathbf{HAP}_\epsilon$ is the system **HAP** extended with a constant ϵ. The constant ϵ will act as an oracle to give answers to the following constructive questions; If $\exists x.A(x)$ and $B \vee D$ are valid, then for which x is $A(x)$ valid and is B or D valid. To make ϵ work as an oracle the following two axioms are added to the system $\mathbf{HAP}_\epsilon$ for arithmetical formulas B, D:

$$\exists x.B(x,y) \rightarrow (\epsilon@^\ulcorner \exists x.B(x,z)^\urcorner@y\downarrow \wedge B(\epsilon@^\ulcorner \exists x.B(x,z)^\urcorner@y, y),$$

$$\begin{aligned}
B(y) \vee D(y)) \rightarrow \Big(&\epsilon@^\ulcorner B(z) \vee D(z)^\urcorner@y\downarrow\\
&\wedge ((\epsilon@^\ulcorner B(z) \vee D(z)^\urcorner@y = 0 \wedge B(y)) \ \vee \ (\epsilon@^\ulcorner B(z) \vee D(z)^\urcorner = 1 \wedge D(y)))\Big).
\end{aligned}$$

Here $^\ulcorner A^\urcorner$ codes A, by asigning to A its Gödel number. A Gödel numbering is a function which assigns to each formula a unique natural number, its Gödel number. So basically it is an encoding which allows ϵ to interact with it.

1.2 Properties of HAP

To get some interesting results we need to get to know our system **HAP** a bit better. That is what this chapter is for, we will prove that there are fixed point operators and a primitive recursor operator, to come to the conclusion that there is recursion in our system **HAP**. In [**?**], and [**?**] similar results have been shown for **APP** and PCA's.

Proposition 1.2.1. *There are semiterms $g, h \in \mathcal{L}(\boldsymbol{HAP})$, called fixed point operators, such that for all semiterms x:*

1. $(g@x\downarrow \leftrightarrow x@(g@x)\downarrow) \wedge (g@x\downarrow \rightarrow g@x = x@(g@x)$

2. $h@x\downarrow$ *and for all* y, $h@x@y = x@(h@x)@y$

Proof. Let $a = \lambda z.x@(z@z)$, $b = \lambda cy.x@(c@c)@y$, $g = \lambda x.a@a$, $h = \lambda x.b@b$. Then:
$g@x = a@a$, $x@(g@x) = x@(a@a)$. So $g@x\downarrow \wedge x@(g@x)\downarrow$
$g@x = a@a = x@(a@a) = x@(g@x)$
$h@x = b@b$. So $h@x\downarrow$
$h@x@y = x@(b@b)@y = x@(h@x)@y$ □

Proposition 1.2.2. *In* ***HAP*** *there is a semiterm rec such that for all* x, y, z

$rec@x@y@0 = y$,
$rec@x@y@(Sz) = x@z@(rec@x@y@z)$.

Proof. Use the fixed point operator h, then we can take $h@\rho$ for $rec@x@y$ such that

$$\rho@f@0 = y, \quad \rho@f@(Sz) = x@z@(f@z).$$

Then we have

$$h@\rho@0 = \rho@(h@\rho)@0 = y, \quad h@\rho@z = \rho@(h@\rho)@(Sz) = x@z@(h@\rho@z)0.$$

So put

$$\rho := \lambda f, z.\mathbf{d}@(\mathbf{k}@y)(\lambda u.x@(Pu)@(f@(Pu)))@z@0@z,$$

$$rec := \lambda x, y.h@\rho.$$

If we fill everything in we get:

$$\begin{aligned} rec@x@y@0 = h@\rho@0 &= \rho@(h@\rho)@0 \\ &= \mathbf{d}@(\mathbf{k}@y)(\lambda u.x@u@(h@\rho@u))@0@0@0 = \mathbf{k}@y@0 = y \end{aligned}$$

And for Sz

$$\begin{aligned} rec@x@y@Sz = h@\rho@0 &= \rho@(h@\rho)@0 \\ &= \mathbf{d}@(\mathbf{k}@y)(\lambda u.x@u@(h@\rho@u))@Sz@0@Sz \\ &= (\lambda u.x@u@(h@\rho@u))@Sz \\ &= x@z@(h@\rho@z) = x@z@(rec@x@y@z) \end{aligned}$$

□

Note that the P mentioned in this proof is the predecessor which can be recursively defined from S.

$$Px = \begin{cases} 0 & \text{if } x = 0 \\ z & \text{if } x = Sz \end{cases}$$

Proposition 1.2.3 (Basic functions in **HAP**)**.** *There are semiterms in* $\mathcal{L}(\boldsymbol{HAP})$ *for the basic partial recursive functions:*

- *0, a 0-ary function,*

- *the zero function* $Z(x) = 0$,
- *the successor function* $S(x) = x + 1$,
- *the projection functions* $\Pi_i^n(x_1, \ldots, x_n) = x_i$ *(for* $1 \leq i \leq n$*).*

Proof. The semiterms are

- 0 is the constant 0,
- $Z = \lambda x.0$,
- $S = \lambda x.x + 1$,
- $\Pi_i^n = \lambda x_1, \ldots, x_n.x_i$.

□

With these three propositions we have the needed tools to come to the interesting result that there is recursion in our system **HAP**. This means that for every partial recursive function there is an combinator in **HAP** which represents this function.

Theorem 1.2.4 (Recursion in **HAP**)**.** *In* $\mathcal{L}(\boldsymbol{HAP})$*, we can make for every partial recursive function* f *a semiterm* a_f *such that* $\forall x_1, \ldots, x_n \in \mathbb{N}$ $a_f @ x_1 @, \ldots, @x_n \downarrow$ *if and only if* $f(x_1 \ldots, x_n) \downarrow$ *and are equal if this is the case.*

Proof. A partial recursive function f is constructed from the basic functions using primitive recursion, minimalization and composition. Such an a_f can be constructed using the previous propositions.

The recursor *rec* can be used to define a_f for an f which is defined by primitive recursion.

If f and g are partial recursive and there are $a_f, a_g \in \mathbf{HAP}$ then $a_{f(g)} = \lambda x.a_f @ (a_g @ x)$ works for composition.

The last thing to check is minimalization. The formula $g = \mu y.(f(y) = 0)$ is said to be defined from f by minimalization. So if we have a_f, we must be able to make a_g. Let h be the fixed point operator. Then we need to define $\tilde{a}_g$ such that $\tilde{a}_g = \lambda y.\mathbf{d} @ (a_g @ Sy) @ y @ (a_f @ y) @ 0$ and $a_g = \tilde{a}_g @ 0$. Then the following would happen

$$\tilde{a}_g @ 0 = \begin{cases} 0 & \text{if } f(0) = 0 \\ \tilde{a}_g @ 1 & \text{otherwise} \end{cases}$$

$$\tilde{a}_g @ 1 = \begin{cases} 1 & \text{if } f(1) = 0 \\ \tilde{a}_g @ 2 & \text{otherwise} \end{cases}$$

$$\vdots$$

So if we are able to define $\tilde{a}_g$ in this way, this a_g works for minimalization. Define $\tilde{a}_g := h@\sigma$ and $\sigma := \lambda b, c.\mathbf{d}@(b@Sc)@c@(a_f@c)@0$. Then we get

$$\begin{aligned}\tilde{a}_g@y &= h@\sigma@y \\ &= \sigma@(h@\sigma)@y \\ &= \mathbf{d}@((h@\sigma)@Sy)@y@(a_f@y)@0 \\ &= \mathbf{d}@(\tilde{a}_g@Sy)@y@(a_f@y)@0\end{aligned}$$

□

This means that every partial recursive function is represented by an element of $\mathcal{L}(\mathbf{HAP})$.

Chapter 2

$\mathbf{HAP}_\epsilon$ is conservative over HAP

In this chapter we will prove that the system $\mathbf{HAP}_\epsilon$ is conservative over **HAP**. In order to do that we will introduce another system $\mathbf{HAP}_E$ and a proof interpretation called forcing. Then we will prove that $\mathbf{HAP}_\epsilon$ is conservative over $\mathbf{HAP}_E$ and that $\mathbf{HAP}_E$ is conservative over **HAP**.

2.1 $\mathbf{HAP}_E$

The system $\mathbf{HAP}_E$ is the system **HAP** extended with a ternary relation symbol E. We add the axiom $E(x,y,z) \wedge E(x,y,z') \rightarrow z = z'$. With this axiom, we can use E as a binary partial function symbol, like we did for the relation symbol *hap* and the function symbol @. The system $\mathbf{HAP}_E$ is also extended with the following two axioms for all arithmetical formulas B, D:

$$\exists x.B(x,y) \rightarrow (E(\ulcorner \exists x.B(x,z) \urcorner, y)\downarrow \wedge B(E(\ulcorner \exists x.B(x,z) \urcorner, y), y)$$

$$\begin{aligned}(B(y) \vee D(y)) \rightarrow (&E(\ulcorner B(z) \vee D(z) \urcorner, y)\downarrow \\ &\wedge (E(\ulcorner B(z) \vee D(z) \urcorner, y) = 0 \wedge B(y) \ \vee \ E(\ulcorner B(z) \vee D(z) \urcorner) = 1 \wedge D(y))\end{aligned}$$

2.2 Forcing

Forcing was introduced by Cohen in the sixties [**?**] for proving consistency and independence results in set theory. Later it was modified to a method which is really useful in recursion theory and logic. With this method we try to build conditions with a relation to a sentence in the language we are working in. Intuitively our conditions are approximations to some object, later called E. So if p and q are conditions and p is stronger then q, then p agrees with everything q is saying about our object but also has some new information.

We have $\mathbf{HAP}_E$ for a partial function symbol E. Now we want to associate to each formula A of $\mathbf{HAP}_E$ a formula $p\,\mathbf{f}\,A$ of **HAP**. The forcing conditions will be in a set C with an ordering $(C, \subset)$. The forcing conditions will be finite partial functions p, q. Here we use $q \subset p$ as q extends p. This means that q is

defined wherever p is, and agrees with q on the domain of p, but q may have a bigger domain. We use p, q, s, t as variables ranging over C.

First we have to formalise these notations because at the moment p, q, and the notation $\subset$ are not part of $\mathcal{L}(\mathbf{HAP})$. These forcing conditions can be coded as sequences such that we can work with them as finite partial functions. $q : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$.

By theorem **??**, all partial recursive functions can be represented by appropriate conbinators in **HAP**. So we in particular there are combinators in **HAP** who represent the needed operations to code sequences.

We will need a primitive recursive bijection $j : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ with primitive recursive projection functions j_1, j_2, such that:

$$j_1 j(x, y) = x, \quad j_2 j(x, y) = y, \quad j(j_1 z, j_2 z) = z.$$

From these j, j_1, j_2 we can construct a coding of sequences.

$$\bigcup_{n \geq 0} \mathbb{N}^n \longrightarrow \mathbb{N}$$
$$(x_1, \ldots, x_n) \longmapsto < x_1, \ldots, x_n >$$

With this coding of sequences the following functions are primitive recursive as well.

$lh(\sigma)$ the length function, such that $\begin{cases} lh <>= 0 \\ lh < x_1, \ldots, x_n >= n + 1, \end{cases}$

$\sigma * \tau$ the concatenation function, such that

$$< x_0, \ldots, x_i > * < x_{i+1}, \ldots, x_{i+j} >=< x_0, \ldots, x_{i+j} >,$$

$(\sigma)_i$ the decoding function, such that

$$(< x_0, \ldots, x_n >)_i = \begin{cases} x_i & \text{if } i < n + 1 \\ \text{undefined} & \text{otherwise .} \end{cases}$$

Now a finite partial function $\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ can be taken to be a coded sequance σ.

σ codes the function $n, m \longmapsto (\sigma)_{j(m,n)}$. So if we are working with a forcing condition q, $q(m, n)$ means $(q)_{j(m,n)}$
Now the last thing to define is $q \subset p$

$$q \subset p := p(m, n)\downarrow \rightarrow (q(m, n)\downarrow \wedge q(m, n) = p(m, n)).$$

With these operations and definitions in **HAP** we can work with our forcing conditions as finite partial functions. With that stated, we are ready to define the proof interpretation forcing.

Definition 2.2.1.

$$
\begin{array}{ll}
p \boldsymbol{f} A & \textit{is } A \quad \textit{if } A \textit{ is atomic and } E \textit{ is not a part of } A\ , \\
p \boldsymbol{f} E(x,y,z) & \textit{is } (x,y) \in dom(p)\ \wedge\ p(x,y) = z, \\
p \boldsymbol{f} A \to B & \textit{is } \forall q \subset p (q \boldsymbol{f} A \to \exists s \subset q\, s \boldsymbol{f} B), \\
p \boldsymbol{f} A \wedge B & \textit{is } p \boldsymbol{f} A \wedge p \boldsymbol{f} B, \\
p \boldsymbol{f} A \vee B & \textit{is } p \boldsymbol{f} A \vee p \boldsymbol{f} B, \\
p \boldsymbol{f} \forall x\, A(x) & \textit{is } \forall x \forall q \subset p \exists s \subset q\, s \boldsymbol{f} A(x), \\
p \boldsymbol{f} \exists x\, A(x) & \textit{is } \exists x\, p \boldsymbol{f} A(x).
\end{array}
$$

$p\ \mathbf{f}\ A$ can be read as p forces A. The definition is arranged such that the monotonicity property holds

$$p \subset q \wedge q\ \mathbf{f}\ A \to p\ \mathbf{f}\ A.$$

Remark that the formula saying that E is total, $\forall x, y E(x,y)\downarrow$, will be forced if and only if $\forall x, y \forall q \subset p \exists s \subset q . s(x,y)\downarrow$. We can not guarantee this for all the forcing conditions q, we will need to prove that $\mathbf{HAP}_E$ is conservative over **HAP**. So E must be partial.

We will now prove a basic result of **HAP** about forcing which we will later use in some other results.

Lemma 2.2.2. *If A is arithmetic, then $HAP \vdash (\forall p \exists q \subset p.q \boldsymbol{f} A) \leftrightarrow A$.*

Proof. We shall prove the lemma by induction on the complexity of A. Note that A is arithmetic, so E does not occur in A.
A is atomic;

$$
\begin{array}{ll}
\mathbf{HAP} \vdash \forall p \exists q \subset p.q\ \mathbf{f}\ A & \\
\quad \leftrightarrow \forall p \exists q \subset p . A & \\
\quad \leftrightarrow A &
\end{array}
$$

A is $B \to C$;

$$
\begin{array}{ll}
\mathbf{HAP} \vdash \forall p \exists q \subset p.q\ \mathbf{f}\ A & \\
\quad \to \forall p \exists q \subset p \forall s \subset q (s\ \mathbf{f}\ B \to \exists t \subset s.t\ \mathbf{f}\ C) & \\
\quad \to \forall p \exists s \subset p (s\ \mathbf{f}\ B \to \exists t \subset s.t\ \mathbf{f}\ C) & \\
\quad \to B \to C \quad \text{by the induction hypothesis} & \\
\mathbf{HAP} \vdash B \to C & \\
\quad \to (\forall p \exists q \subset p.q\ \mathbf{f}\ B \to \forall s \exists t \subset s.t\ \mathbf{f}\ C) & \\
\quad \to \forall p \exists q \subset p \forall v \subset q (v\ \mathbf{f}\ B \to \exists t \subset v.t\ \mathbf{f}\ C) \quad \text{by monotonicity} & \\
\quad \to \forall p \exists q \subset p.q\ \mathbf{f}\ B \to C \quad \text{by the induction hypothesis} &
\end{array}
$$

A is $B \wedge C$;

$$
\begin{array}{ll}
\mathbf{HAP} \vdash \forall p \exists q \subset p.q\ \mathbf{f}\ A & \\
\quad \leftrightarrow \forall p \exists q \subset p.q\ \mathbf{f}\ B \wedge \forall p \exists q \subset p.q\ \mathbf{f}\ C & \\
\quad \leftrightarrow B \wedge C \quad \text{by the induction hypothesis} &
\end{array}
$$

A is $B \vee C$;

$\mathbf{HAP} \vdash \forall p \exists q \subset p.q \; \mathbf{f} \; A$
$\leftrightarrow \forall p \exists q \subset p.q \; \mathbf{f} \; B \vee \forall p \exists q \subset p.q \; \mathbf{f} \; C$
$\leftrightarrow B \vee C$ by the induction hypothesis

A is $\forall x \, B(x)$;

$\mathbf{HAP} \vdash \forall p \exists q \subset p.q \; \mathbf{f} \; A$
$\rightarrow \forall p \exists q \subset p \forall x \forall s \subset q \exists t \subset s.t \; \mathbf{f} \; B(x)$
$\rightarrow \forall x \forall s \exists t \subset s.t \; \mathbf{f} \; B(x)$
$\rightarrow \forall x.B(x)$ by the induction hypothesis
$\mathbf{HAP} \vdash \forall x.B(x)$
$\rightarrow \forall x \forall p \exists q \subset p.q \; \mathbf{f} \; B(x)$
$\rightarrow \forall s \exists t \subset s \forall x \forall p \subset t \exists q \subset p.q \; \mathbf{f} \; B(x)$ we restricted p to $p \subset t$
$\rightarrow \forall s \exists t \subset s.t \; \mathbf{f} \; \forall x.B(x)$

A is $\exists x \, B(x)$;

$\mathbf{HAP} \vdash \forall p \exists q \subset p.q \; \mathbf{f} \; A$
$\leftrightarrow \forall p \exists q \subset p \exists x.q \; \mathbf{f} \; B(x)$
$\leftrightarrow \exists x \, B(x)$ by the induction hypothesis

□

2.3 $\mathbf{HAP}_E$ is conservative over HAP

In this chapter we will prove that $\mathbf{HAP}_E$ is conservative over $\mathbf{HAP}$. In order to do that we will need te use forcing, which we introduced above. We will use forcing to get partial functions which are approximations of E. By making better approximations we can force the 'oracle' axioms which were added to $\mathbf{HAP}$ and then we will be able to prove that $\mathbf{HAP}_E$ is conservative over $\mathbf{HAP}$.

Theorem 2.3.1. *$\mathbf{HAP}_E$ is conservative over $\mathbf{HAP}$*

Proof. If $\mathbf{HAP}_E \vdash A$, then in the proof of $\mathbf{HAP}_E \vdash A$ we need only a finite amount of instances of the extra axioms of $\mathbf{HAP}_E$. Namely for a finite amount of arithmetical formulas. Let us call γ a finite collection of instances of the extra axioms of $\mathbf{HAP}_E$. Then it suffices to prove that $\mathbf{HAP}_E^- = \mathbf{HAP} + \gamma + E(x,y,z) \wedge E(x,y,z') \rightarrow z = z'$ is conservative over $\mathbf{HAP}$. We need to work with $\mathbf{HAP}_E^-$ instead of $\mathbf{HAP}_E$, because of how we want to define the forcing conditions. In the definition of the forcing conditions we will quantify over the arithmetical formulas for which there are instances of the extra axioms in γ. If we were still working in $\mathbf{HAP}_E$ we would have to quantify over all the arithmetical formula in order to define the forcing conditions. THhis is not possible in $\mathbf{HAP}$, so that would not work. Let us now turn to the familiar steps of the proof.

1. $\mathbf{HAP}_E^- \vdash A \Rightarrow \mathbf{HAP} \vdash \forall p \exists q \subset p.q \; \mathbf{f} \; A$ for all $\mathbf{HAP}_E$-formulas A.

2. $\mathbf{HAP} \vdash \forall p \exists q \subset p.q \mathbf{f} A \leftrightarrow A$ for all arithmetical formulas A.

Note that we proved part two in lemma **??**. To prove step one, we have to check this for all the axioms of $\mathbf{HAP}_E^-$. Let us check the interesting axioms first, the axioms which were added to $\mathbf{HAP}$.

In order to prove that the axioms of γ are forced, we first have to define the forcing conditions C. Let C consist of all partial functions q with finite domain, with the following conditions. We do not have these conditions for all arithmetical formulas, just for the arithmetical formulas for which there are instances of the extra axioms in γ.

1) $(\exists x.B(x,y) \wedge q(\ulcorner \exists x.B(x,z) \urcorner, y)\downarrow) \rightarrow B(q(\ulcorner \exists x.B(x,z) \urcorner, y), y)$

2) $(B(y) \vee D(y) \ \wedge \ q(\ulcorner B(z) \vee D(z) \urcorner, y)\downarrow) \rightarrow (q(\ulcorner B(z) \vee D(z) \urcorner, y) = 0 \wedge B(y) \ \vee \ q(\ulcorner B(z) \vee D(z) \urcorner, y) = 1 \wedge D(y))$

Now it becomes clear what was meant by approximations of E. The forcing conditions work just like E, as an oracle, but only when they are defined. E on the other end is defined if it needs to work as an oracle.

First look at the axiom $(\exists x.B(x,y) \wedge q(\ulcorner \exists x.B(x,z) \urcorner, y)\downarrow) \rightarrow B(q(\ulcorner \exists x.B(x,z) \urcorner, y), y)$. Suppose $\forall p \exists q \subset p.q \mathbf{f} \exists x.B(x,y)$, then we need to find an s such that

$$s \subset q.s \mathbf{f} (E(\ulcorner \exists x.B(x,z) \urcorner, y)\downarrow) \wedge B(E(\ulcorner \exists x.B(x,z) \urcorner, y), y)$$

This means that we need to find an s such that it is an approximation of E which is defined at $\exists x.B(x,y)$, written down explicitly this gives:

$$s \mathbf{f} E(\ulcorner \exists x.B(x,z) \urcorner, y)\downarrow \quad \text{is } s(\ulcorner \exists x.B(x,z) \urcorner, y)\downarrow$$

$$s \mathbf{f} B(E(\ulcorner \exists x.B(x,z) \urcorner, y), y) \quad \text{is } B(s(\ulcorner \exists x.B(x,z) \urcorner, y), y)$$

If $q(\ulcorner \exists x.B(x,z) \urcorner, y)\downarrow$ then choose $s = q$. If $q(\ulcorner \exists x.B(x,z) \urcorner, y)$ is undefined then we can extend q to a forcing condition s by defining $s(< \exists x.B(x,y) >, y) = x$. Then by 1) we automatically get $B(s(\ulcorner \exists x.B(x,z) \urcorner, y), y)$. So s does what it is supposed to do, and $\forall p \exists q \subset p$ such that q forces the axiom.

Now look at the second axiom $(B(y) \vee D(y) \ \wedge \ q(\ulcorner B(z) \vee D(z) \urcorner, y)\downarrow) \rightarrow (q(\ulcorner B(z) \vee D(z) \urcorner, y) = 0 \wedge B(y) \ \vee \ q(\ulcorner B(z) \vee D(z) \urcorner, y) = 1 \wedge D(y))$.
Suppose $\forall p \exists q \subset pq \mathbf{f} B(y) \vee D(y)$, then we need to find an s such that

$$s \subset q.s \mathbf{f} E(\ulcorner B(z) \vee D(z) \urcorner, y)\downarrow$$
$$\wedge ((E(\ulcorner B(z) \vee D(z) \urcorner, y) = 0 \wedge B(y)) \vee (E(\ulcorner B(z) \vee D(z) \urcorner, y) = 1 \wedge D(y)))$$

Which means

$$s \mathbf{f} E(\ulcorner B(z) \vee D(z) \urcorner, y)\downarrow \quad \text{is } s(\ulcorner B(z) \vee D(z) \urcorner, y)\downarrow$$

$$s \mathbf{f} (E(\ulcorner B(z) \vee D(z) \urcorner, y) = 0 \wedge B(y)) \vee (E(\ulcorner B(z) \vee D(z) \urcorner, y) = 1 \wedge D(y))$$
$$\text{is } (s(\ulcorner B(z) \vee D(z) \urcorner, y) = 0 \wedge B(y)) \vee (q(\ulcorner B(z) \vee D(z) \urcorner, y) = 1 \wedge D(y))$$

If $q(\ulcorner B(z) \vee D(z) \urcorner, y)\downarrow$ then choose $s = q$. If $q(\ulcorner B(z) \vee D(z) \urcorner, y)$ is undefined then we can extend q to a forcing condition s by defining

$$s(\ulcorner B(z) \vee D(z) \urcorner, y) = \begin{cases} 0 & \text{if } B(y) \\ 1 & \text{otherwise} \end{cases}$$

Then $s \in C$ and by 2) we automatically get $(s(\ulcorner B(z) \vee D(z) \urcorner, y) = 0 \wedge B(y) \vee s(\ulcorner B(z) \vee D(z) \urcorner, y) = 1 \wedge D(y))$. So s does what it is supposed to do, and $\forall p \exists q \subset p$ such that q forces the axiom.
Now look at the less interesting extra axiom which makes sure that E acts like a function. $E(x,y,z) \wedge E(x,y,z') \rightarrow z = z'$. Let $\forall p \exists q \subset p.q \ \mathbf{f}\ E(x,y,z) \wedge E(x,y,z')$ this is $\forall p \exists q \subset p.q(x,y) = z \wedge q(x,y) = z'$. But q is a function so this gives $z = z'$, which is an atomic function. So q forces $z = z'$ and as a consequence $\forall p \exists q \subset p$ such that q forces the axiom.

To finish step one we will have to check the rest of the axioms as well. They are listed below. It is a bit more of a routine work to check these axioms then the axioms above. For a logical axiom T we will prove that $\forall p.p \ \mathbf{f}\ T$ and hence $\forall p \exists q \subset p.q \ \mathbf{f}\ T$.

Logical axioms

$A \wedge B \rightarrow A$,	Suppose $p \ \mathbf{f}\ A \wedge B$, then $p \ \mathbf{f}\ A$.
$A \wedge B \rightarrow B$,	Suppose $p \ \mathbf{f}\ A \wedge B$, then $p \ \mathbf{f}\ B$.
$A \rightarrow (B \rightarrow A \wedge B)$,	Let $p \ \mathbf{f}\ A$ and $q \subset p$ with $q \ \mathbf{f}\ B$.
	Then by monotinicity $q \ \mathbf{f}\ A \wedge B$.
$A \rightarrow A \vee B$,	Suppose $p \ \mathbf{f}\ A$, then $p \ \mathbf{f}\ A \vee B$.
$B \rightarrow A \vee B$,	Suppose $p \ \mathbf{f}\ B$, then $p \ \mathbf{f}\ A \vee B$.
$(A \rightarrow C)$	Suppose $p \ \mathbf{f}\ A \rightarrow C$. Let $q \subset p$ and $q \ \mathbf{f}\ B \rightarrow C$.
$\rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$,	Let $t \subset q$ and $t \ \mathbf{f}\ A \vee B$.
	Case one: $t \ \mathbf{f}\ A$. We have $t \subset p$,
	so by monotonicity $t \ \mathbf{f}\ A \rightarrow C$.
	Hence some extension of t forces C.
	Case two: $t \ \mathbf{f}\ B$. We have $t \subset q$,
	so by monotonicity $t \ \mathbf{f}\ B \rightarrow C$.
	Hence some extension of t forces C.
$A \rightarrow (B \rightarrow A)$,	Suppose $p \ \mathbf{f}\ A$. Let $q \subset p$ and $q \ \mathbf{f}\ B$.
	Then by monotonicity $q \ \mathbf{f}\ A$.
$(A \rightarrow (B \rightarrow C))$	Let $p \ \mathbf{f}\ A \rightarrow (B \rightarrow C)$. Let $q \subset p$ and $q \ \mathbf{f}\ A \rightarrow B$.
$\rightarrow ((A \rightarrow B), \rightarrow (A \rightarrow C))$	We will show $q \ \mathbf{f}\ A \rightarrow C$. Let $r \subset q.r \ \mathbf{f}\ A$.
	By monotonicity $r \ \mathbf{f}\ A \rightarrow B$,
	this means $\exists s \subset r \, s \ \mathbf{f}\ B$. We also have
	$s \ \mathbf{f}\ A \rightarrow (B \rightarrow C) \wedge s \ \mathbf{f}\ A$, since $s \subset r \subset p$.
	So $\exists v \subset s.v \ \mathbf{f}\ B \rightarrow C$.
	But by monotonicity $v \ \mathbf{f}\ B$.
	So $\exists w \subset v \subset q.w \ \mathbf{f}\ C$.
	This establishes $q \ \mathbf{f}\ A \rightarrow C$

$\forall x(B \to A) \to (B \to \forall x.A)$ $x \notin FV(B)$,	Suppose $p \mathbf{f} \forall x(B \to A)$. This means $\forall x \forall q \subset p \exists s \subset q \forall t \subset s(t \mathbf{f} B \to \exists v \subset t.v \mathbf{f} A)$. Let $a \subset p$ and $a \mathbf{f} B$. Now we need to derive $a \mathbf{f} \forall x.A(x)$. By monotonicity we have $\forall b \subset a.b \mathbf{f} B$. This gives us $\forall x \forall b \subset a \subset p \exists s \subset b \forall t \subset s(t \mathbf{f} B \to \exists v \subset t.v \mathbf{f} A(x))$. So we have $\forall x \forall b \subset a \exists v \subset b.c \mathbf{f} A(x)$. This is $a \mathbf{f} \forall x.A(x)$.
$\forall x\, A(x) \to A(t)$,	Suppose $p \mathbf{f} \forall x\, A(x)$ this is $\forall x \forall q \subset p \exists s \subset q.s \mathbf{f} A(x)$. Using the axiom we have to force, we get $\forall q \subset p \exists s \subset q.s \mathbf{f} A(t)$. So there is a $s \subset p$ such that $s \mathbf{f} A(t)$.
$\forall x(A \to B) \to (\exists x.A \to B)$ $x \notin FV(B)$,	Suppose $p \mathbf{f} \forall x(A \to B)$. Let $q \subset p.q \mathbf{f} \exists x A(x)$, this is $\exists x.q \mathbf{f} A(x)$. By monotonicity $q \mathbf{f} .\forall x(A \to B)(x)$, this is $\forall x \forall v \subset p \exists w \subset v.w \mathbf{f} (A \to B)(x)$. Using the axiom we have to force we get $\exists x \forall v \subset p \exists w \subset v.w \mathbf{f} A(x) \to B$. So there is a $s \subset w \subset q$ such that $s \mathbf{f} B$.
$A(t) \to \exists x\, A(x)$.	Let $p \mathbf{f} A(t) \to (p \mathbf{f} A)(t) \to \exists x.p \mathbf{f} A(x) \to p \mathbf{f} \exists x\, A(x)$.

The rule that is left is modus ponens. Suppose we have $\forall p \exists q \subset p.q \mathbf{f} A$, and $\forall s \exists t \subset s.t \mathbf{f} A \to B$ this is $\forall s \exists t \subset s \forall v \subset t(v \mathbf{f} A \to \exists w \subset v.v \mathbf{f} B)$. Fill in t for p and v for q and we get $\forall s \exists w \subset s.s \mathbf{f} B$.

Non-logical axioms
Just like for the logical axioms are all the equations, and implications between atomic formula forced by all the forcing conditions.
The last axiom to check is induction, $(A(0) \wedge \forall x(A(x) \to A(Sx))) \to \forall y.A(y)$.
Let $q \mathbf{f} (A(0) \wedge \forall x(A(x) \to A(Sx)))$. This means

$$q \mathbf{f} A(0),$$

$$\forall x \forall t \subset q \exists s \subset t.s \mathbf{f} A(x) \to A(Sx).$$

Then prove by induction on y that there is an extension of q that forces $A(y)$.
Case $A(0)$. This extension is just q.
Case $A(Sy)$. Suppose y is given and $r \subset q$ with $r \mathbf{f} A(y)$. Now r has an extension s forcing $A(y) \to A(Sy)$. By monotonicity $s \mathbf{f} A(y)$, so s has an extension that forces $A(Sy)$. That extension is the desired extension of q.

This finishes step one. Since we already proved part two, this also finishes the proof. □

2.4 $\mathbf{HAP}_\epsilon$ is conservative over $\mathbf{HAP}_E$

To finish our proof of $\mathbf{HAP}_\epsilon$ is conservative over $\mathbf{HAP}$ we need to prove that $\mathbf{HAP}_\epsilon$ is conservative over $\mathbf{HAP}_E$. We will apply the idea which was presented in [**?**] to our situation, but where [**?**] was about PCA's, is our situation about the syntactical case.

We will first prove in general a theorem stating that if there is in $\mathcal{L}(\mathbf{HAP})$ a relation symbol F which acts as a function like in appendix **??**, and there is an application, then we can construct a new application and a constant f that 'does the same' as F. At the same time interpreting the constants such that their interpretations act the same with this new application. The idea will be that this function F is an oracle function and when we make an new application we use F in the definition. This definition will enable us to "talk" to our function F in order to extract the information we need to build the constant f.

Afterwards we will apply this theorem to our situation which then allows us to present a simple proof.

Theorem 2.4.1. *Let F be a relation symbol with $F(x,y) \wedge F(x,y') \rightarrow y = y'$ and @ an application. We can define an interpretation $(\cdot)_F$ such that we can define a constant f with the following property, $\forall x. f@_F x = F(x)$. We also have that $A \leftrightarrow A_F$ holds for arithmetical A.*

Proof. First define A_F for all **HAP**-formulas A. The atomic formula $(t = s)_F$ is interpreted as $(t)_F = (s)_F$ and $(t@t')_F = (t)_F@_F(t')_F$. The interpretation for the constants and S are listed below, the notation is explained afterwards:

$0_F := 0$,
$S_F := S$,
$Succ_F := \lambda x.x + 1$,
$\mathbf{k}_F = \lambda x.\mathbf{p}@\mathbf{k}@(\lambda y.\mathbf{p}@\mathbf{k}@x_0)$,
$\mathbf{s}_F := \mathbf{s}_F = \lambda x.\mathbf{p}@\mathbf{k}@(\lambda y.\mathbf{p}@\mathbf{k}@t(x,y))$,
$\mathbf{p}_F = \lambda x.\mathbf{p}@\mathbf{k}@(\lambda y.\mathbf{p}@\mathbf{k}@(\mathbf{p}@x@y)$,
$\mathbf{p}_{0,F} = \lambda x.\mathbf{p}@\mathbf{k}@(\mathbf{p}_0@x)$,
$\mathbf{p}_{1,F} = \lambda x.\mathbf{p}@\mathbf{k}@(\mathbf{p}_1@x)$,
$\mathbf{d}_F = \lambda x.\mathbf{p}@\mathbf{k}@(\lambda y.\mathbf{p}@\mathbf{k}@(\lambda v.\mathbf{p}@\mathbf{k}@(\lambda w.\mathbf{p}@\mathbf{k}@(\mathbf{d}@x@y@v@w))))$.

Here the constants, which are terms, are interpreted as semiterms. In appendix **??** it is explained what happens to the formulas if one maps terms to semiterms. The conclusion is that we can work with A_F as a formula if A is a formula.

For these constants the same axioms need to hold as in **HAP** only now with application $@_F$. Here will be explained how the application $@_F$ works, why the axioms still holds, what t is, really everything we need to make this interpretation work:

To define $@_F$ we need an F-dialogue between $a, b \in \mathcal{L}(\mathbf{HAP})$. This is a code of a sequence $u = [u_0, \ldots, u_n]$ such that for all $i < n$ there is a $\mathbf{HAP}$-term v_i such that

$$a@([b] * u^{<i}) = \mathbf{p}@\overline{\mathbf{k}}@v_i \text{ and } F(v_i) = u_i$$

Now $a@_F b$ is defined with value c if there is a F-dialogue between a, b such that

$$a@([b] * u) = \mathbf{p}@\mathbf{k}@c$$

Here $u^{<i}$ denotes $[u_0, \ldots, u_{i-1}]$, ${}^{i\leq}u^{<j}$ denotes $[u_i, \ldots, u_{j-1}]$ and $\overline{\mathbf{k}} = \lambda x, y.y$, where $\mathbf{k}$ and $\overline{\mathbf{k}}$ work as booleans. With this definition is $\lambda x.\mathbf{p}@\mathbf{k}@x$ the lambda-abstraction of our new application. This explaines the definitions of the interpretations of the constants, only a part of $\mathbf{s}_F$ is not yet defined. Now we can, by primitive recursion, construct the term $t(x, y)$ of $\mathbf{HAP}$. We can in fact construct this such that for all u the application $t(x, y)@u$ is given by the following instructions, here $\text{Not}@\mathbf{k} = \overline{\mathbf{k}}$:

$t(x, y)@u =$

$x@u$ if $\forall i \leq \text{lh}u\ \text{Not}@(\mathbf{p}_0@(x@u^{<i}))$

If i is minimal such that $\mathbf{p}_0@(x@u^{<i})$, let $\alpha = \mathbf{p}_1@(x@u^{<i})$

and output $y@([u_0] * u^{\geq i})$ if

$\forall j(i \leq j < \text{lh}u \to \text{Not}@(\mathbf{p}_0@(y@([u_0] *^{i\leq} u^{<j}))))$

If j is minimal such that $\mathbf{p}_0@(y@([u_0] *^{i\leq} u^{<j}))$,

let $\beta = \mathbf{p}_1@(y@([u_0] *^{i\leq} u^{<j}))$and output $\alpha@([\beta] * u^{\geq} j)$

if $\forall k(j \leq k < \text{lh}u \to \text{Not}@(\mathbf{p}_0@(\alpha@([\beta] *^{j\leq} u^{<k}))))$

If k is minimal such that $\mathbf{p}_0@(\alpha@([\beta] *^{j\leq} u^{<k}))$,

output $\mathbf{p}_1@(\alpha@([\beta] *^{j\leq} u^{<k}))$.

This is an algorithm that follows an F-dialogue and says what it should do with it, to make sure that $\mathbf{s}_F$ works how it should work. $\mathbf{s}_F@_F x@_F y@_F z$ of course should work like this $(x@_F z)@_F(y@_F z)$. When we look at the alqorithm the first thing to observe is that u^0 is z. And $u^{>0}$ are the dialogues which occur when calculating $\mathbf{s}_F@_F x@_F y@_F z$. Since we want this to be $\mathbf{s}_F@_F x@_F y@_F z$, the algorithm first looks at the dialogue for calculating $x@_F z$, if that ends we put $\alpha = x@_F z$. Then it looks at $y@_F z$, if that ends as well we put $\beta = y@_F z$. Lastly it looks at $\alpha@_F\beta$ if that dialogue ends as well it outputs the correct value for $\mathbf{s}_F@_F x@_F y@_F z$. So note that $t(x, y)@_F z = (x@_F z)@_F(y@_F z)$. Therefore $\mathbf{s}_F$ works how it should work.

The last thing to define is f. f is a constant such that $f@_F x = F(x)$. f can easily be defined with the new application $@_F$ since you can ask for the value of the function F at x.

$$f@x = \begin{cases} \mathbf{p}@\overline{\mathbf{k}}@x & \text{if } x \text{ is a code of sequence of length one, say } x = [v] \\ \mathbf{p}@\mathbf{k}@v_1 & \text{if } x \text{ is a code of a finite sequence of length greater then one,} \\ & \text{say } x = [v_0, \ldots, v_n] \\ \uparrow & \text{otherwise} \end{cases}$$

This way if one wants to calculate $f@_F x$, one first looks at the dialogue $f@[x] = \mathbf{p}@\overline{\mathbf{k}}@x$ and our oracle F gives the value u_0, $F(x) = u_0$. Then $f@([x] * u_0) = \mathbf{p}@\mathbf{k}@u_0$, so f tracks F: $f@_F x = F(x) = u_0$. As a last remark we should note that the logical operators stay the same. So the only things that have changed are the constants and the application, both do not occur in the arithmetical formulas, so:

$$A \leftrightarrow A_F \quad \text{for arithmetical formula } A\ .$$

□

We will apply this more general case to our specific situation.

Theorem 2.4.2. $\boldsymbol{HAP}_\epsilon$ *is conservative over* $\boldsymbol{HAP}_E$

Proof. We will prove this with the following two steps

1. $\mathbf{HAP}_\epsilon \vdash A \Rightarrow \mathbf{HAP}_E \vdash A_E$ for all $\mathbf{HAP}_\epsilon$-formulas A,
2. $\mathbf{HAP}_E \vdash A_E \leftrightarrow A$ for all arithmetical formulas A.

Use the previous theorem, let $F := E$ and $f := e$, where e is the constant which tracks E. The interpretations stay the same we just add $\epsilon_E = e$. To prove step one, we have to check this for all the axioms of $\mathbf{HAP}_\epsilon$. Since we do not change the logical operators, but only the constants, and the application, a lot of the axioms simply just stay the same. The only axioms that might have changed from $\mathbf{HAP}$ are the axioms for the constants. But the new constants are constructed to make sure these axioms hold, like we saw for $\mathbf{s}_F$. For example, clearly $\mathbf{k}_E@_E x@_E y = (\lambda y.p@\mathbf{k}@x)@_E y = x$.
Now all that is left are the extra axioms from $\mathbf{HAP}_\epsilon$. But these will exactly be interpreted as the extra axioms from $\mathbf{HAP}_E$ as we will show for the first one.

$$\begin{aligned}
&(\exists x.B(x,y) \to (\epsilon@^\ulcorner \exists x.B(x,z)^\urcorner@y\downarrow \wedge B(\epsilon@^\ulcorner \exists x.B(x,z)^\urcorner@y), y)_E\\
=&\exists x.B(x,y) \to (e@_E{}^\ulcorner \exists x.B(x,z)^\urcorner@_E y\downarrow \wedge B(e@_E{}^\ulcorner \exists x.B(x,z)^\urcorner@_E y, y)\\
=&\exists x.B(x,y) \to (E(^\ulcorner \exists x.B(x,z)^\urcorner, y)\downarrow \wedge B(E(^\ulcorner \exists x.B(x,z)^\urcorner, y), y)).
\end{aligned}$$

To prove step two we have to prove that $\mathbf{HAP}_E \vdash A_E \leftrightarrow A$, for all arithmetical formulas A. But the only things that are changed in the $(\cdot)_E$ interpretation are the constants and the application. So $A_E = A$, for all arithmetical formulas A. This finishes the proof. □

Chapter 3

$\mathbf{HAP}_\epsilon$ and self-realising formulas

In this chapter we will give the definition of realizability and try to give an intuitionistic explanation and what it means to be self-realising. Then we will give our result that our system $\mathbf{HAP}_\epsilon$ has the interesting property that all arithmetical formulas are self-realising.

3.1 Realizability

In 1945 Kleene introduced realizability. To a constructivist or an intuitionist the things like existential, universal statements and the implication have a different meaning then for most people who practise classical logic. For example $\forall n.P(n)$ would mean something like: There is an effective method for any n to obtain the information that n has the property P. Kleene defined, with some notions from recursion theory, which items of information realizes a certain statement. The property of realizability will then be an intuitionistic truth notion.

This method ended up connecting intuitionism and recursive function theory.
With realizability we start with a PCA and define realizers for it, for the definition of a PCA see appendix **??**. Kleene gave his realizability originally for the PCA $\mathcal{K}_1$, the definition of $\mathcal{K}_1$ is also in appendix **??**. Kleene's definition of realizability specifies what it means for a natural number n to realize a formula A. Here we will give the definition of 'n realizes A' by induction on the complexity of A:

1. n realizes $(t = s)$ iff "$t = s$" is true.
2. n realizes $A \wedge B$ iff $n = \langle m, k\rangle$, and m realizes A and k realizes B. ($\langle\cdot,\cdot\rangle$ denotes a primitive recursive bijection from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$.)
3. n realizes $A \vee B$ iff either $n = \langle 0, m\rangle$ and m realizes A or $n = \langle 1, m\rangle$ and m realizes B.
4. n realizes $A \to B$ iff for each m realizing A, $\{n\}(m)$ is defined and realizes B.

5. n realizes $\exists x A(x)$ iff $n = \langle m, k\rangle$ and m realizes $A(\bar{k})$.
   ($\bar{k}$ is a term which denotes k.)

6. n realizes $\forall x A(x)$ iff for all m, $\{n\}(m)$ is defined and realizes $A(\bar{m})$.

This way n is a coding for all the information constructivists need for A.

Realizability can be formalised in **HAP** with respect to the application @. We shall associate to each formula A of **HAP** another formula $e\ \mathbf{r}\ A$.

Definition 3.1.1.

$$
\begin{array}{ll}
e\ \boldsymbol{r}\ A & \textit{is } A \textit{ if } A \textit{ is atomic},\\
e\ \boldsymbol{r}\ A \to B & \textit{is } \forall q(q\ \boldsymbol{r}\ A \to e@q\!\downarrow\ \wedge\ e@q\ \boldsymbol{r}\ B),\\
e\ \boldsymbol{r}\ A \wedge B & \textit{is } \boldsymbol{p}_0@e\ \boldsymbol{r}\ A\ \wedge\ \boldsymbol{p}_1@e\ \boldsymbol{r}\ B,\\
e\ \boldsymbol{r}\ A \vee B & \textit{is } \boldsymbol{p}_0@e = 0 \to \boldsymbol{p}_1@e\ \boldsymbol{r}\ A\ \wedge\ (\boldsymbol{p}_0@e \neq 0 \to \boldsymbol{p}_1@e\ \boldsymbol{r}\ B),\\
e\ \boldsymbol{r}\ \forall x\, A(x) & \textit{is } \forall x(e@x\!\downarrow\ \wedge\ e@x\ \boldsymbol{r}\ A(x)),\\
e\ \boldsymbol{r}\ \exists x\, A(x) & \textit{is } \boldsymbol{p}_0@e\ \boldsymbol{r}\ A(\boldsymbol{p}_1@e).
\end{array}
$$

$e\ \mathbf{r}\ A$ can be read as e realizes A.

Theorem 3.1.2 (Soundness of realizability)**.** *If* $\boldsymbol{HAP} \vdash A$ *then* $\boldsymbol{HAP} \vdash \exists e.e@y\ \boldsymbol{r}\ A$. *Here* y *are the free variables of* A.

Proof. To prove the soundness of realizability, you have to find a realizer for all the axioms of the system **HAP**.

Logical axioms

$(\lambda y, x.\mathbf{p}_0@x)@y\ \mathbf{r}\ A \wedge B \to A,\ \lambda y, x.\mathbf{p}_1@x\ \mathbf{r}\ A \wedge B \to B,$
$(\lambda y, x, z.\mathbf{p}@x@z)@y\ \mathbf{r}\ A \to (B \to (A \wedge B)),$
$(\lambda y, x.\mathbf{p}(0, x))@y\ \mathbf{r}\ A \to A \vee B;\ \lambda y, x.\mathbf{p}(1, x)\ \mathbf{r}\ B \to A \vee B,$
$(\lambda y, x, w, z.(1 - sg(\mathbf{p}_0@z))(x(\mathbf{p}_1@z)) + sg(\mathbf{p}_0@z)(w(\mathbf{p}_1@z)))@y\ \mathbf{r}$
$\quad (A \to C) \to ((B \to C) \to (A \vee B \to C)),$
$(\lambda y, x, z.x)@y\ \mathbf{r}\ A \to (B \to A),$
$(\lambda y, x, a, z.x@z@(a@z))@y\ \mathbf{r}\ (A \to (B \to C)) \to ((A \to B) \to (A \to C)),$
$(\lambda y, q, s, x.q@x@s)@y\ \mathbf{r}\ \forall x(B \to A) \to (B \to \forall x.A) \quad x \notin FV(B),$
$(\lambda y, q.q@t)@y\ \mathbf{r}\ \forall x.A \to A(t),$
$(\lambda y, q, s.(q@(\mathbf{p}_0@s))@(\mathbf{p}_0@s))@y\ \mathbf{r}\ \forall x(A \to B) \to (\exists x.A \to B) \quad x \notin FV(B),$
$(\lambda y, q.\mathbf{p}@q@t)@y\ \mathbf{r}\ A(t) \to \exists x.A.$

Here sg is the sign-function which is partial recursive:

$$sg(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise .} \end{cases}$$

The last one to check is modus ponens. If $e\ \mathbf{r}\ A$ and $f\ \mathbf{r}\ A \to B$ then $f@e\ \mathbf{r}\ B$.

Non-logical axioms
All the equations are realized by 0. Implications between atomic formulas

are realised by $\lambda x.0$. So the only axiom that is left is induction, **I**: $(A(0) \wedge \forall x\,(A(x) \to A(Sx))) \to \forall y\,A(y)$. To prove that an e realizes induction we have to prove that $e@q\ \mathbf{r}\ \forall y\,A(y)$ if $q\ \mathbf{r}\ A(0) \wedge \forall x(A(x) \to A(Sx))$. Let this be the case for q then

$$\mathbf{p}_0@q\ \mathbf{r}\ A(0),$$
$$\forall x(\mathbf{p}_1@q@x\downarrow\ \wedge \forall a(a\ \mathbf{r}\ A(x) \to \mathbf{p}_1@q@x@a\ \mathbf{r}\ A(Sx))).$$

Then define, by using the recursion combinator defined in **HAP**, a term $t(q)$, such that

$$t(q)@0 = \mathbf{p}_0@q,$$
$$t(q)@Sx = \mathbf{p}_1@q@x@(t(q)@x).$$

Then prove by induction on x, $t(q)@x\ \underline{\mathbf{r}}\ A(x)$.
$x = 0$; $t(q)@ = \mathbf{p}_0@q$ and $\mathbf{p}_0@q\ \mathbf{r}\ A(0)$
For Sx; $t(q)@Sx = \mathbf{p}_1@q@x@(t(q)@x)$, by the induction hypothesis $t(q)@x\ \mathbf{r}\ A(x)$.
So with the information we have of q, given by the fact that $q\ \mathbf{r}\ A(0) \wedge \forall x(A(x) \to A(Sx))$ we have $t(q)@Sx\ \mathbf{r}\ A(Sx)$.
So $e\ \mathbf{r}\ (A(0) \wedge \forall x(A(x) \to A(Sx))) \to \forall x\,A(x)$ with $e = \lambda q.t(q)$

□

If one thinks of being realizable as a truth notion it is natural to ask which formulas have the property $\exists e.e\ \mathbf{r}\ A$. Then one comes to the following definition

Definition 3.1.3. *A formula A is self-realizing if there is a closed semiterm $j_A@y$ such that **HAP** proves*

i) $A \to j_A@y\ \boldsymbol{r}\ A$,

ii) $(e\ \boldsymbol{r}\ A) \to A$.

Here $y = y_1, \ldots, y_n$ are the free variables of A.

The previous classical realizers do not take into account their functional behaviour. When we will prove Goodman's theorem for the extensional case we will need this. The following definition of extensional realizability does take their functional behaviour into account, this definition comes from [**?**]:

Definition 3.1.4.

$e\ \boldsymbol{r}_e\ A$	*is* A *if* A *is atomic,*
$a =_A b$	*is* $a = b \wedge A$,
$e\ \boldsymbol{r}_e\ A \to B$	*is* $\forall b, b'(b =_A b' \to (e@b\downarrow \wedge e@b' \downarrow \wedge e@b =_B e@b'))$,
$a =_{A\to B} b$	*is* $a\ \boldsymbol{r}_e\ A \to B \wedge b\ \boldsymbol{r}_e\ A \to B \wedge \forall c(c\ \boldsymbol{r}_e\ A \to a@c =_B b@c)$,
$e\ \boldsymbol{r}_e\ A \wedge B$	*is* $\boldsymbol{p}_0@e\ \boldsymbol{r}_e\ A\ \wedge\ \boldsymbol{p}_1@e\ \boldsymbol{r}_e\ B$,
$a =_{A\wedge B} b$	*is* $\boldsymbol{p}_0@a =_A \boldsymbol{p}_0@b \wedge \boldsymbol{p}_1@a =_B \boldsymbol{p}_1@b$,
$e\ \boldsymbol{r}_e\ A \vee B$	*is* $\boldsymbol{p}_0@e = 0 \to \boldsymbol{p}_1@e\ \boldsymbol{r}_e\ A\ \wedge\ \boldsymbol{p}_0@e \neq 0 \to p_1@e\ \boldsymbol{r}_e\ B$,
$a =_{A\vee B} b$	*is* $\boldsymbol{p}_0@a = \boldsymbol{p}_0@b \wedge \boldsymbol{p}_0@a = 0 \to \boldsymbol{p}_1@a =_A \boldsymbol{p}_1@b$ $\wedge\ \boldsymbol{p}_0@a \neq 0 \to \boldsymbol{p}_1@a =_B \boldsymbol{p}_1@b$,
$e\ \boldsymbol{r}_e\ \forall x\, A(x)$	*is* $\forall x(e@x\downarrow\ \wedge\ e@x\ \boldsymbol{r}_e\ A(x)))$,
$a =_{\forall x.A} b$	*is* $\forall x(a@x =_{A(x)} b@x')$,
$e\ \boldsymbol{r}_e\ \exists x.A(x)$	*is* $\boldsymbol{p}_0@e\ \boldsymbol{r}_e\ A(p_1@e)$,
$a =_{\exists x.A(x)} b$	*is* $\boldsymbol{p}_1@a = \boldsymbol{p}_1@b \wedge \boldsymbol{p}_0@a =_{A(\boldsymbol{p}_1@a)} \boldsymbol{p}_0@b$.

Here $a =_A b$ can be read as a and b are equal as realizers for A. With this definition we have $e =_A e \leftrightarrow e\ \mathbf{r}_e\ A$.

Theorem 3.1.5 (Soundness of extensional realizability)**.** *If* $\boldsymbol{HAP} \vdash A$ *then* $\boldsymbol{HAP} \vdash \exists e.e@y\ \boldsymbol{r}_e\ A$*. Here* y *are the free variables of* A*.*

Proof. The same realizers, as in the proof of the soundness of realizability above, can be used for the axioms of **HAP** to prove this theorem. □

3.2 Self-realising formulas

In **HA** all negative formulas are self-realising, a negative formula is a formula with no occurrence of $\vee$ or $\exists$. A problem arises when we try to make a positive formula self-realising, since we do not know for which x specifically $\exists x.A$ holds, and we do not know if A holds or B when we have $A \vee B$. In $\mathbf{HAP}_\epsilon$ there is a solution for this problem since we can ask our oracle ϵ these questions. Below we will give a proof that has as a corollary that in $\mathbf{HAP}_\epsilon$ all arithmetical formulas are self-realising. This proof holds in **HA** for negative formulas A as well. We will do this by giving a specific realizers e for which $A \leftrightarrow e\ \mathbf{r}\ A$ holds, we will call these realizers canonical realizers.

Definition 3.2.1. *The canonical realizers j_A for the arithmetical formulas A are:*

$$
\begin{aligned}
j_A &:= \lambda y.0 \quad \textit{If } A \textit{ is atomic },\\
j_{A\wedge B} &:= \lambda y.\boldsymbol{p}(j_A@y_A, j_B@y_B),\\
j_{\forall x.A} &:= \lambda y\lambda x.j_A@y@x,\\
j_{A\to B} &:= \lambda y\lambda x.j_B@y_B,\\
j_{\exists x.A} &:= \lambda y.\boldsymbol{p}@(j_A@y@(\epsilon@ < \exists x.A > @y))@(\epsilon@ < \exists x.A > @y),\\
j_{A\vee B} &:= \lambda y.\boldsymbol{p}@(\epsilon@(< A \vee B >)@y)@\alpha.
\end{aligned}
$$

$$
\alpha = \begin{cases} j_A@y & \textit{if } \epsilon@(< A \vee B >)@y = 0\\ j_B@y & \textit{if } \epsilon@(< A \vee B >)@y = 1. \end{cases}
$$

Here $y = y_1, \ldots, y_n$ are the free variables of A. If it is not clear that the free variables belong to formula A we will write y_A.

Lemma 3.2.2. *Let A be an arithmetical formula in $\mathcal{L}(\boldsymbol{HAP}_\epsilon)$ then*

$$
\begin{aligned}
\boldsymbol{HAP}_\epsilon \vdash & A\\
&\leftrightarrow \exists e.e\ \boldsymbol{r}\ A\\
&\leftrightarrow j_A@y\ \boldsymbol{r}\ A\\
&\leftrightarrow \exists e.e\ \boldsymbol{r}_e\ A
\end{aligned}
$$

Proof. We will first prove $\mathbf{HAP}_\epsilon \vdash \exists e.e\ \mathbf{r}\ A \to A \to j_A@y\ \mathbf{r}\ A \to \exists e.e\ \mathbf{r}\ A$ by induction on the structure of A. Here the last implication is trivial.
Let A be atomic

$$
\begin{aligned}
\mathbf{HAP}_\epsilon \vdash\ & \exists e.e\ \mathbf{r}\ A\\
&\to \exists e.A\\
&\to A\\
&\to 0\ \mathbf{r}\ A\\
&\to \exists e.e\ \mathbf{r}\ A
\end{aligned}
$$

Look at $A \wedge B$

$$
\begin{aligned}
\mathbf{HAP}_\epsilon \vdash\ & \exists e.e\ \mathbf{r}\ A \wedge B\\
&\to \exists e.\mathbf{p}_0@e\ \mathbf{r}\ A \wedge \mathbf{p}_1@e\ \mathbf{r}\ B\\
&\to A \wedge B\\
&\to j_A@y_A\ \mathbf{r}\ A \wedge j_B@y_B\ \mathbf{r}\ B\\
&\to j_{A\wedge B}@y\ \mathbf{r}\ A \wedge B\\
&\to \exists e.e\ \mathbf{r}\ A \wedge B
\end{aligned}
$$

Look at $\forall x.A$

$$
\begin{aligned}
\mathbf{HAP}_\epsilon \vdash\ & \exists e.e\ \mathbf{r}\ \forall x.A\\
&\to \exists e.\forall x(e@x\downarrow \wedge e@x\ \mathbf{r}\ A)\\
&\to \forall x.A\\
&\to \forall x(j_A@y@x\downarrow \wedge j_A@y@x\ \mathbf{r}\ A)\\
&\to j_{\forall x.A}@y\ \mathbf{r}\ \forall x.A\\
&\to \exists e.e\ \mathbf{r}\ \forall x.A
\end{aligned}
$$

Look at $A \to B$

$$\begin{aligned}
\mathbf{HAP}_\epsilon \vdash\ & \exists e.e\ \mathbf{r}\ A \to B \\
& \to \exists e.\forall q(q\ \mathbf{r}\ A \to e@q\downarrow \wedge e@q\ \mathbf{r}\ B) \\
& \to A \to B \\
& \to \forall q(q\ \mathbf{r}\ A \to j_B@y_B\downarrow \wedge j_B@y_B\ \mathbf{r}\ B) \quad \text{since } q\ \mathbf{r}\ A \to A \text{ and } A \to B \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{and } B \to j_B@y_B\ \mathbf{r}\ B \\
& \to j_{A\to B}@y\ \mathbf{r}\ A \to B \\
& \to \exists e.e\ \mathbf{r}\ A \to B
\end{aligned}$$

Look at $\exists x.A(x,y)$. Use the extra axioms in $\mathbf{HAP}_\epsilon$, the induction hypothesis for $A(x,y)$, and the definition of realizability.

$$\begin{aligned}
\mathbf{HAP}_\epsilon \vdash\ & \exists e.e\ \mathbf{r}\ \exists x.A(x,y) \\
& \to \exists e.\mathbf{p}_0@e\ \mathbf{r}\ A(\mathbf{p}_1@e,y) \\
& \to A(\mathbf{p}_1@e,y) \\
& \to \exists x.A(x,y) \\
& \to \epsilon@{\ulcorner}\exists x.A(x,z){\urcorner}@y\downarrow \wedge A(\epsilon@{\ulcorner}\exists x.A(x,z){\urcorner}@y,y) \\
& \to j_A@y@(\epsilon@{\ulcorner}\exists x.A(x,z){\urcorner}@y)\ \mathbf{r}\ A(\epsilon@{\ulcorner}\exists x.A(x,z){\urcorner}@y,y) \\
& \quad \to j_{\exists x.A(x,y)}@y\ \mathbf{r}\ \exists x.A(x,y) \\
& \quad \to \exists e.e\ \mathbf{r}\ \exists x.A(x,y)
\end{aligned}$$

Look at $A(y) \vee B(y)$. Use the extra axioms in $\mathbf{HAP}_\epsilon$, the induction hypothesis for $A(y)$ and $B(y)$, and the definition of realizability.

$$\begin{aligned}
\mathbf{HAP}_\epsilon \vdash\ & \exists e.e\ \mathbf{r}\ A(y) \vee B(y) \\
& \to (\mathbf{p}_0@e = 0 \to \mathbf{p}_1@e\ \mathbf{r}\ A(y)) \ \wedge\ (\mathbf{p}_0@e \neq 0 \to \mathbf{p}_1@e\ \mathbf{r}\ B(y)) \\
& \to \mathbf{p}_1@e\ \mathbf{r}\ A(y) \vee \mathbf{p}_1@e\ \mathbf{r}\ B(y) \\
& \to A(y) \vee B(y) \\
& \to \epsilon@({\ulcorner}B(z) \vee D(z){\urcorner})@y\downarrow \wedge((E@({\ulcorner}B(z) \vee D(z){\urcorner})@y = 0 \wedge B(y) \\
& \qquad \vee\ \epsilon@({\ulcorner}B(z) \vee D(z){\urcorner})@y = 1 \wedge D(y))) \\
& \quad \to j_{A\vee B}@y\ \mathbf{r}\ A(y) \vee B(y) \\
& \quad \to \exists e.e\ \mathbf{r}\ A(y) \vee B(y)
\end{aligned}$$

Now what is left to prove is that $\exists e.e\ \mathbf{r}_e\ A$ is equivalent to the other statements. We will prove this by proving the following, $j_A@y\ \mathbf{r}\ A \to j_A@y\ \mathbf{r}_e\ A \to \exists e.e\ \mathbf{r}_e\ A \to \exists e.e\ \mathbf{r}\ A$. The second implication is trivial, the first and last implication can be proven by induction on the structure of A. The only interesting case is the implication so look at $A \to B$.

$$\begin{aligned}
\mathbf{HAP}_\epsilon \vdash & j_{A\to B}@y\ \mathbf{r}\ A \to B \\
& \to \forall q(q\ \mathbf{r}\ A \to j_B@y_B\downarrow \wedge j_B@y_B\ \mathbf{r}\ B) \\
& \to \forall q(q\ \mathbf{r}\ A \to j_B@y_B\downarrow \wedge j_B@y_B\ \mathbf{r}_e\ B) \wedge (\forall b,b'(b =_A b' \to b =_A b \to b\ \mathbf{r}\ A) \\
& \to \forall b,b'(b =_A b' \to (j_B@y_B\downarrow \wedge j_B@y_B =_B j_B@y_B)) \\
& \to j_{A\to B}@y\ \mathbf{r}_e\ A \to B
\end{aligned}$$

$\mathbf{HAP}_\epsilon \vdash e\ \mathbf{r}_e\ A \to B$
$\quad \to \forall q, q'(q =_A q' \to e@q\downarrow \wedge e@q'\downarrow \wedge e@q =_B e@q'$
$\quad \to \forall q((q =_A q \to e@q\downarrow \wedge e@q =_B e@q$
$\quad \to \forall q(q\ \mathbf{r}_e\ A \to e@q\downarrow \wedge e@q\ \mathbf{r}_e\ B$
$\quad \to \forall q(q\ \mathbf{r}\ A \to e@q\downarrow \wedge e@q\ \mathbf{r}\ B$

This proves the original statement □

Corollary 3.2.3. *In* $\boldsymbol{HAP_\epsilon}$ *every arithmetical formula is self-realizing.*

Corollary 3.2.4. *In* ***HA*** *every negative arithmetical formula is self-realizing.*

Chapter 4

Application: Goodman's theorem

In this chapter we will give a proof for the fact that $\mathbf{E\text{-}HA}^\omega + \mathbf{AC}$ and $\mathbf{I\text{-}HA}^\omega + \mathbf{AC}$ are conservative over Heyting Arithmetic. The results from the previous chapters already did the hardest part of the work for this result. What is left is to prove that **HAP** is conservative over **HA** and that $\mathbf{E\text{-}HA}^\omega + \mathbf{AC}$ and $\mathbf{I\text{-}HA}^\omega + \mathbf{AC}$ are conservative over $\mathbf{HAP}_\epsilon$. We will be able to prove the second part by choosing the appropriate notion of realisability and build realisers for the axiom of choice and the extensionality axiom. Then we will use the fact that in $\mathbf{HAP}_\epsilon$ all arithmetical formulas are self-realising to complete the proof.

4.1 HAP is conservative over HA

Theorem 4.1.1. ***HAP*** *is conservative over* ***HA***

Proof. This can be proven with the following proof interpretation. We need a proof for

1. $\mathbf{HAP} \vdash A \Rightarrow \mathbf{HA} \vdash [A]$ for all **HAP**-formulas A.
2. $\mathbf{HA} \vdash [A] \leftrightarrow A$ for all arithmetical formulas A.

We need to define the interpretion $[\cdot]$. We will do this by defining the interpretation for all the constants, functions and the application which were added to the language **HAP**. We do this by using the interpretation of **APP** in the

partial recursive operations (PRO) from [**?**].

$[hap(x,y,z)] := \{[x]\}([y]) = [z]$,
$[0] := 0$,
$[S] := S$,
$[Succ] := \lambda x.Sx$,
$[\mathbf{p}] := \lambda x, y.j(x,y)$,
$[\mathbf{p}_i] := \lambda x.j_{i+1}x \quad (i = 0, 1)$,
$[\mathbf{k}] := \lambda x, y.x$,
$[\mathbf{s}] := \lambda x, y, z.\{\{x\}(z)\}(\{y\}(z))$,
$[\mathbf{d}] := \lambda x, y, u, v.x \cdot (1 - sg)|u - v| + y \cdot sg|u - v|$.

Here sg is again the sign function. Under the interpretation the logical operators will stay the same.

First note that step two is trivial, since the interpretian $[\cdot]$ only changes the things which were added to $\mathcal{L}(\mathbf{HA})$ to make $\mathcal{L}(\mathbf{HAP})$. So for all arithmetical A, $[A] = A$.

Step one can be proven by checking this for all the axioms of **HAP**. Step one is also trivial for the logical axioms, since the interpretation does nothing to the logical symbols and the logical basis of **HA** and **HAP** are the same. Then we have left the axioms defining the constants and the arithmetical axioms. The interpretation $[\cdot]$ is defined in a way such that the axioms defining the constants stay valid in **HA**.
For example

$$\begin{aligned}[\mathbf{p}_0@(\mathbf{p}@x@y)] &= \{[\mathbf{p}_0]\}([\mathbf{p}@x@y]) \\ &= j_1(j(x,y)) \\ &= x.\end{aligned}$$

The arithmetical axioms from **HAP** are also axioms of **HA** and the interpretation $[\cdot]$ does not change them.
This proves step one and hence the theorem. □

4.2 E-HA$^\omega$ + AC and I-HA$^\omega$ + AC are conservative over HAP$_\epsilon$

In this section we will prove that both **I-HA**$^\omega$ + **AC** and **E-HA**$^\omega$ + **AC** are conservative over **HAP**$_\epsilon$. The difference between the two proofs is that in the first proof everything is intensional and in the second everything is extensional, which means we have to realize the intensionality axiom or the extensionality axiom. In order to realise these axioms and **AC** we will need two different notions of realisability. But first of all before we can get to those proofs, we have to define a type structure over **HAP** since **HA**$^\omega$ has a type structure. This will help us to define the notion of realizability we need.

4.2.1 Finite type structures over HAP

In order to define our finite type structures we will use sets to write down what we mean instead of the **HAP**-formulas needed for the formal definition, because this way it is a lot clearer what is meant. Later on we will also use $x \in I_\sigma$ as an abbreviation for the formal **HAP**-formula.

An intensional finite type structure over **HAP** which associates to any type σ a set I_σ is as follows:

$$\begin{aligned} I_0 &= \{a | a = a\}, \\ I_{\sigma\to\tau} &= I_\sigma \Rightarrow I_\tau \\ &= \{a | \forall b \in I_\sigma (a@b\downarrow \text{ and } a@b \in I_\tau)\}. \end{aligned}$$

An extensional finite type structure will be defined recursively, simultaneously with the symmetric transitive relation $\sim_\sigma$:

$$E_0 = \{a | a = a\}, \quad a \sim_0 b \leftrightarrow a = b,$$

$$\begin{aligned} E_{\sigma\to\tau} = \{a | &\forall x \in E_\sigma (a@x\downarrow \wedge a@x \in E_\tau) \wedge \forall x, y \in E_\sigma ((x \sim_\sigma y), \\ &\to (a@x \sim_\tau a@y))\}, \\ a \sim_{\sigma\to\tau} b \leftrightarrow &\forall x, y \in E_\sigma (x \sim_\sigma y \to a@x\downarrow \wedge b@y\downarrow \wedge a@x \sim_\tau b@y). \end{aligned}$$

4.2.2 I-HA$^\omega$ + AC is conservative over HAP$_\epsilon$

Here we will take the last step towards proving Goodman's Theorem. We will 'replace' the axiom of choice by our oracle ϵ and get rid of the types. We will do this by finding realisers for all the formulas which are valid in **I-HA**$^\omega$ + **AC**. This sounds like a lot of work but all we have to do is find realizers for the basis of **I-HA**$^\omega$ + **AC**, for the axioms. Since the rest of the formulas valid in **I-HA**$^\omega$ + **AC** follow from the axioms is this enough to prove that we have realizers for all the formulas. Then we will use the fact that all the arithmetical formulas are self-realising in **HAP**$_\epsilon$ to finish the proof.

Theorem 4.2.1. **I-HA**$^\omega$ + **AC** *is conservative over* ***HAP***$_\epsilon$

Proof. We will prove this with the following steps

1. **I-HA**$^\omega$ + **AC** $\vdash A \Rightarrow$ **HAP**$_\epsilon \vdash \exists e\, e\, \underline{\mathbf{r}}\, [A]$ for all **HA**$^\omega$-formulas.
2. **HAP**$_\epsilon \vdash \exists e\, e\, \underline{\mathbf{r}}\, [A] \leftrightarrow A$ for all arithmetical formulas A.

First define the formula $e\, \underline{\mathbf{r}}\, [A] \in$ **HAP**$_\epsilon$ for every **HA**$^\omega$-formulas A

Definition 4.2.2.

$e\, \underline{\boldsymbol{r}}\, [A]$	*is* $[A]$ *if* A *is atomic,*
$e\, \underline{\boldsymbol{r}}\, [A \to B]$	*is* $\forall q (q\, \underline{\boldsymbol{r}}\, [A] \to e@q\downarrow \wedge e@q\, \underline{\boldsymbol{r}}\, [B])$,
$e\, \underline{\boldsymbol{r}}\, [A \wedge B]$	*is* $\boldsymbol{p}_0@e\, \underline{\boldsymbol{r}}\, [A] \wedge \boldsymbol{p}_1@e\, \underline{\boldsymbol{r}}\, [B]$,
$e\, \underline{\boldsymbol{r}}\, [A \vee B]$	*is* $(\boldsymbol{p}_0@e = 0 \to \boldsymbol{p}_1@e\, \underline{\boldsymbol{r}}\, [A]) \wedge (\boldsymbol{p}_0@e \neq 0 \to p_1@e\, \underline{\boldsymbol{r}}\, [B])$,
$e\, \underline{\boldsymbol{r}}\, [\forall x^\sigma\, A(x)]$	*is* $\forall x (x \in I_\sigma \to (e@x\downarrow \wedge e@x\, \underline{\boldsymbol{r}}\, [A(x)]))$,
$e\, \underline{\boldsymbol{r}}\, [\exists x^\sigma\, A(x)]$	*is* $\boldsymbol{p}_1@e \in I_\sigma \wedge \boldsymbol{p}_0@e\, \underline{\boldsymbol{r}}\, [A(p_1@e)]$.

Note that we integrated the intentional type structure I_σ into our notion of realizability, this ensures we do not lose the meaning of the types in the quantifiers.

The atomic formulas $[t =_\sigma s]$ are interpreted as $[t] = [s]$ and $[Ap(t, t')]$ as $[t]@[t']$. The constants are interpreted as follows

$$\begin{array}{l} [0] := 0 \quad \text{(both zero's are the zero from } \mathbf{HA}\text{)}, \\ [S] := S \quad \text{(both S's are the S from } \mathbf{HA}\text{)}, \\ [k^{\sigma,\tau}] := \mathbf{k}, \\ [s^{\rho,\sigma,\tau}] := \mathbf{s}, \\ [p^{\sigma,\tau}] := \mathbf{p}, \\ [p_i^{\sigma,\tau}] := \mathbf{p}_i \ (i = 0, 1) \\ [r^\sigma] := rec. \end{array} ,$$

Note that $\mathbf{k}, \mathbf{s}, \mathbf{p}, \mathbf{p}_i$ are in the appropriate I_σ. For example $\forall \sigma, \tau(\mathbf{k} \in I_{\sigma\to(\tau\to\sigma)})$: $\forall x \in I_\sigma \mathbf{k}@x\downarrow$ and $k@x \in I_{\sigma\to\tau}$, since $\forall y \in I_\tau.\mathbf{k}@x@y = x \in I_\sigma$.
For step one we need to find for all the axioms A of $\mathbf{I}\text{-}\mathbf{HA}^\omega + \mathbf{AC}$ a realizer e such that $e \ \underline{\mathbf{r}}\ [A]$.
Let us look at the most important axiom first, the axiom of choice.

$$\mathbf{AC}: \ \forall x^\sigma \exists y^\tau A(x, y) \to \exists f^{\sigma\to\tau} \forall x^\sigma A(x, f(x)).$$

We show that the realizer t works as a realizer for AC, where t is:

$$t := \lambda q.\mathbf{p}@(\lambda x.\mathbf{p}_0@(q@x))@(\lambda x.\mathbf{p}_1@(q@x)).$$

First we assume that $q \ \underline{\mathbf{r}}\ [\forall x^\sigma \exists y^\tau \ A(x, y)]$ this is

$$\forall x(x \in I_\sigma \to q@x\downarrow \ \wedge \ \mathbf{p}_1@(q@x) \in I_\tau \ \wedge \ \mathbf{p}_0@(q@x) \ \underline{\mathbf{r}}\ [A(x, \mathbf{p}_1@(q@x))]).$$

For $t \ \underline{\mathbf{r}}\ [\mathbf{AC}]$ we need to prove $t@q \ \underline{\mathbf{r}}\ [\exists f^{\sigma\to\tau} \forall x^\sigma A(x, f(x))]$, which means we need to prove:

$$t@q\downarrow \wedge \mathbf{p}_1@(t@q) \in I_{\sigma\to\tau} \wedge \ \forall x(x \in I_\sigma \to \mathbf{p}_0@(t@q)@x \ \underline{\mathbf{r}}\ [A(x, \mathbf{p}_1@(t@q)@x)])$$

By the construction of t; $\mathbf{p}_1@(t@q) = \lambda x.\mathbf{p}_1@(q@x)$, where $x \in I_\sigma$ and $\mathbf{p}_1@(t@q) \in I_\tau$. So by definition $\mathbf{p}_1@(t@q) \in I_{\sigma\to\tau}$. $\mathbf{p}_0@(t@q)@x = \mathbf{p}_0@(q@x)$ and $\mathbf{p}_1@(t@q)@x = \mathbf{p}_1@(q@x)$. This gives, since $q \ \underline{\mathbf{r}}\ [\forall x^\sigma \exists y^\tau \ A(x, y)]$, that $\forall x(x \in I_\sigma \to \mathbf{p}_0@(t@q)@x \ \underline{\mathbf{r}}\ [A(x, \mathbf{p}_1@(t@q)@x)]$. Putting these things together and one gets, $t \ \underline{\mathbf{r}}\ [AC]$.

Now let us look at the intensionality axiom

$$\begin{array}{ll} \mathbf{I}: \mathbf{e}^\sigma xy = 0 \vee \mathbf{e}^\sigma xy = 1 & \text{here } \mathbf{e} \text{ is an equality functional}, \\ \quad \mathbf{e}^\sigma xy = 0 \leftrightarrow x =_\sigma y. & \end{array}$$

The two axioms defining the equality functional are both really easy to realise since both are not much more then atomic formulas. So we get

$$\mathbf{p}@0@0 \ \underline{\mathbf{r}}\ (\mathbf{e}^\sigma@x@y = 0 \vee \mathbf{e}^\sigma@x@y = 1) \vee \mathbf{p}@1@0 \ \underline{\mathbf{r}}\ (\mathbf{e}^\sigma@x@y = 0 \vee \mathbf{e}^\sigma@x@y = 1),$$
$$\mathbf{p}@\lambda q.0@\lambda q.0 \ \underline{\mathbf{r}}\ (\mathbf{e}^\sigma@x@y = 0 \to x = y \wedge x = y \to \mathbf{e}^\sigma@x@y = 0).$$

Now look at the rest of the axioms of $\mathbf{I\text{-}HA}^\omega + \mathbf{AC}$.

Logical axioms

$\lambda x.\mathbf{p}_0@x \ \underline{\mathbf{r}}\ [A \wedge B \to A]$; $\lambda x.\mathbf{p}_1@x \ \underline{\mathbf{r}}\ [A \wedge B \to B]$,
$\lambda x, y.\mathbf{p}(x,y) \ \underline{\mathbf{r}}\ [A \to (B \to A \wedge B)]$,
$\lambda x.\mathbf{p}(0,x) \ \underline{\mathbf{r}}\ [A \to A \vee B]$; $\lambda x.\mathbf{p}(1,x) \ \underline{\mathbf{r}}\ [B \to A \vee B]$,
$\lambda x, y, z.(1 - sg(\mathbf{p}_0@z))(x(\mathbf{p}_1@z)) + sg(\mathbf{p}_0@z)(y(\mathbf{p}_1@z)) \ \underline{\mathbf{r}}$
$[(A \to C) \to ((B \to C) \to (A \vee B \to C))]$,
$\lambda x, y.x \ \underline{\mathbf{r}}\ [A \to (B \to A)]$,
$\lambda x, y, z.x@z@(y@z) \ \underline{\mathbf{r}}\ [(A \to (B \to C)) \to ((A \to B) \to (A \to C))]$,
$\lambda q, s, x.q@x@s \ \underline{\mathbf{r}}\ [\forall x^\sigma (B \to A) \to (B \to \forall x\, A)] \quad x \notin FV(B)$,
$\lambda q.\mathbf{p}_0@q@t \ \underline{\mathbf{r}}\ [\forall x^\sigma.A \to A(t^\sigma)]$,
$\lambda q, s.(q@(\mathbf{p}_1@s))@(p_0@s) \ \underline{\mathbf{r}}\ [\forall x^\sigma (A \to B) \to (\exists x^\sigma\, A(x) \to B)] \quad x \notin FV(B)$,
$\lambda q.\mathbf{p}(q,t) \ \underline{\mathbf{r}}\ [A(t^\sigma) \to \exists x^\sigma\, A(x)]$.

The last one to check is modus ponens. If $e \ \underline{\mathbf{r}}\ [A]$ and $f \ \underline{\mathbf{r}}\ [A \to B]$ then $f@e \ \underline{\mathbf{r}}\ [B]$.

Non-logical axioms
All the equations are realized by 0. Implications between atomic formulas are realised by $\lambda x.0$. So the only axiom that is left is induction, **I**: $(A(0) \wedge \forall x^0(A(x) \to A(Sx))) \to \forall x^0\, A(x)$.
Suppose $q \ \underline{\mathbf{r}}\ [A(0) \wedge \forall x^0(A(x) \to A(Sx))]$, then we will construct an e such that $e@q \ \underline{\mathbf{r}}\ [\forall x^0\, A(x)]$. This means that we have

$$\mathbf{p}_0@q \ \underline{\mathbf{r}}\ [A(0)],$$
$$\forall x(\mathbf{p}_1@q@x\downarrow \ \wedge \forall a(a \ \underline{\mathbf{r}}\ [A(x)] \to \mathbf{p}_1@q@x@a \ \underline{\mathbf{r}}\ [A(Sx)]),$$

and we need to prove

$$\forall x(e@q@x\downarrow \wedge e@q@x \ \underline{\mathbf{r}}\ [A(x)].$$

Define a term $t(q)$, by using the recursor combinator rec, such that

$t(q)@0 = \mathbf{p}_0@q$,
$t(q)@Sx = \mathbf{p}_1@q@x@(t(q)@x)$.

Then prove by induction on x, $\forall x(t(q)@x\downarrow \wedge t(q)@x \ \underline{\mathbf{r}}\ [A(x)]$.
$x = 0$; $t(q)@0 = \mathbf{p}_0@q$ and $\mathbf{p}_0@q \ \underline{\mathbf{r}}\ [A(0)]$
For Sx; $t(q)@Sx = \mathbf{p}_1@q@x@(t(q)@x)$, by the induction hypothesis $t(q)@x \ \underline{\mathbf{r}}\ [A(x)]$.
So $t(q)@Sx \ \underline{\mathbf{r}}\ [A(Sx)]$.
So $\lambda q.t(q) \ \underline{\mathbf{r}}\ [(A(0) \wedge \forall x^\sigma(A(x) \to A(Sx))) \to \forall x\, A(x)]$.

This finishes step one.

For step two we should realise that for all arithmetical formulas A we only have variables of type 0. Then $a \in I_\sigma$ always means $a \in I_0$, but everything is

in I_0. So for arithmetical A, $e\ \underline{\mathbf{r}}\ [A] = e\ \mathbf{r}\ A$. Together with lemma **??** this gives the following result

$$\mathbf{HAP}_\epsilon \vdash \exists e\, e\ \underline{\mathbf{r}}\ [A] \leftrightarrow \exists e\, e\ \mathbf{r}\ A \leftrightarrow A.$$

□

4.2.3 E-HA$^\omega$ + AC is conservative over HAP$_\epsilon$

To prove that **E-HA**$^\omega$ + AC is conservative over $\mathbf{HAP}_\epsilon$ we have to do the same steps as above only now we have to realize the extensionality axiom as well. In order to make that possible we need a different notion of realizability. Namely a notion which makes it possible to find realizers for the extensionality axiom and the axiom of choice. With this notion a fixed arithmetical formula A still needs to be self-realising. The extensional realisability with the addition of the extensional finite type structure makes this possible. In the second step there could arise a problem with the realizer for an implication. Since that is where the two types of realizability really differ, but this is not a problem for the reason that we proved in lemma **??** that the canonical realizers can be used as self-realizers in the extensional realisability as well as the normal variant. This can be done since in the canonical realizer for the implication the hypothesis is ignored.

Theorem 4.2.3. **E-HA**$^\omega$ + **AC** *is conservative over* $\boldsymbol{HAP}_\epsilon$

Proof. We will prove this with the following steps

1. $\vdash A \Rightarrow \mathbf{HAP}_\epsilon \vdash \exists e\, e\ \underline{\mathbf{r}}_e\ [A]$ for all $\mathbf{HA}^\omega$-formulas.
2. $\mathbf{HAP}_\epsilon \vdash \exists e\, e\ \underline{\mathbf{r}}_e\ [A] \leftrightarrow A$ for all arithmetical formula A.

First define the **HAP** formula $e\ \underline{\mathbf{r}}_e\ [A]$ for every $\mathbf{HA}^\omega$-formula A

Definition 4.2.4.

$e\ \underline{\boldsymbol{r}}_e\ [A]$	*is* $[A]$ *if* A *is atomic*,
$a =_{[A]} b$	*is* $a = b \wedge [A]$,
$e\ \underline{\boldsymbol{r}}_e\ [A \to B]$	*is* $\forall b, b'(b =_{[A]} b' \to (e@b\downarrow \wedge e@b'\downarrow \wedge e@b =_{[B]} e@b'))$,
$a =_{[A\to B]} b$	*is* $a\ \underline{\boldsymbol{r}}_e\ [A \to B] \wedge b\ \underline{\boldsymbol{r}}_e\ [A \to B] \wedge \forall c(c\ \boldsymbol{r}_e\ [A] \to a@c =_{[B]} b@c)$,
$e\ \underline{\boldsymbol{r}}_e\ [A \wedge B]$	*is* $\boldsymbol{p}_0@e\ \underline{\boldsymbol{r}}_e\ [A]\ \wedge\ \boldsymbol{p}_1@e\ \underline{\boldsymbol{r}}_e\ [B]$,
$a =_{[A\wedge B]} b$	*is* $\boldsymbol{p}_0@a =_{[A]} \boldsymbol{p}_0@b \wedge \boldsymbol{p}_1@a =_{[B]} \boldsymbol{p}_1@b$,
$e\ \underline{\boldsymbol{r}}_e\ [A \vee B]$	*is* $\boldsymbol{p}_0@e = 0 \to \boldsymbol{p}_1@e\ \underline{\boldsymbol{r}}_e\ [A]\ \wedge\ \boldsymbol{p}_0@e \neq 0 \to p_1@e\ \underline{\boldsymbol{r}}_e\ [B]$,
$a =_{[A\vee B]} b$	*is* $\boldsymbol{p}_0@a = \boldsymbol{p}_0@b \wedge \boldsymbol{p}_0@a = 0 \to \boldsymbol{p}_1@a =_{[A]} \boldsymbol{p}_1@b$ $\wedge\, \boldsymbol{p}_0@a \neq 0 \to \boldsymbol{p}_1 =_{[B]} \boldsymbol{p}_1@b$,
$e\ \underline{\boldsymbol{r}}_e\ [\forall x^\sigma\ A(x)]$	*is* $\forall x(x \in E_\sigma \to (e@x\downarrow\ \wedge e@x\ \underline{\boldsymbol{r}}_e\ [A(x)]))$,
$a =_{[\forall x^\sigma . A]} b$	*is* $\forall x(x \in E_\sigma \to b@x'\downarrow \wedge a@x =_{[A(x)]} b@x')$,
$e\ \underline{\boldsymbol{r}}_e\ [\exists x^\sigma . A(x)]$	*is* $\boldsymbol{p}_1@e \in E_\sigma\ \wedge\ \boldsymbol{p}_0@e\ \underline{\boldsymbol{r}}_e\ [A(p_1@e)]$,
$a =_{[\exists x^\sigma . A(x)]} b$	*is* $\boldsymbol{p}_1@a \sim_\sigma \boldsymbol{p}_1@b \wedge \boldsymbol{p}_1@a \in E_\sigma \wedge \boldsymbol{p}_0@a =_{[A(\boldsymbol{p}_1@a)]} \boldsymbol{p}_0@b$.

Remember that with this definition we have $e =_{[A]} e \leftrightarrow e\ \underline{\mathbf{r}}_e\ [A]$. The atomic formula $[t =_\sigma s]$ is interpreted as $[t] = [s]$ and $[Ap(t, t')]$ as $[t]@[t']$. The constants are interpreted as follows

$[0] := 0$ (both zero's are the zero from **HA**),
$[S] := S$ (both S's are the S from **HA**),
$[k^{\sigma,\tau}] := \mathbf{k}$,
$[s^{\rho,\sigma,\tau}] := \mathbf{s}$,
$[p^{\sigma,\tau}] := \mathbf{p}$,
$[p_i^{\sigma,\tau}] := \mathbf{p}_i$ $(i = 0, 1)$,
$[r^\sigma] := rec$.

Note that just like in the intensional case the $\mathbf{k}, \mathbf{s}, \mathbf{p}, \mathbf{p}_i$ are in the appropriate E_σ. For the example of $\mathbf{k}$ we now have to add a few things to show that $\forall\sigma, \tau.\mathbf{k} \in E_{\sigma\to(\tau\to\sigma)}$:
$\forall x, x' \in E_\sigma((x \sim_\sigma x' \wedge \mathbf{k}@x\downarrow) \to (\mathbf{k}@x'\downarrow \wedge \mathbf{k}@x \sim_{\tau\to\sigma} \mathbf{k}@x'))$. $\mathbf{k}@x$ is always defined so that leaves us to prove $\mathbf{k}@x \sim_{\tau\to\sigma} \mathbf{k}@x'$. This is $\forall y, y' \in E_\tau(y \sim_\tau y' \to \mathbf{k}@x@y\downarrow \wedge \mathbf{k}@x'@y'\downarrow \wedge)\mathbf{k}@x@y \sim_\sigma \mathbf{k}@x'@y'$. We know $\mathbf{k}@x@y = x$ so it is defined and $\mathbf{k}@x@y \sim_\sigma \mathbf{k}@x'@y'$ is $x \sim_\sigma x'$, which is also given.

In the definition of the extensional realizability the case of the existential quantifier seems a bit asymmetrical, since in the definition of $a =_{\exists x^\sigma.A(x)} b$ it is only required that $\mathbf{p}_0@a =_{A(\mathbf{p}_1@a)} \mathbf{p}_0@b$ and not $\mathbf{p}_0@a =_{A(\mathbf{p}_1@b)} \mathbf{p}_0@b$. This second requirement would be trivial because of the next lemma.

Lemma 4.2.5. *Given a formula $A(x_1^{\sigma_1}, \ldots, x_n^{\sigma_n})$ in the language of **HAP**. If $\mathbf{HAP} \vdash a =_{A(u_1^{\sigma_1},\ldots,u_n^{\sigma_n})} b \wedge \forall i(u_i \sim_{\sigma_i} v_i)$. Then $\mathbf{HAP} \vdash a =_{A(v_1^{\sigma_1},\ldots,v_n^{\sigma_n})} b$*

Proof. This can be proven by a simple induction on the structure of A. The least trivial step is $A = B \to C$.

$a =_{A(u_1,\ldots,u_n)} b$
is $a\ \mathbf{r}_e\ (B \to C)(u_1^{\sigma_1}, \ldots, u_n^{\sigma_n}) \wedge b\ \mathbf{r}_e\ (B \to C)(u_1^{\sigma_1}, \ldots, u_n^{\sigma_n}) \wedge$
$\quad \forall c(c\ \mathbf{r}_e\ B(u_1^{\sigma_1}, \ldots, u_n^{\sigma_n}) \to a@c =_{C(u_1^{\sigma_1},\ldots,u_n^{\sigma_n})} b@c)$
is $\forall d, d'(d =_{B(u_1^{\sigma_1},\ldots,u_n^{\sigma_n})} d' \to (a@d\downarrow \wedge a@d'\downarrow \wedge a@d =_{C(u_1^{\sigma_1},\ldots,u_n^{\sigma_n})} a@d'$
$\quad \wedge b@d\downarrow \wedge b@d'\downarrow \wedge b@d =_{C(u_1^{\sigma_1},\ldots,u_n^{\sigma_n})} b@d')$
$\quad \wedge \forall c(c\ \mathbf{r}_e\ B(u_1^{\sigma_1}, \ldots, u_n^{\sigma_n}) \to a@c =_{C(u_1^{\sigma_1},\ldots,u_n^{\sigma_n})} b@c))$
Then if $\forall i(u_i \sim_{\sigma_i} v_i)$ by induction
$\forall d, d'(d =_{B(v_1^{\sigma_1},\ldots,v_n^{\sigma_n})} d' \to$
$(a@d\downarrow \wedge a@d'\downarrow \wedge a@d =_{C(v_1^{\sigma_1},\ldots,v_n^{\sigma_n})} a@d' \wedge b@d\downarrow \wedge b@d'\downarrow \wedge b@d =_{C(v_1^{\sigma_1},\ldots,v_n^{\sigma_n})} b@d')$
$\quad \wedge \forall c(c\ \mathbf{r}_e\ B(v_1^{\sigma_1}, \ldots, v_n^{\sigma_n}) \to a@c =_{C(v_1^{\sigma_1},\ldots,v_n^{\sigma_n})} b@c))$
is $a =_{A(v_1,\ldots,v_n)} b$.

□

For step one we need to find for all the axioms A of $\mathbf{E}\text{-}\mathbf{HA}^\omega + \mathbf{AC}$ a realizer e such that $\mathbf{HAP}_\epsilon \vdash e\ \underline{\mathbf{r}}_e\ [A]$.

For the axioms of $\mathbf{HA}^\omega$ the same realizers can be used as in the proof of **??**. Now look at the axiom of choice. If e realizes **AC** we get

$e \underline{\mathbf{r}}_e [\forall x^\sigma \exists y^\tau A(x,y) \to \exists f^{\sigma\to\tau} \forall x^\sigma A(x, f(x))]$
this is $\forall b, b'(b =_{[\forall x^\sigma \exists y^\tau A(x,y)]} b' \to e@b\downarrow \wedge e@b' \downarrow \wedge e@b =_{[\exists f^{\sigma\to\tau} \forall x^\sigma A(x,f(x))]} e@b')$
this is $\forall b, b'(\forall x \Big(x \in E_\sigma \to$

$(\mathbf{p}_1@(b@x) \sim_\tau \mathbf{p}_1@(b'@x) \wedge \mathbf{p}_1@(b@x) \in E_\tau$

$\wedge\, \mathbf{p}_0@(b@x) =_{[A(x,\mathbf{p}_1@(b@x))]} \mathbf{p}_0@(b'@x)) \Big)$

$\to \Big(e@b\downarrow \wedge e@b'\downarrow \wedge \mathbf{p}_1@(e@b) \sim_{\sigma\to\tau} \mathbf{p}_1@(e@b') \wedge \mathbf{p}_1@(e@b) \in E_{\sigma\to\tau} \wedge$

$\forall x(x \in E_\sigma \to (\mathbf{p}_0@(e@b)@x =_{[A(x,\mathbf{p}_1@(e@b)@x)]} \mathbf{p}_0@(e@b')@x'))\Big)$.

Use $e = \lambda b.(\lambda x.\mathbf{p}_0@(b@x), \lambda x.\mathbf{p}_1@(b@x))$.
If you fill in this e then $\mathbf{p}_1@(e@b) \sim_{\sigma\to\tau} \mathbf{p}_1@(e@b')$ becomes $\lambda x.\mathbf{p}_1@(b@x) \sim_{\sigma\to\tau} \lambda x.\mathbf{p}_1@(b'@x)$. This follows precisely from $\forall x(x \in E_\sigma \to \mathbf{p}_1@(b@x) \sim_\tau \mathbf{p}_1@(b'@x) \wedge \mathbf{p}_1@(b@x) \in E_\tau)$
Secondly $\mathbf{p}_1@(e@b) \in E_{\sigma\to\tau}$ becomes $\lambda x.\mathbf{p}_1@(b@x) \in E_{\sigma\to\tau}$. This follows also from $\forall x(x \in E_\sigma \to \mathbf{p}_1@(b@x) \sim_\tau \mathbf{p}_1@(b'@x) \wedge \mathbf{p}_1@(b@x) \in E_\tau)$.
Lastly, if you fill in this e then $\mathbf{p}_0@(e@b)@x =_{[A(x,\mathbf{p}_1@(e@b)@x)]} \mathbf{p}_0@(e@b')@x$ becomes exactly

$$\mathbf{p}_0@(b@x) =_{[A(x,\mathbf{p}_1@(b@x))]} \mathbf{p}_0@(b'@x').$$

So for this e it is valid that $e \underline{\mathbf{r}}_e [\forall x^\sigma \exists y^\tau A(x,y) \to \exists f^{\sigma\to\tau} \forall x^\sigma A(x, f(x))]$.

This leaves the extensionality axiom.

$$\mathbf{E} : \forall y^{\sigma\to\tau}, z^{\sigma\to\tau}(\forall x^\sigma(yx =_\tau zx) \to y =_{\sigma\to\tau} z).$$

If e realizes the extensionality axiom we get

$e \underline{\mathbf{r}}_e [\forall y^{\sigma\to\tau}, z^{\sigma\to\tau}(\forall x^\sigma(yx =_\tau zx) \to y =_{\sigma\to\tau} z)]$
is $\forall y(y \in E_{\sigma\to\tau} \to (e@y\downarrow \wedge \forall z(z \in E_{\sigma\to\tau} \to (e@y@z\downarrow \wedge \forall b, b'(\forall x$

$\Big(x \in E_\sigma \to (e@y@z@b@x = e@y@z@b'@x' \wedge y@x \sim_\tau z@x)\Big)$

$\to \Big(e@y@z@b\downarrow \wedge e@y@z@b'\downarrow \wedge e@y@z@b = e@y@z@b' \wedge y \sim_{\sigma\to\tau} z\Big)))))$.

So we have $y, z \in E_{\sigma\to\tau}$, $x \in E_\sigma$ $y@x \sim_\tau z@x$ to prove $y \sim_{\sigma\to\tau} z$.
$y, z \in E_{\sigma\to\tau} \to \forall a, b \in E_\sigma(a \sim_\sigma b \wedge y@a\downarrow \wedge z@a\downarrow \to (y@b\downarrow \wedge z@b\downarrow \wedge y@a \sim_\tau y@b \wedge z@a \sim_\tau z@b))$ So $\forall a, b \in E_\sigma(a \sim_\sigma b \to y@a \sim_\tau y@b \sim_\tau z@b)$. This is equivalent with $y \sim_{\sigma\to\tau} z$.
The last thing to check is whether $e@y@z@b = e@y@z@b'$, if we set $e = \lambda y, z, b, x.0$ then $e@y@z@b = \lambda x.0 = e@y@z@b'$ This means the extensional axiom is realized by $e = \lambda y, z, b, x.0$. This finishes step one.

For step two use lemma **??** and the fact that if A is arithmetic then $\exists e.e \underline{\mathbf{r}}_e [A] \leftrightarrow \exists e.e\, \mathbf{r}_e\, A$. Then we have

$$\mathbf{HAP}_\epsilon \vdash \exists e.e \underline{\mathbf{r}}_e [A] \leftrightarrow \exists e\, e\, \mathbf{r}_e\, A \leftrightarrow A.$$

□

Chapter 5

Literature

In 1976 Nicolas Goodman introduced his theorem. Over the years Goodman and several other people worked on this theorem and on some extensions. In this chapter the ideas of the mathematicians who have worked on this problem will be presented.

5.1 Nicholas Goodman

Goodman started in [**?**] with a proof based on the interpretation of $\mathbf{HA}^\omega$ in his arithmetic theory of constructions (ATC). He had already shown that ATC is conservative **HA** via an argument resembling a bit of forcing. Nowadays ATC is no longer used, so we will not get into ATC here. With this interpretation he did use a technique which resembles realizability quite a bit. He assigns a term $|\phi|$ to every ϕ. And intuitively $|\phi|y \equiv \mathtt{T}$ means that y is a proof of ϕ. This does remind us of realizability. The difference is mostly in the definitions of $|\phi \vee \psi|y \equiv \mathtt{T}$ and $|\exists x.\phi|y \equiv \mathtt{T}$. Other than that all the definitions Goodman uses are similar to our definition of realizability, but written down more complicated with a lot more variables and constants, which makes it a lot less pleasant to read.

In 1978 Goodman wrote another paper [**?**] in which he addresses his theorem. This time his paper was mostly about a new kind of realizability. He calls it relativised realizability, this is a combination of realizability and forcing, also he includes types. Goodman gave an intuitive explanation for his notion. It is a notion in which he uses indices of partial recursive functions relative to a partial function p. This p is thought of as growing in time. Then let T be a set of partial functions, the partial ordering of T will play the role of a partially ordered set in a Kripke structure. Then we do not attach a model to each node, but rather realize sentences by indices of partial functions, recursive relative to the functions in T. This means that by building these p in time we are building our mathematical ability. So it is a combination of classical realizability, and forcing conditions in T which allowed Goodman to prove his theorem again.

5.2 Michael Beeson

In 1979 Michael Beeson wrote an article [**?**] in which he also proved Goodman's theorem and mentioned some extensions of the theorem like the extensional case. Later he wrote a book [**?**] in which both were incorporated again. His method was the one which is the most similar to the method which was used here and was a natural consequence of Goodman's proof. He made two separate definitions, one for realizability and one for forcing. His plan of attack was the following. First prove $\mathbf{HA} \vdash A \Rightarrow \mathbf{HA} \vdash e\ \mathbf{r}\ A$ and extend this notion to $\mathbf{HA}^\omega$ by using hereditarily recursive operations (HRO). Then the axiom of choice is realized, so what is left to prove is $\mathbf{HA}^\omega \vdash e\ \mathbf{r}\ A \to A$. It is easy to prove that $\mathbf{HA}^\omega$ is conservative over $\mathbf{HA}$, see chapter **??**. The only thing that has to be changed in chapter **??** is that we need to use the interpretation in HRO instead of PRO. The problem with this paper is that Beeson leaves out a lot of the details, he explains no more then was stated here except for the proof of $\mathbf{HA}^\omega \vdash e\ \mathbf{r}\ A \to A$. Also in that proof some details are left out as will be mentioned later. It did leave a good framework to work with and get inspiration from for our proof. A big difference is that we did not use realizability and forcing together in the same proof. We either used realizability or forcing to prove something. Where Beeson introduced a theorem using forcing to prove something about realizers, we broke our proof into parts introducing $\mathbf{HAP}$, $\mathbf{HAP}_\epsilon$ and $\mathbf{HAP}_E$. This gave us a new theory $\mathbf{HAP}_\epsilon$ in which all arithmetical formulas are self-realising. This also made every step a lot simpler to understand, when there is always just one thing going on at the time.

To prove $\mathbf{HA}^\omega \vdash e\ \mathbf{r}\ A \to A$ Beeson introduces forcing, a partial function a (which is to act as an oracle) and the system $\mathbf{HA}^\omega a$. This partial function is supposed to be generic with respect to a suitable set of forcing conditions and in $\mathbf{HA}^\omega a$, a added to $\mathbf{HA}^\omega$. What that means and how it should work remains a bit vague. We tried to make this precise in chapter **??**. Furthermore he proved Goodman's theorem by stating that if A is arithmetic then $\mathbf{HA}^\omega \vdash (A \leftrightarrow p\ \mathbf{f}\ A)$ and $\mathbf{HA}^\omega a + \mathbf{AC} \vdash A \Rightarrow \mathbf{HA}^\omega a \vdash \exists e.e\ \mathbf{r}\ A$. With his key lemma to proving Goodman's theorem he did work out the details, the lemma says:

> Fix an arithmetical A. Then there is a set C of forcing conditions such that $\mathbf{HA}^\omega \vdash \forall p \exists q \subset p(q\ \mathbf{f}\ (e\ \mathbf{r}\ A \to A))$.

In his proof for this lemma he defined the forcing conditions and used the canonical realizers to prove his lemma. In his proof it is hidden that he proves that the arithmetical formula A he fixed is self-realising. Then putting this all together he gets his proof of Goodman's theorem: Fix A arithmetic with $\mathbf{HA}^\omega + \mathbf{AC} \vdash A$. This gives $\mathbf{HA}^\omega a \vdash e\ \mathbf{r}\ A$ for some e. This again gives $\mathbf{HA}^\omega \vdash \exists p(p\ \mathbf{f}\ (e\ \mathbf{r}\ A))$. Then by his key lemma $\mathbf{HA}^\omega \vdash \exists q(q\ \mathbf{f}\ A)$, hence $\mathbf{HA}^\omega \vdash A$.

Beeson also states the extensional case, he explained that the problem is that we need to use a notion of realizability such that

1. A arithmetical formula still needs to be self-realising.

2. The extensional axiom and $\mathbf{AC}$ are both realised.

In order to make this both possible he states that one has to use extensional realisability. The only problem could arise in the implication, but with the

canonical realizers this will not be a problem. Trying to prove this ourselves, we came to the same conclusion that the extensional realisability is the best solution. A bonus with our proof was that when we wanted to prove the extensional variant, we only had to change the part, $\mathbf{HA}^\omega + \mathbf{AC}$ is conservative over $\mathbf{HAP}_\epsilon$, and the rest could just stay the same.

5.3 Gerard Renardel de Lavalette

In 1990 Gerard Renardel de Lavalette proved in his paper [**?**] that **TAPP** + **EAC** is conservative over **HA**. Here **TAPP** is a type-free theory with a total application. Just like in **APP** there are elements which are natural numbers and elements which are not. **EAC** is the following choice principle

$$\forall x(A(x) \to \exists y.B(x,y)) \to \exists f \forall x(A(x) \to B(x, fx)), \quad \text{here is } A \text{ negative .}$$

At this stage we introduced our ϵ and the axioms that come with that. Lavalette did something similar. He defined a theory $\mathbf{TAPP}_\epsilon$ by adding some constants ϵ_A and the following axiom schema $\epsilon AX(A)$ for every arithmetical formula $A(m,n)$ to **TAPP**

$$\forall m(\exists n.A(m,n) \to \exists n(A(m,n) \wedge n = \epsilon_A m)).$$

Note that the m, n are natural numbers. So $\forall n.A(n)$ actually means $\forall n(n \in \mathbb{N} \to A(n))$. This new theory $\mathbf{TAPP}_\epsilon$ also turned out te be a theory where all arithmetical formula are self-realising, since he proved that $\mathbf{TAPP}_\epsilon \vdash A \leftrightarrow \exists x.x\ \mathbf{r}\ A \leftrightarrow \tau_A\ \mathbf{r}\ A$. Here the terms τ_A are his canonical realizers, which look a bit different but are similar to the canonical realizers presented in our proof. This allowed him to prove that for arithmetical A $\mathbf{TAPP} + \mathbf{EAC} \vdash A \Rightarrow \mathbf{TAPP}_\epsilon \vdash A$. Of course now the thing to prove is that for arithmetical A, $\mathbf{TAPP}_\epsilon \vdash A \Rightarrow \mathbf{TAPP} \vdash A$. But if $\mathbf{TAPP}_\epsilon \vdash A$, then a finite amount of axioms of the form ϵ_A have been used in the proof. So also $\mathbf{TAPP} + \epsilon AX(B) \vdash A$ where $B = (n = 0 \wedge A_0) \vee \ldots \vee (n = k \wedge A_k)$. Call this new theory $\mathbf{TAPP}_\epsilon(B)$. To eliminate the ϵ from this theory he defined forcing over a monoid M instead over a poset. A monoid M is defined as a formula of **TAPP** with the following conditions.

$\mathbf{TAPP} \vdash \lambda x.x \in M$,
$\mathbf{TAPP} \vdash f, g \in M \to \lambda x.f(gx) \in M$.

Here $x \in M$ is written for $M(x)$. The thing to do now is to prove the soundness of forcing as an interpretation of $\mathbf{TAPP}_\epsilon(B)$ in **TAPP**. Unfortunately that does not work, there is not a monoid to get both **TAPP** and $\epsilon AX(B)$ forced. The problem lies in quantification over terms containing ϵ. So also in this proof we need to go via a weaker theory $\mathbf{TAPP}_\epsilon(B)^-$. This weaker theory is $\mathbf{TAPP}_\epsilon(B)$, but with the axioms $\forall x.A \to A(t/x)$, $A(t/x) \to \exists x.A$ restricted to t not containing ϵ and the equality axioms and the axioms for the constants written with terms (which can contain ϵ) instead of variables. Proving that $\mathbf{TAPP}_\epsilon(B)$ is conservative over $\mathbf{TAPP}_\epsilon(B)^-$ is pretty easy. With the mapping ${}^\epsilon : \mathbf{TAPP}_\epsilon(B) \to \mathbf{TAPP}_\epsilon(B)^-$ where the only interesting definition is $x^\epsilon = x\epsilon$. It can be proven that $\mathbf{TAPP}_\epsilon(B)^- \vdash A^\epsilon \leftrightarrow A$ and $\mathbf{TAPP}_\epsilon(B) \vdash A \Rightarrow$

$\mathbf{TAPP}_\epsilon(B)^- \vdash A^\epsilon$. Now it is time to prove the soundness of forcing. This was done by Lavalette in two steps first for the theory $T_\epsilon = \mathbf{TAPP}_\epsilon(B)^- - \epsilon AX(B)$. Instead of doing this directly Lavalette chose to use an interpretation in an modal theory with modal logic. The second step was to define the monoid in order to get $\epsilon AX(B)$ forced. The monoid he used was

$$M_0 := \{f | \forall m(\forall x(fxm = xm) \vee \exists n(B(m,n) \wedge \forall x(fxm = n)))\}.$$

Here f can be seen as an approximation of ϵ_B, just like the forcing conditions we used were an approximation of E. With this monoid it can be proven that $\mathbf{TAPP} \vdash \exists p.p\ \mathbf{f}\ \epsilon AX(B)$. As a corollary this gives us that $\mathbf{TAPP} + \mathbf{EAC}$ is conservative over $\mathbf{TAPP}$.

5.4 Thierry Coquand

In 2012 Thierry Coquand presented a paper [**?**] to clarify the papers which were presented on Goodman's theorem so far. Only Coquand's proof is again a bit different from all of the other papers. An important difference is that by proving the theorem for an specific formula, his proof is written down almost like an algortihm. What is interesting about that, is that he writes down the specific formulas which need to be forced. Which are exactly the extra axioms from our system $\mathbf{HAP}_\epsilon$ for each subformula B, D from the specific formula A. Just like Gerard Renardel de Lavalette, Coquand used the system TAPP in between $\mathbf{HA}^\omega + \mathbf{AC}$ and $\mathbf{HA}$ to construct his proof.

5.5 Ulrich Kohlenbach

In 1997 Ulrich Kohlenbach puplished a paper [**?**] in which he proved a contrasting result. He proved that (**E**-)$\mathbf{HA}^\omega_- + \mathbf{AC}_{\mathrm{ar}}$ is not conservative over $\mathbf{HA}_-$, where (**E**-)$\mathbf{HA}^\omega{}_-$ is (**E**-)$\mathbf{HA}^\omega$ with the restriction to quantifier-free induction and $\mathbf{HA}_-$ is $\mathbf{HA}$ with the restriction to quantifier-free induction. The schema of quantifier-free induction:

$$\mathbf{QF\text{-}IND}: \ A \wedge \forall x^0(A(x) \to A(Sx)) \to \forall x^0 A(x).$$

where A is quantifier free, instead of the constants of (**E**-)$\mathbf{HA}^\omega$ we only have 0^0 and symbols for every primitive recursive function plus their defining equations. $\mathbf{AC}_{ar}$ is the arithmetical axiom of choice:

$$\forall x^0 \exists y^0 A(x,y) \to \exists f^{0(0)} \forall x^0 A(x, fx).$$

where A contains only quantifiers of type 0 and $\mathbf{HA}^\omega_- + \mathbf{AC}_{ar} - qf$ is $\mathbf{HA}^\omega_- + \mathbf{AC}_{ar}$ with the restriction for A to the quantifier-free formulas.

We will give a overview of his proof for $\mathbf{HA}^\omega_- + \mathbf{AC}_{ar} - qf$ is not conservative over $\mathbf{HA}_-$. We will do this be showing that there is an $\tilde{A}$ such that $\mathbf{HA}^\omega_- + \mathbf{AC}_{ar} - qf \vdash \tilde{A}$ and $\mathbf{PA}_- \nvdash \tilde{A}$ and hence $\mathbf{HA}_- \nvdash \tilde{A}$.

We will start with the fact (see [**?**]) that there is an instance of the collection principle

$$\tilde{A} := \forall a, \bar{b}(\forall x < a \exists y. A_0(x,y,a,\bar{b}) \to \exists y_0 \forall x < a \exists y < y_0. A_0(x,y,a,\bar{b})).$$

where $A \in \Pi_0^0$ of the arithmetical hierarchy and is quantifier free, such that $\mathbf{PA}_{\_} \nvdash \tilde{A}$.
At the same time we do have $\mathbf{HA}_{\_}^{\omega} + \mathbf{AC}_{ar} \vdash \tilde{A}$:
First we will fill in $\forall x < a \exists y.A(x,y)$ in the arithmetical axiom of choice this gives a function f such that $\forall x < a A(x, fx)$. Now we can construct a y_0 by putting $y_0 = max(f0, \ldots, f(a-1)) + 1$. Except of course when $a = 0$ then $y_0 = f0 + 1$.

Now in $\mathbf{HA}_{\_}$ $\tilde{A}$ can be rearranged to the form

$$\forall a(\forall x \exists y.A_0(x,y,a) \rightarrow \exists y.B_0(y,a)).$$

Here are A_0, B_0 quantifier-free as well. Now the expression of the axiom of choice needed to prove $\tilde{A}$ in $\mathbf{HA}_{\_}^{\omega} + \mathbf{AC}_{ar}$ is quantifier-free, since A_0 is quantifier-free.
So $\mathbf{HA}_{\_}^{\omega} + \mathbf{AC}_{ar} - qf \vdash \tilde{A}$ and $\mathbf{PA}_{\_} \nvdash \tilde{A}$. This completes the proof.

His proof is completely different from the proofs mentioned so far.

Appendix A

Appendix

A.1 Partial function symbols

We have a language $\mathcal{L}$ a system T and a $n+1$-ary relation symbol F. And the axiom in the system T

$$\forall x_1, \ldots, x_n, y, y'.F(\vec{x}, y) \wedge F(\vec{x}, y') \rightarrow y = y'.$$

We will construct a way to use a n-ary partial function symbol f instead of the relation symbol F. First define semiterms; these are the constants and variables of $\mathcal{L}$ closed under the function symbols of $\mathcal{L}$ and f. These are terms in the language $\mathcal{L}' = \mathcal{L} \cup f$. Now define for all the semiterms t a formula $D_t(\vec{x}, y)$ in $\mathcal{L}$ with free variables $\vec{x}, y$. We can read this as t is defined on $\vec{x}$ as y. Here $FV(t) \subseteq \{x_1, \ldots, x_n\}$.

$D_c(y) := c = y,$
$D_x(x, y) := x = y,$
$D_{g(t_1,\ldots,t_m)}(\vec{x}, y) = \exists z_1, \ldots, z_m.D_{t_1}(\vec{x}, z_1) \wedge \ldots \wedge D_{t_m}(\vec{x}, z_m) \wedge y = g(z_1, \ldots, z_m),$
g is a function symbol in $\mathcal{L}$
$D_{f(t_1,\ldots,t_n)}(\vec{x}, y) = \exists z_1, \ldots, z_n.D_{t_1}(\vec{x}, z_1) \wedge \ldots \wedge D_{t_n}(\vec{x}, z_n) \wedge F(z_1, \ldots, z_n, y).$

Then we define for a semiterm t, $t\downarrow := \exists y.D_t(\vec{x}, y)$.
We translate the formulas $\phi \in \mathcal{L}'$ by defining a map $(\cdot)^*$. For this map ϕ means intuitively the same as $\phi^* \in \mathcal{L}$. We just have to give the definition of this map for the atomic formulas since the map leaves the logical operators the same.

$$\begin{aligned} \phi^* &\longmapsto \phi, \\ R(t_1, \ldots, t_n)^* &\longmapsto \exists z_1, \ldots, z_n.D_{t_1}(\vec{x}, z_1) \wedge \ldots \wedge D_{t_n}(\vec{x}, z_n) \wedge R(z_1, \ldots, z_n), \\ (s = t)^* &\longmapsto \exists y.D_s(\vec{x}, y) \wedge D_t(\vec{x}, y). \end{aligned}$$

This also means that if we have a map which projects terms on semiterms. Say $t \longmapsto [t]$, where t is a term in $\mathcal{L}$ and $[t]$ is a semiterm. Then this map extends to formulas; $\phi \longmapsto [\phi]$ with $\phi \in \mathcal{L}$, $[\phi] \in \mathcal{L}'$. Just like above, there is for every $[\phi] \in \mathcal{L}'$ a $[\psi]^* \in \mathcal{L}$ such that the intuitively mean the same.

A.2 Partial Combinatory Algebras

This section is about Partial Combinatory Algebras (PCA). This notion was defined by Feferman in 1975 [**?**], he generalized the notion of total combinatory algebras which was introduced by M. Schönfinkel around 1920. We can think of a PCA as a model of a computation. The general idea is to consider a set A with an application for every pair of elements $a, b \in A$, $a \cdot b$. This application can of course be partial. See [**?**] for more information.

A.2.1 Basic Definitions

Definition A.2.1. *A partial applicative structure (pas) is a pair $(A, \cdot)$. Where A is a nonempty map and we have a partial map from $A \times A$ to A denoted by $(a, b) \mapsto a \cdot b$. This map is called the application.*

Since it is shorter we will often just write ab instead of $a \cdot b$. The application does not have to be associative. So to avoid a lot of brackets we will adopt the usual association to the left. Which means we write $(ab)c$ as abc.

Let $(A, \cdot)$ be a pas and V a set of infinite variable. The set $E(A)$ of terms over A is the least set such that the following holds:

1. $A \subseteq E(A)$
2. $V \subseteq E(A)$
3. if $s, t \in E(A)$ then $ts \in E(A)$

We will write $ab\downarrow$ to say ab denotes or is defined.

Definition A.2.2. *A pas $(A, \cdot)$ is combinatory complete if for any $n \in \mathbb{N}$ and any term $t(x_1, \ldots, x_{n+1})$ there is an element $a \in A$ such that for all $a_1, \ldots, a_{n+1} \in A$ the following holds:*

1. *$aa_1 \cdots a_n\downarrow$*
2. *$aa_1 \cdots a_{n+1} \backsimeq t(a_1, \ldots, a_{n+1})$*

Where $t \backsimeq s$ means that if t or s denotes then they both denote and $t = s$.

A pas is a PCA if it os combinatory complete.

Theorem A.2.3. *Let $(A, \cdot)$ be a pas. A is a PCA if and only if there are $k, s \in A$ satisfying for all $a, b, c \in A$:*

1. *$sab\downarrow$*
2. *$kab = a$*
3. *$sabc \backsimeq ac(bc)$*

Proof. " $\Rightarrow$ " Choose for k, s an element satisfying definition **??** for the terms $t(x_1, x_2) = x_1$ and $t(x_1, x_2, x_3) = x_1x_3(x_2x_3)$ respectively. Then 2) is clear, 1) follows from 1) in and 3) also holds. " $\Leftarrow$ " Assume k, s satisfying 1)-3) of the theorem. Then we can just as in section 3.3 have λ-abstraction defined by induction on the terms t.

Then we can set $a = \lambda x_1, \ldots, x_{n+1}.t$. □

A.2.2 Kleene's first model

The best known PCA is Kleene's first model ($\mathcal{K}_1$). It is the set $\mathbb{N}$ with partial recursive application. The partial recursive application is $a, b \mapsto \{a\}(b)$.